National University of Engineering (UNI)

School of Sciences

School of Computer Science



Book of Syllabi

– 2027-I–

Lima: October 20, 2026

Equipo de Trabajo

Dr. Ciro Nuñez Iturri Director (e) de la Escuela de Ciencia de la Computación email: *ciro.nunez.i@uni.edu.pe*	**Dr. Carlos Javier Solano Salinas** Director del Departamento de Ciencia de la Computación e-mail: *jsolano@uni.edu.pe*
Dr. José Luis Segovia-Juárez PhD en Ciencia de la Computación. Wayne State University (Detroit, USA) (2001) Profesor del Departamento de Ciencia de la Computacion. UNI email: *jsegoviaj@uni.edu.pe* *http://www.tecnociencia9.com/jlsegovia/es*	**Dr. Marcos Antonio Alania Vicente** PhD en Ciencias Universidad de Amberes. Bélgica (2017) Profesor del Departamento de Ciencia de la Computacion. UNI Desarrollador de algoritmos de IA para procesos complejos interdisiplinarios email: *alania.vicente@gmail.com*
MSc. Eduardo Yauri Lozano MSc. en Ciencia de la Computación. UNI (2024) Profesor del Departamento de Ciencia de la Computacion. UNI email: *eduardo.yauri@uni.edu.pe*	**MSc. Ronald Ricardo Martinez Chunga** MSc. Ciencias Ing Electrónica (2021) Profesor del Departamento de Ciencia de la Computacion. UNI email: *rmartinezch@uni.edu.pe* *https://www.linkedin.com/in/rmartinezchunga/*
Cesar Martin Cruz-Salazar Mag(c) en Automática e Instrumentación, UNI Profesor de Ciencia de la Computación, UNI, Lima-Perú email: *ccruz@uni.edu.pe*	**Prof. Miguel Arrunategui Angulo** Docente del Departamento de Ciencia de la Computación email: *garrunateguia@uni.edu.pe*
Mag. Yuri Ccoicca Departmento de Ciencia de la Computación, UNI, Lima-Perú email: *yccoicca@uni.edu.pe*	**M.Sc. Raul Huarote Zegarra** Mag en Ciencia de la Computación, UNT (2014) Profesor del Departamento de Ciencia de la Computación, UNI, Lima-Perú email: *rhuarotez@uni.edu.pe*
M.Sc. Carlos Alberto Bazán Cabanillas Magister en Ciencias, Mención matemáticas, UNI 1980 Licenciado en Matemáticas, UNI 1977 email: *cbazanc@uni.edu.pe*	**Dr. Jose Alfredo Herrera-Quispe** Doctor en Ciencia de la Computación, UNSA (2013) Docente del Departamento de Ciencia de la Computación

Mag. Sidney Weisser Valer Quispe
Maestro en Ciencia con mención en Telemática, FIEE-UNI (2026)
Docente del Departamento de Ciencia de la Computación
email: *svalerq@uni.edu.pe*

Mag. Johnny Quispe Obregón
Docente del Departamento de Ciencia de la Computación
email: *johnny.quispe.o@uni.edu.pe*

Dr. Ernesto Cuadros-Vargas
Dr. en Ciencia de la Computación, Universidad de Sao Paulo, Brasil (2004)
Orador distinguido para la *Association of Computing Machinery* (ACM) (2025-2027)
Miembro del Directorio de Gobernadores de la Sociedad de Computación del IEEE (2020-2023)
Miembro del *Steering Committee ACM/IEEE-CS Computing Curricula 2020 (CS2020)*
Miembro del *Steering Committee ACM/IEEE-CS Computing Curricula/Computer Science (CS2013)*
Secretario Ejecutivo del Centro Latinoamericano de Estudios en Informática (CLEI) 2008-2016
email: *ecuadros@spc.org.pe*

Contents

First Semester **7**
1.1 BIC101. Introduction to Computing . . . 7
1.2 BMA101. Linear Algebra . . . 13
1.3 BMA102. Differential Calculus . . . 16
1.4 BCH101. Chemistry I . . . 19
1.5 BEI101. Technical and professional English I . . . 25

Second Semester **29**
2.1 CS100. Introduction to Computer Science . . . 29
2.2 CS112. Objects-oriented programming I . . . 46
2.3 CS1D1. Discrete Structures . . . 53
2.4 BMA103. Integral Calculus . . . 56
2.5 BFI101. Physics I . . . 59
2.6 BEI102. Technical and professional English II . . . 66

Third Semester **71**
3.1 CS113. Objects-oriented programming II . . . 71
3.2 CS2B1. Platform Based Development . . . 80
3.3 AI161. Applied AI . . . 86
3.4 BMA104. Advanced Differential and Integral Calculus . . . 90
3.5 STA251. Probability Calculation . . . 94
3.6 BEI201. Technical and professional English III . . . 100

Fourth Semester **105**
4.1 CS210. Algorithms and Data Structures . . . 105
4.2 CS211. Theory of Computation . . . 111
4.3 CS221. Computer Systems Architecture . . . 119
4.4 CS271. Data Management . . . 125
4.5 CS401. Research Methodology . . . 131
4.6 MA202. Numerical Methods . . . 136
4.7 BEI202. Technical and professional English IV . . . 142

Fifth Semester **147**
5.1 CS212. Analysis and Design of Algorithms . . . 147
5.2 CS261. Artificial Intelligence . . . 155
5.3 CS272. Databases II . . . 168
5.4 CS291. Software Engineering I . . . 173
5.5 CS2S1. Operating systems . . . 182

5.6 FG106. Theater . . . 189
5.7 BEI203. Technical and professional English V . . . 194

Sixth Semester **199**
6.1 CS231. Networking and Communication . . . 199
6.2 CS311. Algorithms for Complex Problems . . . 205
6.3 CS312. Advanced Data Structures . . . 209
6.4 CS342. Compilers . . . 213
6.5 AI263. Introduction to Machine Learning . . . 218
6.6 FI201. Computational Physics . . . 221

Seventh Semester **224**
7.1 CS251. Computer Graphics . . . 224
7.2 CS292. Software Engineering II . . . 228
7.3 CS2H1. User Experience (UX) . . . 236
7.4 AI264. Deep Learning . . . 243
7.5 FG211. Professional Ethics . . . 246
7.6 FG350. Leadership and Performance . . . 249

Eighth Semester **253**
8.1 CS281. Computing in Society . . . 253
8.2 CS3I1. Computer Security . . . 260
8.3 CS3P1. Parallel and Distributed Computing . . . 268
8.4 CS402. Capstone Project I . . . 284
8.5 EX301. Extracurricular Activities . . . 288
8.6 ET201. Entrerpreneurship I . . . 290
8.7 AI268. Computational Vision . . . 294
8.8 CS391. Software Engineering III . . . 304
8.9 CS393. Information systems . . . 311
8.10 CS394. Software Architecture and DevOps for Cloud Software . . . 314
8.11 CS395. Software Quality . . . 323

Ninth Semester **333**
9.1 CS370. Big Data . . . 333
9.2 CS400. Pre-professional internships . . . 337
9.3 CS403. Capstone Project II . . . 339
9.4 AI365. Advanced Generative AI Models . . . 343
9.5 CB309. Bioinformatics . . . 346
9.6 ET301. Entrerpreneurship II . . . 350
9.7 AI369. Robotics . . . 353
9.8 CS351. Topics in Computer Graphics . . . 358
9.9 CS353. Quantum Computing . . . 363
9.10 CS392. Tópicos en Ingeniería de Software . . . 367
9.11 CS3P3. Internet of Things . . . 378

Tenth Semester **381**
10.1 CS3P2. Cloud Computing . . . 381
10.2 CS404. Research Workshop . . . 387
10.3 AI367. Topics in Artificial Intelligence . . . 391
10.4 AI368. Evolutionary Computing . . . 403

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BIC101. Introduction to Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BIC101. Introduction to Computing
2.2 Semester	:	1st Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the first course in the sequence of introductory courses to Computer Science. This course introduces participants to the fundamental concepts of programming using Python. Topics include data types, control structures, functions, lists, recursion, and the mechanics of execution, testing, and debugging. Programming is one of the pillars of Computer Science; any professional in the area will need to program to materialize their models and proposals.

5. GOALS

- Introduce the fundamental concepts of programming.
- Develop the ability of abstraction using a programming language.
- Develop problem-solving skills through algorithmic thinking.
- Understand basic data structures and their applications.
- Master the Python programming language for problem solving.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Familiarity)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Introduction to Programming and History (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Brief history of computing and programming languages. • Overview of Python programming language. • Programming environments and tools. • Basic program structure and execution. • Writing, running, and debugging simple programs.	• Identify important trends in the history of computing. [Familiarizarse] • Discuss the historical context of programming language paradigms. [Familiarizarse] • Compare daily life before and after personal computers and the Internet. [Evaluar] • Write, compile, and run a simple Python program. [Usar]
Readings : [**Zelle2010**], [**Guttag2013**], [**Brooks2019**]	

Unit 2: Type Systems Fundamentals (6 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• A type as a set of values together with a set of operations: – Primitive types (e.g., numbers, Booleans) – Compound types built from other types (e.g., records/structs, unions, arrays, lists, functions, references using set operations) • Type safety and errors caused by using values inconsistently given their intended types • Goals and limitations of static and dynamic typing: detecting and eliminating errors as early as possible. • Primitive data types in Python (int, float, str, bool). • Type conversion and casting.	• Describe, for both a primitive and a compound type, the values that have that type [Familiarizarse] • Describe examples of program errors detected by a type system [Familiarizarse] • Use types and type-error messages to write and debug programs [Usar] • Explain Python's dynamic typing system. [Familiarizarse]
Readings : [**Zelle2010**], [**Guttag2013**]	

Unit 3: Object-Oriented Programming Fundamentals (14 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Imperative programming as a subset of object-oriented programming. • Definition of classes: fields, methods, and constructors. • Dynamic vs static properties. • Variables, expressions, and assignments. • Input and output operations. • Conditional statements (if, elif, else). • Iteration structures (for loops, while loops). • Nested loops and loop control statements. • Exception handling basics.	• Compose a class through design, implementation, and testing to meet behavioral requirements [Usar] • Predict and validate control flow in a program using dynamic dispatch [Evaluar] • Use object-oriented encapsulation mechanisms such as interfaces and private members [Usar] • Implement programs using conditional and iterative structures. [Usar] • Debug programs with syntax and logical errors. [Usar] • Design and implement simple algorithms using loops. [Usar]
Readings : [**Zelle2010**], [**Guttag2013**], [**Brooks2019**]	

Unit 4: Functions and Parameter Passing (10 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Function definition and invocation. • Parameter passing mechanisms (by value, by reference). • Return values and multiple returns. • Variable scope and lifetime (local, global, nonlocal). • Recursion fundamentals. • Lambda expressions and higher-order functions. • Functions as first-class objects. • Documentation and testing of functions.	• Define and implement functions with proper parameter passing. [Usar] • Explain the difference between pass-by-value and pass-by-reference. [Familiarizarse] • Implement recursive solutions to simple problems. [Usar] • Use lambda expressions for functional programming constructs. [Usar] • Design functions with appropriate scope and documentation. [Evaluar]
Readings : [**Zelle2010**], [**Guttag2013**], [**Pragmatic2009**]	

Unit 5: Basic Data Structures (10 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Lists and list operations (indexing, slicing, mutability). • Tuples and immutability. • Dictionaries and key-value mapping. • Sets and set operations. • Strings as sequences. • List comprehensions and generator expressions. • Basic searching and sorting algorithms. • Introduction to algorithmic complexity for basic operations.	• Implement programs using lists, dictionaries, and sets. [Usar] • Choose appropriate data structures for given problems. [Evaluar] • Explain the trade-offs between different data structures. [Familiarizarse] • Implement basic search and sort operations on lists. [Usar]
Readings : [**Zelle2010**], [**Guttag2013**]	

Unit 6: Object-Oriented Programming Fundamentals (10 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Subclasses, inheritance (including multiple inheritance), and method overriding. • Dynamic dispatch: definition of method-call. • Object-oriented idioms for encapsulation: – Privacy, data hiding, and visibility of class members. – Interfaces revealing only method signatures. – Abstract base classes, traits and mixins. • Classes and objects in Python. • Attributes and methods. • Inheritance and polymorphism basics. • Encapsulation and information hiding.	• Build a simple class hierarchy utilizing subclassing that allows code to be reused for distinct subclasses [Usar] • Explain the relationship between object-oriented inheritance (code-sharing and overriding) and subtyping (the idea of a subtype being usable in a context that expects the supertype) [Familiarizarse] • Compare and contrast the benefits and costs/impact of using inheritance (subclasses) and composition (specifically, how to base composition on higher order functions) [Evaluar] • Design and implement simple classes in Python. [Usar] • Explain the benefits of OOP for code organization. [Familiarizarse]
Readings : [**Zelle2010**], [**Guttag2013**]	

Unit 7: Algorithmic Strategies (6 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Paradigms: – Brute-Force (e.g., linear search, selection sort, traveling salesperson, knapsack) – Decrease-and-Conquer: ∗ By a Constant (e.g., insertion sort, topological sort) ∗ By a Constant Factor (e.g., binary search) ∗ By a Variable Size (e.g., Euclid's) – Divide-and-Conquer (e.g., binary search, quicksort, mergesort, Strassen's) – Greedy (e.g., Dijkstra's, Kruskal's, Knapsack) – Transform-and-Conquer: ∗ Instance simplification (e.g., find duplicates via list presort) ∗ Representation change (e.g., heapsort) ∗ Problem reduction (e.g., least-common-multiple, linear programming) ∗ Dynamic programming (e.g., Floyd's, Marshall, Bellman-Ford) – Space vs time tradeoffs (e.g., hashing) • Iteration vs recursion (e.g., factorial, tree search) • Problem decomposition strategies. • Algorithm design techniques. • Debugging and testing strategies. • Tracing program execution.	• For each of the paradigms in this unit: – Explain its definitional characteristics – Explain an example that demonstrates the paradigm including how this example satisfies the paradigm's characteristics. [Familiarizarse] • Give examples of iterative and recursive algorithms that solve the same problem, explain the benefits and disadvantages of each approach [Usar] • Evaluate whether a greedy approach leads to an optimal solution [Evaluar] • Apply systematic debugging techniques to locate and fix errors. [Usar] • Analyze algorithm complexity for simple problems. [Familiarizarse]

Readings : [**Guttag2013**], [**Cormen2009**]

Unit 8: Modern Development Methods and Tool Usage (4 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Using modern programming environments (IDEs, notebooks). • Code search and reuse best practices. • Using standard libraries and APIs. • Debugging and testing techniques. • Version control basics. • Code documentation and style guidelines. • Virtual environments and package management.	• Use modern development tools effectively. [Usar] • Apply code reuse principles through libraries and APIs. [Usar] • Debug and test programs systematically. [Usar] • Follow coding standards and documentation practices. [Familiarizarse] • Manage Python packages and environments. [Familiarizarse]

Readings : [**Zelle2010**], [**Pragmatic2009**]

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BMA101. Linear Algebra (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BMA101. Linear Algebra
2.2 Semester	:	1st Semester.
2.3 Credits	:	4
2.4 Hours	:	3 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Linear algebra is a fundamental mathematical tool shared by computer science and engineering disciplines, providing the foundations for structural analysis, circuit networks, control systems, data processing, and algorithm design. This course provides a solid foundation in the concepts and techniques of elementary and advanced linear algebra, with emphasis on its application in scientific and engineering contexts.

5. GOALS

- Understand the fundamental concepts of linear algebra, including vector spaces, linear transformations, and matrix theory.
- Apply linear algebra techniques to solve systems of equations and eigenvalue/eigenvector problems arising from scientific and engineering contexts.
- Develop skills in abstract reasoning and logical thinking to address mathematical problems.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

7. TOPICS

Unit 1: Matrices and Systems of Linear Equations (19 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Matrix algebra: addition, scalar multiplication, matrix multiplication, transpose, types of matrices, and the inverse of a matrix • Solving systems of linear equations via Gaussian elimination and Gauss-Jordan; row echelon form, rank, and consistency of systems • Determinants: definition, properties, Cramer's rule, and geometric interpretation • Matrix decompositions: LU, QR, and singular value decomposition (SVD)	• Identify types of matrices and perform basic matrix algebra operations, including computing the inverse of a matrix [Familiarity] • Solve systems of linear equations using Gaussian elimination or Gauss-Jordan, determining the rank and consistency of the system [Usage] • Apply Cramer's rule and the LU, QR, and SVD decompositions to solve linear systems and factor matrices [Assessment]
Readings : [**Strang2016**], [**Lay2016**]	

Unit 2: Vector Spaces and Linear Transformations (14 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Vector spaces and subspaces; bases, dimension, and linear independence • Linear maps, kernel and image, rank-nullity theorem, and matrix representations	• Identify whether a set with given operations forms a vector space and determine a basis [Familiarity] • Determine the kernel, image, and matrix representation of a linear map, verifying the rank-nullity theorem [Usage] • Apply linear transformations to model and solve problems involving change of basis and composition of maps [Assessment]
Readings : [**Strang2016**], [**Lay2016**]	

Unit 3: Eigenvalues, Inner Product Spaces, and Quadratic Forms (15 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Eigenvalues and eigenvectors; diagonalization and the spectral theorem • Inner product spaces, orthogonality, Gram-Schmidt process, and least squares • Quadratic forms and definiteness: classification as positive, negative, or indefinite via eigenvalues	• Identify the eigenvalues and eigenvectors of a matrix and determine whether it is diagonalizable [Familiarity] • Compute the diagonalization of matrices via the spectral theorem and apply the Gram-Schmidt process to obtain orthogonal bases [Usage] • Apply least squares methods and the classification of quadratic forms via eigenvalues to solve approximation and optimization problems [Assessment]
Readings : [**Strang2016**], [**Lay2016**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BMA102. Differential Calculus (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BMA102. Differential Calculus
2.2 Semester	:	1st Semester.
2.3 Credits	:	5
2.4 Hours	:	4 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Differential calculus is a fundamental mathematical tool shared by computer science and engineering disciplines, providing the foundations for the rigorous study of change, the analysis of rates of change, the optimization of systems and functions, and the modeling of continuous phenomena. This course provides a solid foundation in the concepts and techniques of differential calculus of one variable, with emphasis on its application in scientific and engineering contexts.

5. GOALS

- Rigorously understand the concepts of limit, continuity, and derivative of functions of one variable.
- Apply differentiation rules to compute derivatives of algebraic, composite, and implicit functions arising from scientific and engineering contexts.
- Use differential calculus to solve optimization problems, related rates problems, and function behavior analysis.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

7. TOPICS

Unit 1: Limits and Continuity (12 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Epsilon-delta definition of a limit and formal proof techniques • Limit laws, one-sided limits, and limits at infinity • Continuity: types, properties, and the Intermediate Value Theorem • L'Hôpital's rule and indeterminate forms • Limits of sequences: convergence, Squeeze Theorem, and Bolzano-Weierstrass	• State the epsilon-delta definition of a limit and explain its geometric meaning [Familiarity] • Evaluate limits using limit laws, L'Hôpital's rule, and asymptotic analysis [Usage] • Prove continuity or discontinuity of a function and apply the Intermediate Value Theorem [Assessment]
Readings : [**Stewart2015**], [**LarsonEdwards2014**]	

Unit 2: Differential Calculus (36 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Derivative as a limit; geometric and physical interpretations; differentiability • Product, quotient, and chain rules; implicit differentiation; related rates • Rolle's theorem, Mean Value Theorem, and monotonicity criteria • Higher-order derivatives, Taylor polynomials, and remainder estimation • Extrema: first and second derivative tests, curve sketching, and applied optimization	• Interpret the derivative as a rate of change and as the slope of the tangent line [Familiarity] • Apply differentiation rules and implicit differentiation to compute derivatives of composite and implicit functions [Usage] • Solve optimization problems and sketch curves using the Mean Value Theorem and derivative tests [Assessment]
Readings : [**Stewart2015**], [**LarsonEdwards2014**]	

Unit 3: Parametric Equations and Polar Coordinates (6 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Parametric curves and their calculus, applied to trajectories in kinematics • Polar coordinates and their conversion to Cartesian coordinates	• Calculate the velocity and acceleration of a trajectory described parametrically [Usage] • Convert between polar and Cartesian coordinates when modeling engineering geometries [Usage]
Readings : [**Stewart2016**], [**Thomas2014**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BCH101. Chemistry I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BCH101. Chemistry I
2.2 Semester	:	1st Semester.
2.3 Credits	:	5
2.4 Hours	:	4 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Chemistry provides the foundation for understanding the composition, structure, and properties of matter – knowledge required across computing, and engineering programs alike as part of a rigorous grounding in the natural sciences. This course introduces the basic principles of general chemistry and extends them to chemical equilibrium and electrochemistry, together with the applied chemistry of water, polymers, and cement, topics that recur in materials science, environmental analysis, and engineering practice.

5. GOALS

- Understand the structure of matter at the atomic and molecular level.
- Apply the principles of stoichiometry to perform chemical calculations.
- Understand the different types of chemical bonds and their influence on the properties of substances.
- Apply chemical equilibrium, including acid-base and solubility equilibria, to predict the behavior of aqueous systems.
- Apply electrochemical principles to analyze corrosion of metals and reinforced concrete.
- Relate the composition and properties of water, polymers, and cement to their applications in materials science and civil engineering.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

7. TOPICS

Unit 1: Matter and Energy (8 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Classification of matter: elements, compounds, homogeneous and heterogeneous mixtures • Physical and chemical properties, and physical and chemical changes of matter • International System of units, significant figures, and dimensional analysis • Forms of energy, heat and temperature; the law of conservation of mass and energy • The scientific method applied to chemistry: observation, hypothesis, and experimentation	• Classify a sample of matter as an element, compound, or mixture, and distinguish homogeneous from heterogeneous mixtures [Familiarity] • Apply dimensional analysis and significant-figure rules in calculations involving chemical measurements [Usage] • Relate observed energy changes to specific physical and chemical processes, applying the law of conservation of energy [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 2: Atomic Structure (10 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Historical evolution of the atomic model: Dalton, Thomson, Rutherford, and Bohr • Subatomic particles, atomic number, mass number, and isotopes • Electron configuration, quantum numbers, and the Aufbau principle • Organization of the periodic table: groups, periods, and blocks • Periodic trends: atomic radius, ionization energy, electron affinity, and electronegativity	• Describe the historical evolution of the atomic model and each model's contribution to the current understanding of the atom [Familiarity] • Determine the electron configuration of an atom or ion from its position in the periodic table [Usage] • Predict periodic trends (atomic radius, ionization energy) to compare two or more elements [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 3: Chemical Bonding (10 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Ionic bonding: formation, lattice energy, and properties of ionic compounds • Covalent bonding, Lewis structures, and the concept of resonance • Molecular geometry via Valence Shell Electron Pair Repulsion (VSEPR) theory • Electronegativity, bond polarity, and molecular polarity • Intermolecular forces: dipole-dipole, hydrogen bonding, and London dispersion forces	• Draw correct Lewis structures, including resonance cases, for molecules and polyatomic ions [Usage] • Predict the molecular geometry and polarity of a molecule by applying VSEPR theory [Usage] • Relate the type of intermolecular force present in a substance to its observed physical properties (boiling point, solubility) [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 4: Stoichiometry (10 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Chemical formulas, molar mass, and the mole concept • Balancing chemical equations and their quantitative interpretation • Mole-to-mole, mass-to-mass, and mass-to-mole stoichiometric calculations • Limiting reagent, excess reagent, and percent yield • Molarity, solution preparation, and solution stoichiometry	• Balance chemical equations by inspection, verifying conservation of mass [Familiarity] • Calculate the amounts of reactants and products in a chemical reaction using mole ratios [Usage] • Determine the limiting reagent and percent yield of a reaction from experimental data [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 5: States of Matter (10 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Gas laws: Boyle's, Charles's, Avogadro's, and the ideal gas equation • Kinetic-molecular theory of gases and its applications • Phase changes, heating curves, and phase diagrams • Crystalline and amorphous solids; types of crystalline solids • Colligative properties of solutions: vapor pressure lowering, boiling point elevation, freezing point depression, and osmotic pressure	• Apply the ideal gas equation and the gas laws to solve problems involving pressure, volume, and temperature [Usage] • Interpret a phase diagram to identify the states and transitions of a substance under different conditions [Familiarity] • Calculate the colligative properties of a solution (vapor pressure, boiling point, freezing point) from its concentration [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 6: Environmental and Water Chemistry (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Chemical composition of natural water and quality parameters (pH, hardness, dissolved oxygen, BOD) • Fundamentals of water treatment: coagulation, flocculation, and disinfection • Major air and water pollutants and their sources • Biogeochemical cycles of carbon, nitrogen, and phosphorus • Atmospheric chemistry and the greenhouse effect	• Interpret water quality parameters to assess fitness for use [Usage] • Describe the basic processes of drinking water and wastewater treatment [Familiarity] • Identify the main sources and effects of air and water pollutants [Familiarity] • Explain the chemical mechanism of the greenhouse effect [Usage]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 7: Polymers and Geosynthetics (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Classification of polymers: thermoplastics, thermosets, and elastomers • Polymerization reactions: addition and condensation • Mechanical and chemical properties of polymers relevant to engineering • Geosynthetics: geotextiles, geomembranes, and geogrids, and their applications • Degradation and durability of polymers under environmental conditions	• Classify a polymer by its structure and thermal behavior [Usage] • Identify whether a polymerization reaction is addition or condensation [Familiarity] • Select the appropriate geosynthetic type for a given civil engineering application [Usage] • Assess the durability of a polymer under environmental exposure [Assessment]
Readings : [**ChangGoldsby2016**], [**MamloukZaniewski2017**]	

Unit 8: Chemical Equilibrium (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• The equilibrium constant K_c and K_p, and its relation to reaction stoichiometry • The reaction quotient Q and predicting the direction of a reaction • Le Chatelier's principle: effect of concentration, pressure, and temperature • ICE tables for calculating equilibrium concentrations	• Write the equilibrium constant expression for a given reaction [Usage] • Calculate equilibrium concentrations using ICE tables [Usage] • Predict the shift of an equilibrium upon changes in concentration, pressure, or temperature using Le Chatelier's principle [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 9: Ionic Equilibrium and Salt Hydrolysis (12 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Arrhenius, Brønsted-Lowry, and Lewis acid-base theories • Calculating pH and pOH for strong acids and bases • Equilibrium of weak acids and bases: K_a, K_b, and degree of dissociation • Salt hydrolysis and predicting the acidic, basic, or neutral character of a solution • Buffer solutions and the Henderson-Hasselbalch equation	• Calculate the pH of solutions of strong and weak acids and bases [Usage] • Predict the acid-base character of a salt solution from the hydrolysis of its ions [Assessment] • Calculate the pH of a buffer solution using the Henderson-Hasselbalch equation [Usage]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 10: Salts, Solubility, and Precipitation Equilibria (6 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• The solubility product K_{sp} and its relation to molar solubility • Predicting precipitation by comparing the ion product with K_{sp} • Common ion effect on solubility	• Calculate K_{sp} from the molar solubility of a salt, and vice versa [Usage] • Predict whether a precipitate will form when mixing two solutions given their concentrations and the corresponding K_{sp} [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 11: Electrochemistry and Electrolysis (8 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Oxidation-reduction reactions and balancing by the ion-electron method • Galvanic cells, standard electrode potentials, and the Nernst equation • Electrolytic cells and electrolysis; Faraday's laws • Electrochemical corrosion of metals and protection methods • Corrosion of reinforcing steel in reinforced concrete	• Balance redox equations by the ion-electron method in acidic or basic medium [Usage] • Calculate the potential of a galvanic cell and predict the spontaneity of a redox reaction [Usage] • Apply Faraday's laws to calculate the mass deposited or the time in an electrolysis process [Usage] • Analyze the conditions that favor the corrosion of a metal or of reinforcing steel in concrete, and propose protection methods [Assessment]
Readings : [**BrownLeMayBursten2017**], [**ChangGoldsby2016**]	

Unit 12: Cement Chemistry (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Chemical composition of Portland cement: calcium silicates, aluminates, and ferroaluminates • Cement hydration reactions and strength development • Role of each compound (C_3S, C_2S, C_3A, C_4AF) in cement properties • Supplementary cementitious materials: pozzolans, slag, and fly ash	• Describe the chemical composition of Portland cement and its main compounds [Familiarity] • Explain the cement hydration process and its relation to strength development [Usage] • Evaluate the effect of supplementary cementitious materials on concrete properties [Assessment]
Readings : [**MamloukZaniewski2017**], [**ChangGoldsby2016**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BEI101. Technical and professional English I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BEI101. Technical and professional English I
2.2 Semester	:	1st Semester.
2.3 Credits	:	2
2.4 Hours	:	4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

English is the primary language of science, engineering, and technology worldwide. For computing and engineering students, mastering English means gaining direct access to technical documentation, research papers, and professional collaboration without translation barriers. This course builds foundational English skills using contexts drawn from computing systems, hardware, and the everyday work of engineers and scientists.

5. GOALS

- Acquire foundational English grammar structures applied to computing and engineering contexts.
- Identify and use technical vocabulary related to hardware, software, and scientific concepts.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Welcome to Computing (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Verb To Be: Affirmative, Negative, and Questions. • Personal Pronouns and Possessive Adjectives. • Number Expressions (1-100): Ports, addresses, and binary basics. • Social Expressions in academic and lab environments.	• By the end of the unit, students use the verb 'to be' to introduce themselves as computing students and describe basic properties of computing systems (e.g., "The processor is fast. It is a 64-bit CPU.").
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 2: Parts and Systems (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Possessive 's and Plural Nouns applied to components (the CPU's speed, the system's memory). • Hardware components and their relationships. • Adjective + Noun structures for technical descriptions. • Lab interactions: Asking about equipment and tools.	• Students describe computer hardware components and their relationships using possessives and noun phrases (e.g., "the motherboard's slots", "the server's processors").
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 3: How It Works (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Simple I (Third person: He/She/It) to describe processes. • Vocabulary: How programs, algorithms, and devices operate. • Telling the time and interpreting system clocks and schedules. • Verbs of operation: runs, compiles, executes, stores, transmits.	• Students explain how a program or device works using the present simple in the third person (e.g., "The compiler translates source code. It generates an executable file.").
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 4: What Engineers Do (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Simple II (I/You/We/They) for describing professional routines. • Questions and negatives with 'Do/Does' in technical contexts. • Engineering roles: developer, analyst, researcher, systems engineer. • Adverbs of frequency in describing work cycles and testing schedules.	• Students describe the daily tasks and professional roles of computing engineers and scientists using the present simple with accuracy.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 5: Inside a Computer (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Existential: There is / There are applied to architecture descriptions. • Prepositions of place: components on a motherboard, files in a directory. • Vocabulary: memory, storage, buses, slots, ports, layers. • Describing network topology and physical infrastructure.	• Students describe the spatial organization of hardware components and file systems, enabling them to interpret and produce technical architecture descriptions.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 6: Can Computers Think? (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Modal verb Can/Can't to describe computing capabilities and limitations. • Adverbs of manner applied to system performance. • Past of verb 'To Be' (Was/Were): early computing systems and pioneers. • Basic expressions for technical support and troubleshooting.	• Students describe what computing systems can and cannot do, and identify basic facts about the early history of computing using simple past structures.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 7: Computing History (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Past Simple: Regular and Irregular verbs to narrate tech history. • Time expressions for the past: invented, developed, published, launched. • Key figures: Turing, Babbage, Lovelace, von Neumann. • Vocabulary: Generations of computers, milestones, inventions.	• Students narrate key events in the history of computing and science using the simple past correctly, identifying the contributions of foundational figures.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 8: My Professional Goals in Computing (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Introduction to Technical English for Computing and Engineering. • Reading and understanding simple technical specifications. • Working in multicultural engineering teams: communication norms. • Oral presentation: "My role in the computing field".	• Students present their professional goals in computing using basic technical vocabulary and connect them to the roles and responsibilities of software and hardware engineers.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS100. Introduction to Computer Science (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS100. Introduction to Computer Science
2.2 Semester	:	2nd Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course serves as the foundation for understanding the fundamental concepts of computational thinking applicable across various professions.

The course provides, starting from ground zero, a panoramic view of: introductory computational thinking, data storage, computer architecture, operating systems, networks and the Internet, algorithms, sorting methods, software engineering, databases, data structures, software engineering, computer graphics, artificial intelligence among others.

Designed as an introductory course to Computer Science, the concepts are presented in a playful manner and using an Active Learning methodology. Throughout the course, active audience participation is encouraged, akin to a theatrical performance.

The related knowledge areas covered are directly aligned with the Computing Curricula ACM/IEEE-CS.

The course **does not require** any prior knowledge in computer handling topics and can be taken by student from any field.

5. GOALS

- Introduce the fundamental concepts of Computational Thinking and Computer Science to students from any professional background.
- Develop their ability to abstract.
- Understand how Computational Thinking is applied in each of their professions.
- Apply advanced concepts in a simplified manner in any career.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Familiarity)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Familiarity)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Familiarity)

7. TOPICS

Unit 1: Computational Thinking (Part I) (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Explanation of the evaluation system. • General course instructions. • Definition of Computing. • Computing as a Human-Computer binomial. • Distortions in the definition of computing. • Computing as the automation of abstraction. • Computing and Engineering: similarities and differences. • Algorithmic problem-solving. • Dynamics: Understanding the execution of an algorithm at human speed.	• Apply the fundamental concepts of computing in real-life situations. [Usar] • Identify distortions of Computing in real-life situations. [Usar] • Clearly identify at least 3 contexts of using the word"Engineer" in English. [Evaluar] • Identify the limitations of humans in solving computational problems. [Usar]
Readings : [**Brookshear2019**]	

Unit 2: Computational Thinking. Part II (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Decomposition (Breaking a problem in smaller pieces) • Abstraction (Focus on important things) • Pattern recongnition (Identify similar sequences based on previous problems) • Designing algorithms	• Apply Computational Thinking to real-world problems. [Usar]
Readings : [**Brookshear2019**]	

Unit 3: General concepts (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Binary vs. decimal numbering. • Character representation: the ASCII table. • Binary search. • Computational complexity of an algorithm.	• Apply various numbering systems to real-world problems. [Usar] • Understand the internal representation of characters in the ASCII and UTF-8 tables. [Familiarizarse] • Apply Divide and Conquer algorithmic strategy. [Usar] • Determine basic analysis of algorithmic complexity. [Usar]
Readings : [**Brookshear2019**]	

Unit 4: Machine-Level Data Representation (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Overview and history of computer architecture • Bits, bytes, and words • Unsigned, signed and two's complement representations • Numeric data representation and number bases: – Fixed-point – Floating-point • Representation of non-numeric data • Representation of records, arrays and UTF data types	• Discuss why everything in computers are data, including instructions [Debate] • Explain how fixed-length number representations can affect accuracy and precision [Explain] • Describe how negative integers are stored in sign-magnitude and two's-complement representations [Describe] • Discuss how different formats can represent numerical data [Debate] • Explain the bit-level representation of non-numeric data, such as characters, strings, records, and arrays [Explain] • Translate numerical data from one format to another [Translate] • Describe how a single adder (without overflow detection) can handle both signed (two's complement) and unsigned (binary) input without "knowing" which format a given input is using [Describe]
Readings : [**Brookshear2019**]	

Unit 5: Program Abstraction and Representation (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Programs that take (other) programs as input such as interpreters, compilers, type-checkers, documentation generators • Components of a language: – Definitions of alphabets, delimiters, sentences, syntax, and semantics – Syntax vs semantics • Program as a set of non-ambiguous meaningful sentences • Basic programming abstractions: constants, variables, declarations (including nested declarations), command, expression, assignment, selection, definite and indefinite iteration, iterators, function, procedure, modules, exception handling • Types of variables: static, local, nonlocal, global; need and issues with nonlocal and global variables. • Scope rules: static vs dynamic; visibility of variables; side-effects. • Environment vs store and their properties • Data and control abstraction • Mechanisms for information exchange between program units such as procedures, functions, and modules: nonlocal variables, global variables, parameter-passing, import-export between modules • Low level instruction representation such as virtual machine instructions, assembly language, and binary representation	• Illustrate the scope of variables and visibility using simple programs [Apply] • Illustrate different types of parameter-passing using simple pseudo programming language [Apply] • Explain side-effect using global and nonlocal variables and how to fix such programs [Explain] • Explain how programs that process other programs treat the other programs as their input data [Explain] • Describe a grammar and an abstract syntax tree for a small language [Describe]
Readings : [**Brookshear2019**]	

Unit 6: Cryptography Fundamentals (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Basic concepts on Cryptography • Public-keys, Private-keys	• Explain the role of cryptography in supporting security and privacy [Explain] • Discuss the risks of inventing one's own cryptographic methods [Debate] • Discuss the importance of prime numbers in cryptography and explain their use in cryptographic algorithms [Debate] • Implement and cryptanalyze classical ciphers [Implement]
Readings : [**Brookshear2019**]	

Unit 7: Memory Management (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Review of physical memory, address translation and memory management hardware See also: Memory Hierarchy, Discrete Mathematics • Impact of memory hierarchy including cache concept, cache lookup, and per-CPU caching on operating system mechanisms and policy See also: Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Logical and physical addressing, address space virtualization. See also: Memory Hierarchy, Discrete Mathematics • Concepts of paging, page replacement, thrashing and allocation of pages and frames • Allocation/deallocation/storage techniques (algorithms and data structure) performance and flexibility – Arenas, slab allocators, free lists, size classes, heterogeneously sized pages (huge pages) • Memory caching and cache coherence and the effect of flushing the cache to avoid speculative execution vulnerabilities. See also: Functional Organization, Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Virtual memory: leveraging virtual memory hardware for OS services and efficiency	• Explain memory hierarchy and cost-performance tradeoffs [Explain] • Summarize the principles of virtual memory as applied to caching and paging [Summarize] • Evaluate the tradeoffs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed [Evaluate] • Describe the reason for and use of cache memory (performance and proximity, how caches complicate isolation and virtual machine abstraction) [Describe]
Readings : [**Brookshear2019**]	

Unit 8: Role and Purpose of Operating Systems (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Operating systems mediate between general purpose hardware and application-specific software. • Universal operating system functions (e.g., process, user and device interfaces, persistence of data) • Influences of security, networking, multimedia, parallel and distributed computing	• Understand the objectives and functions of modern operating systems [Explain] • Evaluate the design issues in different usage scenarios (e.g., real time OS, mobile, server) [Evaluate] • Understand how evolution and stability are desirable and mutually antagonistic in operating systems function [Explain]
Readings : [**Brookshear2019**]	

Unit 9: Principles of Operating System (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Operating system software design and approaches (e.g., monolithic, layered, modular, micro-kernel, unikernel) • Abstractions, processes, and resources • The evolution of the link between hardware architecture and the operating system functions • Leveraging interrupts from hardware level: service routines and implementations. See also: Assembly Level Machine Organization – Timer interrupts for implementing time slices – I/O interrupts for putting blocking threads to sleep without polling	• Understand how the application of software design approaches to operating systems design/implementation (e.g., layered, modular, etc.) affects the robustness and maintainability of an operating system [Explain] • Categorize system calls by purpose [Categorize] • Apply OS techniques for isolation, protection, and throughput across OS functions (e.g., starvation similarities in process scheduling, disk request scheduling, semaphores, etc.) and beyond [Apply]
Readings : [**Brookshear2019**]	

Unit 10: Fundamentals of Networks and Communications (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Importance of networking in contemporary computing, and associated challenges. See also: Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation • Organization of the internet (e.g., users, Internet Service Providers, autonomous systems, content providers, content delivery networks) • Layers and their roles (application, transport, network, datalink, and physical) • Network elements (e.g., routers, switches, hubs, access points, and hosts) • Basic queueing concepts (e.g., relationship with latency, congestion, service levels, etc.)	• Articulate the organization of the internet [Articulate] • List and define the appropriate network terminology [List] • Describe the layered structure of a typical networked architecture [Describe]
Readings : [**Brookshear2019**]	

Unit 11: Networked Applications (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Naming and address schemes (e.g., DNS, and Uniform Resource Identifiers) • Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination • Diversity of networked application demands (e.g., latency, bandwidth, and loss tolerance). See also: Communication, Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context, Digital Equity and Accessibility, Social Impact and AI • Coverage of application-layer protocols (e.g., HTTP) • Interactions with TCP, UDP, and Socket APIs. See also: Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities	• Define the principles of naming, addressing, resource location [Define] • Analyze the needs of specific networked application demands [Analyze]
Readings : [**Brookshear2019**]	

Unit 12: Complexity Analysis Framework (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Complexity Analysis Framework: – Best, average, and worst-case performance of an algorithm – Empirical and relative (Order of Growth) measurements – Input size and primitive operations – Time and space efficiency • Empirical measurements of performance • Time and space tradeoffs in algorithms	• Prepare a presentation that explains to first year students the basic concepts of algorithmic complexity including best, average, and worst-case algorithm behavior, Big-O, Omega, and Theta notations, complexity classes, time and space tradeoffs, empirical measurement, and impact on practical problems [Explain] • For each algorithm in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain its runtime complexity class and why it belongs to this class [Explain] • Informally evaluate the foundational complexity class of simple algorithms [Evaluate] • Develop empirical studies to determine and validate hypotheses about the runtime complexity of various algorithms by running algorithms on input of various sizes and comparing actual performance to the theoretical analysis [Create] • Explain examples that illustrate time-space tradeoffs of algorithms [Explain] • Explain how tree balance affects the efficiency of binary search tree operations [Explain]
Readings : [**Brookshear2019**]	

Unit 13: Asymptotic Notation and Complexity Classes (1 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Asymptotic complexity analysis (average and worst-case bounds): – Big-O, Big-Omega, and Big-Theta formal notations – Foundational Complexity Classes and Representative Examples/Problems: ∗ $O(1)$ Constant (e.g., array access) ∗ $O(\log_2 n)$ Logarithmic (e.g., binary search) ∗ $O(n)$ Linear (e.g., linear search) ∗ $O(n \log_2 n)$ Log Linear (e.g., mergesort) ∗ $O(n^2)$ Quadratic (e.g., selection sort) ∗ $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) ∗ $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) ∗ $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)	• Using examples, explain each of the foundational complexity classes in this unit [Explain] • For each foundational complexity class in this unit, explain an algorithm that demonstrates the associated runtime complexity [Explain] • Explain to a non-technical audience the significance of tractable versus intractable algorithms using an intuitive explanation of Big-O complexity [Explain]
Readings : [**Brookshear2019**]	

Unit 14: Complexity Analysis: Recursion, Amortization, and Tight Bounds (1 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Little-o, Little-Omega, and Little Theta notations	• Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior [Evaluate]
Readings : [**Brookshear2019**]	

Unit 15: Foundational Data Structures (3 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Abstract Data Type (ADT) and operations on an ADT: – Dictionary operations (insert, delete, find) • Arrays: – Numeric vs non-numeric, character strings – Single (vector) vs multidimensional (matrix) • Records/Structs/Tuples and Objects • Linked lists (for historical reasons): – Single vs Double and Linear vs Circular • Stacks • Queues and deques: – Heap-based priority queue • Hash tables/maps: – Collision resolution and complexity (e.g., probing, chaining, rehash) • Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap)	• For each ADT/Data-Structure in this unit: – Explain its definition, properties, representation(s), and associated ADT operations. – Explain step-by-step how the ADT operations associated with the data structure transform it. [Explain] • Explain how collision avoidance and collision resolution is handled in hash tables [Explain] • Explain the heap property and the use of heaps as an implementation of a priority queue [Explain]
Readings : [**Brookshear2019**]	

Unit 16: Foundational Algorithms (3 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Graphs (e.g., [un]directed, [a]cyclic, [un]connected, and [un]weighted): – Graph representation: adjacency list vs matrix • Search algorithms: – $O(n)$ complexity (e.g., linear/sequential array/list search) – $O(\log_2 n)$ complexity (e.g., binary search) – $O(\log_b n)$ complexity (e.g., uninformed depth/breadth-first tree search) • Sorting algorithms (e.g., stable, unstable): – $O(n^2)$ complexity (e.g., insertion, selection) – $O(n \log n)$ complexity (e.g., quicksort, merge, timsort) • Graph algorithms: – Shortest path (e.g., Dijkstra's, Floyd's) – Minimal spanning tree (e.g., Prim's, Kruskal's) • Sorting algorithms: – $O(n \log n)$ complexity heapsort – Pseudo $O(n)$ complexity (e.g., bucket, counting, radix) • Graph algorithms: – Transitive closure (e.g., Warshall's) – Topological sort	• For each algorithm in this unit explain step-by-step how the algorithm operates [Explain] • For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply] • Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create] • Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain]
Readings : [**Brookshear2019**]	

Unit 17: Advanced Algorithms (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Cryptography algorithms (e.g., SHA-256) • Parallel algorithms	• An appreciation of quantum computation and its application to certain problems [Explain]
Readings : [**Brookshear2019**]	

Unit 18: Core Database System Concepts (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Purpose and advantages of database systems • Components of database systems • Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods) • Transaction management • Normalization • Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce) • Distributed databases/cloud-based systems • Use of a declarative query language	• Identify at least four advantages that using a database system provides [Analyze] • Enumerate the components of a (relational) database system [Enumerate] • Defend the value of data independence [Defend] • Compose a simple select-project-join query in SQL [Compose] • Describe the advantages for eliminating duplicate repeated data [Describe] • Outline how MapReduce uses parallelism to process data efficiently [Outline]
Readings : [**Brookshear2019**]	

Unit 19: Object-Oriented Programming Fundamentals (4 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Imperative programming as a subset of object-oriented programming. • Object-oriented design: – Decomposition into objects carrying state and having behavior. – Class-hierarchy design for modeling. • Definition of classes: fields, methods, and constructors. • Subclasses, inheritance (including multiple inheritance), and method overriding. • Dynamic dispatch: definition of method-call. • Object-oriented idioms for encapsulation: – Privacy, data hiding, and visibility of class members. – Interfaces revealing only method signatures. – Abstract base classes, traits and mixins.	• Enumerate the differences between imperative and object-oriented programming paradigms [Enumerate] • Compose a class through design, implementation, and testing to meet behavioral requirements [Compose] • Use object-oriented encapsulation mechanisms such as interfaces and private members [Use]
Readings : [**Brookshear2019**]	

Unit 20: Version Control and CI/CD (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant. • Release management. • Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Tool integration concepts and mechanisms. Software Development Practices • Use of modern IDE facilities - debugging, refactoring, searching/indexing, ML-powered code assistants, etc. Software Development Practices	• Describe the difference between centralized and distributed software configuration management [Describe] • Describe how version control can be used to help manage software release management [Describe] • Identify configuration items and use a source code control tool in a small team-based project [Analyze] • Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] • Demonstrate the capability to use software tools in support of the development of a software product of medium size [Demonstrate]
Readings : [**Brookshear2019**]	

Unit 21: Requirements Engineering (2 hours)

Competences Expected: AG-C02,AG-C08,AG-C12

Topics	Learning Outcomes
• Describe functional requirements using, for example, use cases or user stories. – Using at least one method of documenting and structuring functional requirements. – Understanding how the method supports design and implementation. – Strengths and weaknesses of using a specific approach. • Requirements elicitation. – Sources of requirements, for example, users, administrators, or support personnel. – Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis. • Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context • Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics • Communicating and/or formalizing requirement specifications. • Prototyping a tool for both eliciting and validating/confirming requirements. • Product evolution: when requirements change, how to understand what effect that has and what changes need to be made. • Effort estimation: – Learning techniques for better estimating the effort required to complete a task; – Practicing estimation and comparing it to how long tasks take; – Effort estimation is quite difficult, so students are likely to be way off in many cases, but seeing the process play out with their own work is valuable.	• Compare different methods of eliciting requirements along multiple axes [Compare] • Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze] • Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze] • Collect a set of requirements for a simple software system [Analyze] • Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze] • Identify the functional and non-functional requirements in a set of requirements [Analyze] • Estimate the time to complete a set of tasks, then compare estimates to the actual time taken [Estimate] • Determine an implementation sequence for a set of tasks, adhering to dependencies between them, with a goal to retire risk as early as possible [Determine] • Write a requirement specification for a simple software system [Write]

Readings : [**Brookshear2019**]

Unit 22: Machine Learning Fundamentals (1 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Definition and examples of a broad variety of machine learning tasks: – Supervised learning: ∗ Classification ∗ Regression – Reinforcement learning – Unsupervised learning: ∗ Clustering • Fundamental ideas: – No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. – Sources of error and undecidability in machine learning • A simple statistical-based supervised learning such as linear regression or decision trees: – Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly	• Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] • Differentiate the terms of AI, machine learning, and deep learning [Evaluate] • Explain how machine learning works as an optimization/search process [Explain]
Readings : [**Brookshear2019**]	

Unit 23: Neural Networks and Ethics (1 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Basic neural networks: – Fundamentals of understanding how neural networks work and their training process, without details of the calculations – Basic introduction to generative neural networks (e.g., large language models) • Ethics for Machine Learning: – Focus on real data, real scenarios, and case studies – Dataset/algorithmic/evaluation bias and unintended consequences • Deep learning: – Deep feed-forward networks (intuition only, no mathematics) – Convolutional neural networks (intuition only, no mathematics) – Visualization of learned feature representations from deep nets – Other architectures (generative NN, recurrent NN, transformers, etc.) • Ethics for Machine Learning: – Continued focus on real data, real scenarios, and case studies – Privacy – Fairness – Intellectual property – Explainability	• Describe the neural network training process and resulting learned representations [Explain] • Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] • Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]
Readings : [**Brookshear2019**]	

Unit 24: Fundamental Concepts of Graphics and Interactive Techniques (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Overview of computer graphics pipeline and interactive systems • Basic rendering concepts: rasterization, ray tracing, shading models • Color theory and color spaces • Coordinate systems and transformations • Basic interaction techniques: input devices, event handling • Advanced rendering concepts: global illumination, physically based rendering • Graphics hardware and acceleration techniques • Human perception and visual cognition	• Explain the basic stages of the computer graphics pipeline [Explain] • Describe basic rendering techniques such as rasterization and ray tracing [Describe] • Apply color theory to design visually effective graphics [Apply] • Implement coordinate transformations for 2D and 3D graphics [Implement] • Design basic interactive applications using common input devices and event handling [Design] • Compare advanced rendering techniques such as global illumination and physically based rendering [Compare] • Analyze the role of graphics hardware in accelerating rendering tasks [Analyze] • Explain how human perception influences graphics and interaction design [Explain]
Readings : [**Brookshear2019**]	

Unit 25: Applied Rendering and Techniques (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Ray tracing: acceleration structures, sampling, shading • Texture mapping and filtering	• Implement rasterization algorithms for primitive rendering [Implement] • Implement basic ray tracing with acceleration structures [Implement]
Readings : [**Brookshear2019**]	

Unit 26: Closure Class: How Does a Search Engine Like Google Work? (2 hours)	
Competences Expected: AG-C02,AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Problem analysis • The index does not grow linearly with the size of the indexed information. • Response time does not depend on the size of the "database." • Response time does not depend on the number of occurrences found. • Combining various data structures to reach a solution. • Analyzing the scalability of the solution.	• Understand the principles under which a search engine is created [Usar] • Correctly apply data structures to solve the problem [Usar] • Apply concepts related to algorithmic complexity in a search engine [Usar].
Readings : [**Brookshear2019**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS112. Objects-oriented programming I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS112. Objects-oriented programming I
2.2 Semester	:	2nd Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BIC101. Introduction to Computing. (1st Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the second course in the introductory Computer Science programming sequence. It incorporates a programming language transition as a pedagogical strategy and focuses on core Object-Oriented Programming (OOP) concepts in C++17/C++20, expanding on the foundations from the first course while introducing low-level programming concepts. The course provides a solid foundation for advanced topics covered in the subsequent course of the curriculum.

5. GOALS

- Introduce the student to the foundations of the object orientation paradigm in C++
- Develop skills in C++17/C++20 programming including templates and STL
- Understand memory management and low-level programming concepts
- Develop problem-solving skills using OOP principles
- Master intermediate C++ programming techniques

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Language Pragmatics (2 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Problem domains and programming paradigm • Criteria for good programming language design: – Principles of language design such as orthogonality – Defining control and iteration constructs – Modularization of large software • Brief review of programming paradigms • Comparison between functional and imperative programming • History of programming languages (emphasis on C and C++)	• Discuss the historical context of programming language paradigms [Familiarizarse]. • Evaluate language design trade-offs for specific applications [Evaluar].
Readings : [**Stroustrup2013**], [**Josuttis2019**]	

Unit 2: Language Translation and Execution (4 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Execution models for JIT (Just-In-Time), compiler, interpreter • Use of intermediate code, e.g., bytecode • Execution as native code or within a virtual machine • Virtual machines and intermediate languages • Compilation vs. interpretation models	• Explain and understand the differences between compiled, JIT, and interpreted language implementations, including the benefits and limitations of each [Familiarizarse] • Differentiate syntax and parsing from semantics and evaluation [Usar] • Illustrate ambiguity in parsing using nested if-then-else/arithmetic expression and show resolution using precedence order [Evaluar]
Readings : [**Stroustrup2013**], [**Deitel2017**]	

Unit 3: Type Systems Fundamentals (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• A type as a set of values together with a set of operations: – Primitive types (e.g., numbers, Booleans) – Compound types built from other types (e.g., records/structs, unions, arrays, lists, functions, references using set operations) • Generic types (parametric polymorphism): – Definition and advantages of polymorphism: parametric, subtyping, overloading, and coercion – Comparison of monomorphic and polymorphic types – Comparison with ad-hoc polymorphism (overloading) and subtype polymorphism – Generic parameters and typing – Use of generic libraries such as collections – Comparison with ad hoc polymorphism (overloading) and subtype polymorphism – Prescriptive vs descriptive polymorphism – Implementation models of polymorphic types – Subtyping • Declaration models (linking, visibility, scope, lifetime) • Overview of type checking in C++ • Type conversion and casting in C++	• Define and use program pieces (such as functions, classes, methods) that use generic types, including for collections [Usar] • Discuss the differences among generics, subtyping, and overloading [Familiarizarse] • Explain multiple benefits and limitations of static typing in writing, maintaining, and debugging software [Evaluar]
Readings : [**Stroustrup2013**], [**Josuttis2019**]	

Unit 4: Object-Oriented Programming Fundamentals (14 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Definition of classes: fields, methods, and constructors. • Subclasses, inheritance (including multiple inheritance), and method overriding. • Dynamic dispatch: definition of method-call. • Subtyping: – Subtype polymorphism; implicit upcasts in typed languages. – Notion of behavioral replacement: subtypes acting like supertype. – Relationship between subtyping and inheritance. • Classes and objects in C++ • Constructors and destructors • Access specifiers and encapsulation • Inheritance hierarchies • Polymorphism and virtual functions	• Compose a class through design, implementation, and testing to meet behavioral requirements [Usar] • Build a simple class hierarchy utilizing subclassing that allows code to be reused for distinct subclasses [Usar] • Predict and validate control flow in a program using dynamic dispatch [Evaluar] • Design C++ classes with proper encapsulation [Usar]. • Implement inheritance hierarchies for code reuse [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**]	

Unit 5: Systems Execution and Memory Model (12 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Direct, indirect, and indexed access to memory location • Run-time representation of data abstractions such as variables, arrays, vectors, records, pointer-based data elements such as linked-lists and trees, and objects • Low level allocation and accessing of high-level data structures such as basic data types, n-dimensional array, vector, record, and objects • Return from procedure as automatic deallocation mechanism for local data elements in the stack • Manual memory management: allocating, deallocating, and reusing heap memory • Automated memory management: garbage collection as an automated technique using the notion of reachability • Memory layout (stack, heap, static) • Pointer arithmetic and operations • Dynamic memory allocation (new/delete) • Memory leaks and dangling pointers • Smart pointers (unique_ptr, shared_ptr)	• Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Usar] • Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Evaluar] • Explain how a core language construct, such as data abstractions and control abstractions, is executed [Evaluar] • Implement RAII for resource management [Usar]. • Debug memory-related errors in C++ programs [Usar].
Readings : [**Stroustrup2013**], [**Deitel2017**]	

Unit 6: Functional Programming (10 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Lambda expressions and evaluation: – Variable binding and scope rules. – Parameter-passing. – Nested lambda expressions and reduction order. • Using higher-order functions (taking, returning, and storing functions). • Function templates and class templates • Template specialization • Lambda expressions in C++11/14 • Standard Template Library concepts	• Develop useful functions that take and return other functions [Usar] • Correctly interpret variables and lexical scope in a program using function closures [Evaluar] • Use functional encapsulation mechanisms such as closures and modular interfaces [Usar] • Implement generic functions using templates [Usar]. • Use lambda expressions for functional programming [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**]	

Unit 7: Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Collection classes, iterators, and other common library components. • Sequence containers (vector, list, deque) • Associative containers (set, map, unordered_set) • Container adapters (stack, queue, priority_queue) • Iterators and algorithms • Functors and predicates	• Use collection classes and iterators effectively to solve a problem [Usar] • Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language [Usar] • Compare and contrast mechanisms to define and protect data elements within procedural, functional, and object-oriented approaches [Evaluar] • Select appropriate STL containers for specific needs [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**]	

Unit 8: Advanced Programming Constructs (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods • Metaprogramming: macros, generative programming, model-based development • Operator overloading • Move semantics and rvalue references • Exception handling in C++ • Namespaces and modular programming	• Use various advanced programming constructs and idioms correctly [Usar] • Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Evaluar] • Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Familiarizarse] • Implement move semantics for performance optimization [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**]	

Unit 9: Shell Scripting (4 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• System commands: – Interface with operating systems • Piping • File abstraction and operators • Programs and processes • Automating compilation and testing of C++ programs through scripts (Makefiles, build/test scripts)	• Create and execute automated scripts to manage various system tasks [Usar]
Readings : [**Robbins2005**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS1D1. Discrete Structures (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS1D1. Discrete Structures
2.2 Semester	:	2nd Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Discrete structures provide the theoretical foundation for computer science. This course introduces the essential mathematical tools used to model computational problems, including logic, set theory, and functions. Understanding these concepts is critical for developing rigorous reasoning skills and for the subsequent study of algorithms, databases, and formal languages.

5. GOALS

- Understand the principles of mathematical logic and proof techniques.
- Perform operations with sets, functions, and relations.
- Apply basic counting techniques and recurrence relations to problems.
- Develop the ability to think abstractly and solve problems formally.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Discrete Mathematics (12 hours)	
Competences Expected: AG-C07,AG-C12	
Topics	**Learning Outcomes**
• Logic: propositional and predicate logic, quantifiers, proofs • Sets: operations, cardinality, power sets, Cartesian products • Relations: properties, equivalence relations, partial orders • Functions: injective, surjective, bijective, composition, inverse • Advanced logic: proof systems, completeness, decidability • Advanced set theory: infinite sets, Cantor's diagonal argument	• Explain propositional and predicate logic [Explain] • Construct simple logical proofs [Apply] • Define basic set operations and relations [Explain] • Define properties of relations [Explain] • Define types of functions [Explain]
Readings : [**Rosen2019**], [**Lehman2018**]	

Unit 2: Discrete Mathematics (12 hours)	
Competences Expected: AG-C07,AG-C12	
Topics	**Learning Outcomes**
• Logic: propositional and predicate logic, quantifiers, proofs • Advanced logic: proof systems, completeness, decidability • Advanced set theory: infinite sets, Cantor's diagonal argument • Advanced relations: closures, lattices • Advanced functions: recursive functions, computability	• Construct simple logical proofs [Apply] • Explain concepts of infinite sets and cardinality [Explain] • Apply closure operations to relations [Apply] • Determine if a relation is an equivalence relation [Apply]
Readings : [**Rosen2019**], [**Velleman2019**]	

Unit 3: Discrete Mathematics (12 hours)	
Competences Expected: AG-C07,AG-C12	
Topics	**Learning Outcomes**
• Relations: properties, equivalence relations, partial orders • Functions: injective, surjective, bijective, composition, inverse • Graphs: definitions, paths, cycles, trees, connectivity • Advanced graph theory: planarity, coloring, networks	• Determine if a relation is an equivalence relation [Apply] • Compose functions and find inverses [Apply] • Define basic graph concepts [Explain] • Solve basic graph problems [Apply] • Analyze advanced graph properties [Evaluate]
Readings : [**Rosen2019**], [**Scheinerman2012**]	

Unit 4: Discrete Mathematics (12 hours)	
Competences Expected: AG-C07,AG-C12	
Topics	**Learning Outcomes**
• Counting: permutations, combinations, binomial theorem • Advanced counting: generating functions, inclusion-exclusion • Recurrence relations: solving simple recurrences • Advanced recurrence: generating functions, asymptotic behavior	• Explain permutations and combinations [Explain] • Apply counting techniques to solve problems [Apply] • Define recurrence relations [Explain] • Solve simple recurrence relations [Apply] • Develop generating functions for counting problems [Create]
Readings : [**Rosen2019**], [**Grimaldi2003**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BMA103. Integral Calculus (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BMA103. Integral Calculus
2.2 Semester	:	2nd Semester.
2.3 Credits	:	5
2.4 Hours	:	4 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BMA102. Differential Calculus. (1st Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Integral calculus is a fundamental mathematical tool shared by computer science and engineering disciplines, essential for modeling and solving problems involving accumulation, change, and areas under curves. This course provides the foundations of integral calculus, including integration techniques and their applications in scientific and engineering contexts.

5. GOALS

- Understand the concept of definite and indefinite integrals.
- Apply various integration techniques to solve problems.
- Use integral calculus to model and solve problems in scientific and engineering contexts.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

7. TOPICS

Unit 1: Antiderivatives and Integration Techniques (20 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• The Fundamental Theorem of Calculus and the antiderivative • Integration techniques: substitution, parts, partial fractions, and trigonometric substitution	• State the Fundamental Theorem of Calculus and explain the relationship between differentiation and integration [Familiarity] • Compute definite and indefinite integrals using standard techniques including substitution and integration by parts [Usage]
Readings : [**Stewart2015**], [**LarsonEdwards2014**]	

Unit 2: Improper Integrals (8 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Improper integrals: convergence criteria and comparison tests	• Determine convergence of improper integrals using comparison and other convergence tests [Assessment]
Readings : [**Stewart2015**], [**LarsonEdwards2014**]	

Unit 3: Sequences, Series, and Power Series (20 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Sequences and series: convergence tests (ratio, root, comparison, integral) • Power series, radius of convergence, Taylor and Maclaurin series	• Determine convergence of power series and construct Taylor series for standard functions [Assessment]
Readings : [**Stewart2015**], [**LarsonEdwards2014**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BFI101. Physics I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BFI101. Physics I
2.2 Semester	:	2nd Semester.
2.3 Credits	:	5
2.4 Hours	:	4 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Physics I introduces the foundations of classical mechanics as a mathematical and experimental discipline. Through the study of kinematics, Newtonian dynamics, conservation principles, rotational motion, elasticity, fluid mechanics, vibrations and waves, and an introduction to transport phenomena, students acquire the analytical tools needed to model and solve problems of the physical world. This course provides the indispensable foundation for the study of oscillatory, wave, and electromagnetic phenomena addressed in subsequent courses.

5. GOALS

- Describe the motion of particles in different coordinate systems and formulate the equations of motion from Newton's laws.
- Apply the principles of work, potential energy, and conservation of mechanical energy to systems with conservative and non-conservative forces.
- Solve particle-system dynamics problems using conservation of linear momentum and collision analysis.
- Analyze rotational motion and angular momentum dynamics in systems of particles and extended bodies.
- Apply elastic moduli and Hooke's law to relate stress and strain in solids.
- Apply the principles of hydrostatics and hydrodynamics, including the continuity and Bernoulli equations, to fluid flow problems.
- Analyze simple harmonic motion and mechanical wave propagation, including superposition and resonance phenomena.
- Apply Fourier's and Fick's laws to describe basic heat conduction and mass diffusion processes.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

7. TOPICS

Unit 1: Kinematics and Dynamics of Particles (22 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Vector description of position, velocity, and acceleration; Newton's three laws and inertial reference frames • Formulation and analytical solution of equations of motion for constant and variable forces • Phase space diagrams (position vs. velocity) for visualizing mechanical state, stability, and periodicity • Numerical integration of equations of motion using Euler, Verlet, and Runge-Kutta methods • Constrained motion and holonomic constraints relevant to robotic linkages and articulated bodies	• Identify all forces acting on a particle in a given mechanical scenario and draw a correct free-body diagram [Familiarity] • Implement a numerical integration algorithm such as the Runge-Kutta method to compute the trajectory of a particle under variable forces [Usage] • Construct and interpret phase space diagrams to characterize the stability and periodicity of a simulated mechanical system [Assessment]
Readings : [**Serway2018**], [**Young2015**]	

Unit 2: Work, Energy, and Conservation Laws (12 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Work-energy theorem relating net work done to change in kinetic energy • Potential energy functions and the force-gradient relation for conservative force fields • Conservation of total mechanical energy and conditions for its applicability • Energy diagrams for identifying equilibrium points, turning points, and stability • Elastic and inelastic collision models: momentum and energy exchange in physics engine implementations	• Explain the distinction between conservative and non-conservative forces and the conditions under which mechanical energy is conserved [Familiarity] • Apply energy conservation methods to compute speeds and positions in one-dimensional motion problems common in physics engine implementations [Usage] • Analyze elastic and inelastic collision events, computing post-collision velocities and quantifying energy dissipation [Assessment]
Readings : [**Serway2018**], [**Tipler2010**]	

Unit 3: Systems of Particles and Linear Momentum (8 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Definition of center of mass: $\vec{R}_{CM} = \frac{1}{M}\sum m_i \vec{r}_i$ • Linear momentum of a particle and system: $\vec{p} = m\vec{v}$ and $\vec{P} = M\vec{V}_{CM}$ • Newton's second law for systems: $\vec{F}_{ext} = \frac{d\vec{P}}{dt}$ • Conservation of linear momentum for isolated systems • Distinction between internal and external forces in a system • Two-body collisions including elastic, inelastic, and completely inelastic cases • Coefficient of restitution and its physical meaning in collisions • Systems with variable mass such as the rocket equation: $M\frac{dV}{dt} = v_{ex}\frac{dM}{dt}$ • Many-particle systems and their center of mass dynamics • Continuous mass distributions and their center of mass calculation using integration	• Locate the center of mass for a discrete set of particles in One, Two, and Three dimensions [Usage] • Determine the total linear momentum of a system from the motion of its center of mass [Usage] • Apply conservation of linear momentum to solve collision problems in One and Two dimensions [Usage] • Classify collisions as elastic, inelastic, or completely inelastic based on kinetic energy conservation [Familiarity] • Calculate the coefficient of restitution from pre-collision and post-collision velocities [Usage] • Solve the rocket equation to determine velocity as a function of ejected mass [Usage] • Analyze how internal forces affect individual particles but not the total momentum of a system [Assessment] • Integrate to find the center of mass of symmetric continuous bodies such as rods, disks, and spheres [Usage]
Readings : [**Young2015**], [**Tipler2010**]	

Unit 4: Rotational Motion and Angular Momentum (12 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Rotational kinematics including angular displacement, velocity, and acceleration (θ, ω, α) • Torque definition: $\vec{\tau} = \vec{r} \times \vec{F}$ and its magnitude calculation • Angular momentum of a particle: $\vec{L} = \vec{r} \times \vec{p}$ • Rotational form of Newton's second law: $\vec{\tau}_{net} = \frac{d\vec{L}}{dt}$ • Moment of inertia $I = \sum m_i r_i^2$ and the parallel axis theorem • Rotational kinetic energy: $K_{rot} = \frac{1}{2} I\omega^2$ • Conservation of angular momentum for isolated systems • Gyroscopic motion and the phenomenon of precession • Rolling without slipping dynamics: $v_{CM} = R\omega$ • Coriolis effect as a manifestation of rotational dynamics in non-inertial frames	• Relate linear and angular kinematic quantities for circular motion [Usage] • Calculate torque about a given axis for a force applied at a specific point [Usage] • Determine the moment of inertia for simple geometries such as a rod, hoop, disk, or sphere [Usage] • Apply conservation of angular momentum to analyze changes in rotational speed when the moment of inertia changes [Assessment] • Compute rotational kinetic energy for a rotating rigid body [Usage] • Analyze the motion of objects rolling without slipping down an incline [Assessment] • Predict the direction of precession for a spinning top or gyroscope [Assessment] • Explain the Coriolis effect in terms of rotating reference frames and its effect on projectiles [Familiarity]
Readings : [**Serway2018**], [**Kleppner2013**]	

Unit 5: Elasticity (2 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Definition of stress (force/area) and strain $(\Delta L/L)$ • Young's modulus and Hooke's law for longitudinal strain: $\sigma = E\epsilon$ • Shear modulus (rigidity) and shear strain • Bulk modulus (compressibility) and volumetric strain • Elastic limit, plastic region, and fracture point on the stress-strain diagram • Elastic energy stored in a deformed material	• Apply the scalar Hooke's law to calculate the deformation of a rod under axial load [Usage] • Calculate Young's, shear, or bulk modulus from experimental stress-strain data [Usage] • Identify the elastic region, elastic limit, and fracture point on a stress-strain curve [Familiarity] • Calculate the elastic potential energy stored in a deformed spring or rod [Usage]
Readings : [**Serway2018**], [**Young2015**]	

Unit 6: Fluids: Hydrostatics, Hydrodynamics, and Viscosity (10 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Density, pressure, and its variation with depth in a fluid at rest: $p = p_0 + \rho gh$ • Pascal's principle and its applications (hydraulic press) • Archimedes' principle, buoyant force, and flotation • Pressure measurement with manometers and barometers • Continuity equation for incompressible flow: $A_1 v_1 = A_2 v_2$ • Bernoulli's equation and energy conservation in ideal flow • Applications: Venturi tube, Torricelli's theorem, Pitot tube • Viscosity and Poiseuille's law for laminar flow in pipes • Reynolds number and the transition to turbulent flow • Stokes' law for the drag force on a sphere in a viscous fluid	• Calculate the pressure at a point in a fluid at rest given depth and density [Usage] • Apply Archimedes' principle to determine whether an object floats, sinks, or remains in equilibrium [Usage] • Apply the continuity and Bernoulli equations to solve flow problems in pipes of varying cross-section [Usage] • Interpret manometer and barometer readings to determine absolute and gauge pressures [Assessment] • Apply Poiseuille's law to calculate the flow rate of a viscous fluid in a cylindrical tube [Usage] • Estimate the Reynolds number of a flow and predict whether it is laminar or turbulent [Assessment]
Readings : [**Serway2018**], [**Young2015**]	

Unit 7: Vibrations and Waves (14 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Simple harmonic motion (SHM): equation of motion, position, velocity, and acceleration • Oscillating systems: spring-mass and simple/physical pendulum • Energy in simple harmonic motion • Damped and forced oscillations; introduction to resonance • Transverse and longitudinal waves; propagation speed • Wave equation and speed of a wave on a taut string: $v = \sqrt{T/\mu}$ • Superposition principle and standing waves • Sound waves, intensity, and sound intensity level (decibels) • Doppler effect for sound waves • Beats and interference of waves of nearby frequencies	• Derive the equation of motion and the period of a spring-mass system and a simple pendulum [Usage] • Calculate the kinetic, potential, and total energy of a simple harmonic oscillator at any instant [Usage] • Analyze the amplitude response of a forced oscillator near the resonance frequency [Assessment] • Calculate the propagation speed of a wave on a string from the tension and linear density [Usage] • Determine the frequencies and wavelengths of the normal modes of a standing wave on a fixed string [Usage] • Calculate the sound intensity level in decibels from the intensity of a sound wave [Usage] • Apply the Doppler effect formula for a moving source and/or observer [Usage]
Readings : [**Serway2018**], [**Young2015**]	

Unit 8: Transport Phenomena (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Heat conduction and Fourier's law: $\vec{q} = -k\nabla T$ • Thermal conductivity of materials and thermal resistance in composite walls • Molecular diffusion and Fick's first law: $\vec{J} = -D\nabla C$ • Analogy between heat, mass, and momentum transport (Prandtl and Schmidt numbers)	• Apply Fourier's law to calculate the heat flux through a plane or composite wall [Usage] • Apply Fick's first law to calculate the diffusive flux of a species in a concentration gradient [Usage] • Identify mathematical analogies between heat, mass, and momentum transport processes [Familiarity]
Readings : [**Serway2018**], [**Tipler2010**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BEI102. Technical and professional English II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BEI102. Technical and professional English II
2.2 Semester	:	2nd Semester.
2.3 Credits	:	2
2.4 Hours	:	4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BEI101. Technical and professional English I. (1st Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Building on foundational English skills, this course develops the student's ability to communicate in technical and scientific contexts with greater fluency. Grammar structures are practiced through reading and producing descriptions of technological systems, scientific phenomena, and engineering processes. Students gain vocabulary relevant to computing, data, and the physical sciences.

5. GOALS

- Develop fluency in English applied to technical and scientific communication.
- Expand vocabulary in computing, data, and engineering domains.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Breakthroughs and Discoveries (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Simple Past Review: Affirmative and Negative in scientific contexts. • Time expressions with 'ago': When technologies were invented or released. • Conjunctions: and, so, but, because — structuring technical explanations. • Vocabulary: Scientific milestones, inventions, and discoveries.	• By the end of the unit, students narrate past technological breakthroughs and connect cause-and-effect relationships using simple past and conjunctions (e.g., "Scientists discovered X, so engineers developed Y").
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 2: Data and Variables (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Countable and Uncountable nouns in computing: files, data, code, memory. • Use of some, any, a lot of in quantifying technical resources. • Expressions of quantity: a little memory, a few errors, too many processes. • Vocabulary: Data types, variables, constants, and storage units.	• Students correctly use countable and uncountable noun structures to describe data, resources, and system states in technical contexts.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 3: Future of Technology (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Future plans: Verb + to + infinitive in project planning. • Use of 'Will' vs 'Going to' for predictions and scheduled releases. • Auxiliary verbs in forecasting: AI trends, cloud computing, quantum computing. • Vocabulary: Emerging technologies and career pathways in computing.	• Students express technological intentions and project plans, as well as make evidence-based predictions about computing and engineering trends.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 4: Comparing Technologies (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Comparative and superlative adjectives applied to hardware and software. • Comparing programming languages, operating systems, and algorithms. • Synonyms and antonyms in technical vocabulary. • Reading: Short technology comparison articles and benchmark reports.	• Students make detailed technical comparisons between systems, languages, and tools, using comparative and superlative structures accurately.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 5: Running Processes (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Continuous to describe active and ongoing technical processes. • Current use: "The server is processing requests." • Future use with Present Continuous: scheduled deployments and experiments. • Writing: Status updates and progress reports in technical projects.	• Students describe active computing processes, ongoing experiments, and scheduled technical activities using the present continuous form.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 6: Software Milestones (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Introduction to Present Perfect to describe achievements with current relevance. • Keywords: ever, never, yet, just — applied to software releases and research. • Vocabulary: Version history, patches, upgrades, published papers. • Reading: Technology news and software changelogs.	• Students communicate technical achievements and software milestones with present relevance, without specifying an exact time, using the present perfect.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 7: Technical Challenges (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Adjectives ending in -ed and -ing to describe technical experiences and processes. • Modal verbs of obligation: must, have to — applied to engineering requirements. • Vocabulary: Engineering challenges, constraints, trade-offs, and standards. • Writing: Short technical problem statements.	• Students describe the emotional and technical aspects of engineering challenges and express obligations and constraints using specialized vocabulary.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

Unit 8: English for Technology II (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Technical terminology: Input/output devices, storage, and peripherals. • Reading and following software installation and configuration instructions. • Communication in technical support: Describing symptoms and solutions. • Presentation: Describing a computing system to a non-specialist audience.	• By the end of the course, students interpret technical instructions, describe hardware components, and communicate basic computing concepts to diverse audiences in English.
Readings : [**Soars021S**], [**Cambridge06**], [**MacGrew99**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS113. Objects-oriented programming II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS113. Objects-oriented programming II
2.2 Semester	:	3rd Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS112. Objects-oriented programming I. (2nd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the third course in the sequence of introductory courses in computer science. This course explores advanced C++17/C++20 features focusing on high-performance systems development. Key topics include template metaprogramming, move semantics, perfect forwarding, RAII optimization, multiple/virtual inheritance patterns, concurrency with std::thread and async/await, lock-free programming, modern C++ paradigms such as CRTP and expression templates, and interfacing C++ with other languages. The course prepares students for game engines, high-performance computing, and embedded development where C++ dominates.

5. GOALS

- Master advanced C++17/C++20 features including templates and metaprogramming
- Develop skills in concurrent and systems programming
- Understand performance optimization techniques
- Prepare for high-performance computing applications
- Master template metaprogramming and compile-time computation

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Assessment)

7. TOPICS

Unit 1: Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Collection classes, iterators, and other common library components. • Associative containers (std::set, std::map, unordered variants) • Container adapters (stack, queue, priority_queue) • Advanced STL algorithms • Functors and predicates • Iterator categories and traits	• Use collection classes and iterators effectively to solve a problem [Usar] • Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language [Evaluar] • Compare and contrast the benefits and costs/impact of using inheritance (subclasses) and composition (specifically, how to base composition on higher order functions) [Usar] • Implement custom allocators for STL containers [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**], [**Vandevoorde2017**]	

Unit 2: Type Systems: Polymorphism and Complementary Typing (12 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Generic types (parametric polymorphism): – Definition and advantages of polymorphism: parametric, subtyping, overloading, and coercion – Comparison of monomorphic and polymorphic types – Comparison with ad-hoc polymorphism (overloading) and subtype polymorphism – Generic parameters and typing – Use of generic libraries such as collections – Comparison with ad hoc polymorphism (overloading) and subtype polymorphism – Prescriptive vs descriptive polymorphism – Implementation models of polymorphic types – Subtyping • SFINAE (Substitution Failure Is Not An Error): historical foundation of constrained generic programming in C++, the basis of pre-C++20 libraries. • Concepts (C++20): declarative syntax to constrain template parameters, the modern successor to SFINAE with better error diagnostics. • if constexpr (C++17): compile-time branching, replacing much of the tag dispatch previously handled with SFINAE. • Perfect forwarding and universal references • Variadic templates • Compile-time computation with constexpr and consteval	• Define and use program pieces (such as functions, classes, methods) that use generic types, including for collections [Evaluar] • Discuss the differences among generics, subtyping, and overloading [Evaluar] • Apply Concepts to constrain template parameters and improve error diagnostics [Usar]. • Distinguish when to use Concepts, if constexpr, or legacy SFINAE when maintaining existing C++ code [Evaluar]. • Implement perfect forwarding for efficient argument passing [Usar].
Readings : [**Stroustrup2013**], [**Vandevoorde2017**], [**StroustrupTour2022**]	

Unit 3: Memory Management (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Low level allocation and accessing of high-level data structures such as basic data types, n-dimensional array, vector, record, and objects • Return from procedure as automatic deallocation mechanism for local data elements in the stack • Manual memory management: allocating, deallocating, and reusing heap memory • Automated memory management: garbage collection as an automated technique using the notion of reachability • Lvalues and rvalues in C++11/14/17/20 • Move constructors and move assignment operators • Rvalue references and reference collapsing • Resource Acquisition Is Initialization (RAII) pattern • Smart pointer customization • Memory leak detection and prevention	• Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Usar] • Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Evaluar] • Explain how a core language construct, such as data abstractions and control abstractions, is executed [Evaluar] • Implement move semantics for resource optimization [Usar]. • Design RAII classes for exception-safe resource management [Usar].
Readings : [**Meyers2014**], [**Stroustrup2013**]	

Unit 4: Parallel and Distributed Computing (12 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Safety and liveness: – Race conditions – Dependencies/preconditions – Fault models – Termination • Programming models: – Actor models – Procedural and reactive models – Synchronous/asynchronous programming models – Data parallelism • Communication and coordination: – Mutexes – Message-passing – Shared memory – Cobegin-coend – Monitors – Channels – Threads – Guards • std::thread and thread management • std::jthread (C++20): automatic joining (RAII) and cooperative cancellation via stop_token • Mutexes, locks, and condition variables • C++20 synchronization primitives: std::latch, std::barrier, std::counting_semaphore • Atomic operations and lock-free programming • std::async and futures • std::execution and the senders-receivers model (C++26, experimental): a preview of where standard C++ concurrency is heading, presented at a conceptual level only, not evaluated in lab given its limited availability (partial in GCC) • Thread-safe design patterns • Data races and deadlock prevention	• Implement correct concurrent programs using multiple programming models, such as shared memory, actors, futures, synchronization constructs, and data-parallelism primitives [Usar] • Use synchronization constructions such as monitor/synchronized methods in a simple program [Usar] • Use a message-passing model to analyze a communication protocol [Evaluar] • Implement lock-free data structures using atomics [Usar]. • Differentiate std::thread from std::jthread and justify when to use each [Usar]. • Analyze thread-safety of concurrent algorithms [Evaluar].
Readings : [**Williams2019**], [**Stroustrup2013**]	

Unit 5: Advanced Programming Constructs (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods • Metaprogramming: macros, generative programming, model-based development • Multiple inheritance and virtual inheritance • CRTP (Curiously Recurring Template Pattern) • deducing this (C++23): simplifies CRTP by removing the need for static_cast<Derived&>(*this) for static polymorphism • Expression templates • Type traits and compile-time introspection • Policy-based design • Interface design for libraries	• Use various advanced programming constructs and idioms correctly [Usar] • Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Evaluar] • Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Evaluar] • Implement CRTP for static polymorphism [Usar]. • Design extensible library interfaces using policy-based design [Usar].
Readings : [**Stroustrup2013**], [**Gregor2017**]	

Unit 6: Systems Execution and Memory Model (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Run-time representation of data abstractions such as variables, arrays, vectors, records, pointer-based data elements such as linked-lists and trees, and objects • Data structures for translation, execution, translation, and code mobility such as stack, heap, aliasing (sharing using pointers), indexed sequence and string • Implementation of the dynamic dispatch introduced in OOP I: virtual function table (vtable), vtable pointer (vptr), and its memory/performance cost versus static dispatch • Interfacing C++ with other languages (FFI) • System calls and OS interaction • Memory-mapped I/O • Performance profiling and benchmarking • Optimization techniques for HPC	• Explain how a core language construct, such as data abstractions and control abstractions, is executed [Usar] • Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Evaluar] • Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Usar] • Profile and optimize C++ programs for performance [Usar]. • Explain how the compiler implements runtime polymorphism via vtables [Evaluar]. • Interface C++ code with C libraries [Usar].
Readings : [**Stroustrup2013**], [**Williams2019**]	

Unit 7: Advanced Programming Constructs (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods • Creational, structural, and behavioral patterns • Thread-safe singleton patterns • Modern C++ design idioms • Dependency injection and inversion of control	• Use various advanced programming constructs and idioms correctly [Usar] • Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Evaluar] • Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Familiarizarse] • Implement thread-safe design patterns [Usar]. • Evaluate architectural trade-offs in C++ systems [Evaluar].
Readings : [**Gamma1994**], [**Stroustrup2013**]	

Unit 8: Advanced Programming Constructs (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Control abstractions: exception handling, continuations, monads. • Compiler optimizations and flags • Benchmarking methodologies • Cache-friendly data structures • Branch prediction and alignment • Reducing compilation time • Continuous integration for performance testing	• Use various advanced programming constructs and idioms correctly [Usar] • Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Evaluar] • Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Familiarizarse] • Apply optimization techniques to improve program performance [Usar]. • Interpret benchmark results to guide optimization decisions [Usar].
Readings : [**Meyers2014**], [**Stroustrup2013**]	

Unit 9: Version Control and CI/CD (4 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Build systems for C++: CMake, the de facto industry standard • Modern alternatives: xmake, Meson, and Bazel – comparing declarative approaches versus Lua/Python-based ones • Organizing multi-file projects: translation units, headers, static and dynamic libraries • Dependency management in C++ (vcpkg, Conan)	• Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Familiarizarse] • Configure a multi-file C++ project using a declarative build system [Usar]. • Compare CMake with at least one modern alternative (xmake, Meson, or Bazel) in terms of build speed and reproducibility [Evaluar].
Readings : [**Stroustrup2013**], [**StroustrupTour2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS2B1. Platform Based Development (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS2B1. Platform Based Development
2.2 Semester	:	3rd Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS112. Objects-oriented programming I. (2nd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Platform-based development focuses on the design and implementation of applications within specific computing environments. This course utilizes the Software Process and Design (SPD) framework to explore the unique constraints and capabilities of Web, Mobile, and Interactive platforms. Students will learn to apply shared architectural concerns and platform-specific APIs to build robust, scalable, and user-centric software solutions.

5. GOALS

- Understand the shared aspects and unique constraints of modern platforms.
- Develop professional applications for Web and Mobile environments.
- Apply interactive computing principles to enhance user experience.
- Design software architectures that leverage platform-specific services.

6. COMPETENCES

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Common Aspects: Platforms, APIs and Constraints (5 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Overview of development platforms (i.e., web, mobile, game, robotics, embedded, and interactive). – Input/sensors/control devices/haptic devices – Resource constraints ∗ Computational ∗ Data storage ∗ Memory ∗ Communication – Requirements - security, uptime availability, fault tolerance Reliability Concepts, Reliability Engineering, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering – Output/actuators/haptic devices • Programming via platform-specific Application Programming Interface (API) vs traditional application construction • Programming under platform constraints and requirements (e.g., available development tools, development, security considerations) AI and Hardware Security, Access Control and Applications, Security Mindset and Principles	• List the constraints of mobile programming [List]
Readings : [**Fielding:2000:Rest**], [**Gasca2021**]	

Unit 2: Common Aspects: Platform Languages and Web Programming Patterns (3 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Overview of platform Languages (e.g., Python, Swift, Lua, Kotlin) • Techniques for learning and mastering a platform-specific programming language	• List the characteristics of scripting languages [List] • Describe the three-tier model of web programming [Describe] • Describe how the state is maintained in web programming [Describe]
Readings : [**Fielding:2000:Rest**], [**Gasca2021**]	

Unit 3: Frontend, Frameworks and Web Standards (12 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Web programming languages (e.g., HTML5, JavaScript, Typescript, CSS) • Web platforms, frameworks, or meta-frameworks – Cloud services – API, Web Components • Software as a Service (SaaS) • Web standards such as document object model, accessibility Accessibility and Inclusive Design • Security and Privacy Considerations Computer Crimes and Attacks, Security Laws, Policies and Responsibility	• Compare and contrast web programming with general-purpose programming [Compare] • Describe the differences between Software-as-a-Service (SaaS) and traditional software products [Describe] • Discuss how web standards impact software development [Debate] • Evaluate an existing web application against current web standards [Evaluate]
Readings : [**Flanagan2020**], [**Duckett2014**], [**Fielding:2000:Rest**]	

Unit 4: Mobile Platforms: Development Fundamentals (6 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Development with – Mobile programming languages – Mobile programming environments • Mobile platform constraints – User interface design Understanding the User: Individual goals and interactions with others – Security • Access – Accessing data through APIs Query Construction – Designing API endpoints for mobile apps: pitfalls and design considerations – Network and web interfaces Data Modeling, Fundamentals of Networks and Communications	• Compare mobile programming with general-purpose programming [Compare] • Evaluate the pros and cons of native and cross-platform mobile application development [Evaluate]
Readings : [**Deitel2017**], [**Hellman2013**]	

Unit 5: Mobile Platforms: Architecture and Constraints (3 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Development – Native versus cross-platform development – Software design/architecture patterns for mobile applications API Design and Modeling, Principles and Software Architecture, Software Design Quality and Evaluation • Mobile platform constraints – Responsive user interface design Accessibility and Inclusive Design – Heterogeneity and mobility of devices – Differences in user experiences (e.g., between mobile and web-based applications) – Power and performance tradeoff • Specification and testing Software Development Practices, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices	• Develop a location-aware mobile application with data API integration [Create] • Build a sensor-driven mobile application capable of logging data on a remote server [Create] • Create a communication app incorporating telephony and instant messaging [Create]
Readings : [**Deitel2017**], [**Hellman2013**]	

Unit 6: Mobile Platforms: Capabilities and Asynchronous Computing (3 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Mobile computing affordances – Location-aware applications – Sensor-driven computing (e.g., gyroscope, accelerometer, health data from a watch) – Telephony and instant messaging – Augmented reality Immersion • Asynchronous computing Algorithms – Difference from traditional synchronous programming – Handling success via callbacks – Handling errors asynchronously – Testing asynchronous code and typical problems in testing	• Develop a location-aware mobile application with data API integration [Create] • Build a sensor-driven mobile application capable of logging data on a remote server [Create] • Create a communication app incorporating telephony and instant messaging [Create]
Readings : [**Deitel2017**], [**Hellman2013**]	

Unit 7: SPD-SEP/Interactive (10 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Ethical guidelines when using AI models to assist in journalism and content creation • Accountability for AI-generated outputs • Behavior among prompt programmers and AI developers • Trust with the public when using AI models	• Indicate a framework for accountability in AI model deployment, including clear documentation and attribution [Evaluate] • Discuss ethical codes of conduct and professional standards relevant to prompt programming and AI development [Debate] • Create communication plans and materials to educate the public about AI capabilities, limitations, and ethical safeguards [Create]
Readings : [**Duckett2014**], [**Deitel2017**]	

Unit 8: Interactive Data and Analytics Platforms (6 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Computational Notebook Platforms (e.g., Jupyter notebooks; Google Colab; R; SPSS; Observable) • Cloud SQL and Data Analysis Platforms (e.g., BigQuery) Query Construction – Apache Spark – Data Visualizations Visualization • Interactive Presentations Backed by Data • Design Tools Requiring Low-Latency Feedback Loops – Rendering tools – Graphic design tools	• Analyze large datasets interactively [Analyze] • Create a backing track for a musical performance, such as live coding [Create] • Create compelling computational notebooks that construct a narrative for a given journalistic goal/story [Create] • Implement interactive code that uses a dataset and generates exploratory graphics [Implement] • Contextualize the attributes of different data analysis styles, such as interactive vs engineered pipeline [Contextualize] • Write a program using a notebook computing platform (e.g., searching, sorting, or graph manipulation) [Write]
Readings : [**Flanagan2020**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI161. Applied AI (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	AI161. Applied AI
2.2 Semester	:	3rd Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BIC101. Introduction to Computing. (1st Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course provides a practical introduction to Artificial Intelligence (AI) for students from all scientific and engineering disciplines. Focused on developing AI literacy and practical skills, it covers fundamental concepts, modern AI tools (including Western and Chinese platforms), and responsible usage. Students will learn to effectively interact with diverse AI systems, write quality prompts, and apply AI solutions to problems across various domains while understanding ethical implications and cultural contexts of AI deployment.

5. GOALS

- Develop comprehensive AI literacy by understanding fundamental concepts, capabilities, and limitations of modern AI systems across different platforms and cultures.
- Master effective prompt engineering techniques and interaction patterns with various AI tools (Western: ChatGPT, Claude, Gemini; Chinese: DeepSeek, Kimi, ERNIE).
- Apply diverse AI tools to solve practical problems in scientific, engineering, and academic contexts while maintaining critical evaluation of outputs across platforms.
- Understand ethical considerations, biases, cultural contexts, and responsible usage of AI technologies in global professional and academic settings.
- Develop workflows that integrate multiple AI tools to enhance productivity and problem-solving capabilities while understanding regional strengths and specializations.

6. COMPETENCES

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Familiarity)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: AI Fundamentals and Global Landscape (16 hours)	
Competences Expected: AG-C01,AG-C11	
Topics	**Learning Outcomes**
• What is AI? Definitions, history, and current global landscape. • Types of AI systems: chatbots, image generators, research assistants across regions. • Western AI ecosystems: OpenAI, Anthropic, Google, Microsoft. • Chinese AI ecosystems: DeepSeek, Kimi, ERNIE, Zhipu AI, Baidu. • AI capabilities and limitations: comparative analysis across platforms. • Digital literacy in the AI era: critical thinking about diverse AI outputs.	• Apply fundamental AI knowledge to identify system types and their capabilities [Usar]. • Analyze global intelligent systems and their distinctive characteristics [Evaluar]. • Critically evaluate AI outputs across different cultural contexts [Evaluar].
Readings : [**Ng2024**], [**Mollick2024**], [**UNESCO2023**], [**Tencent2024**]	

Unit 2: Effective AI Interaction and Cross-Platform Prompt Engineering (20 hours)	
Competences Expected: AG-C01,AG-C11	
Topics	**Learning Outcomes**
• Principles of effective prompt writing: clarity, context, constraints across platforms. • Prompt patterns: persona, template, chain-of-thought, few-shot for different AI systems. • Western tools: specific techniques for ChatGPT, Claude, Gemini, Copilot. • Chinese tools: specific features and best practices for DeepSeek, Kimi, ERNIE. • Cross-platform strategies: leveraging different AI strengths for complex tasks. • Iterative refinement: how to improve prompts based on outputs from diverse systems. • Practical sessions: comparative prompt writing workshops across platforms.	• Design effective prompt solutions for diverse AI systems [Evaluar]. • Develop cross-platform workflows that integrate different AI tools [Usar]. • Implement iterative refinement strategies to optimize AI outputs [Usar].
Readings : [**White2023**], [**PromptEngineering2024**], [**Microsoft2024**], [**DeepSeek2024**]	

Unit 3: AI Applications Across Disciplines and Platforms (16 hours)	
Competences Expected: AG-C01,AG-C11	
Topics	**Learning Outcomes**
• AI for research: literature review, data analysis using multiple AI assistants. • AI for writing: academic papers, reports with cross-platform verification. • AI for problem-solving: scientific calculations, engineering design with specialized tools. • AI for creativity: brainstorming, concept development across cultural contexts. • Platform-specific strengths: when to use Western vs. Chinese AI tools. • Discipline-specific workshops: tailored applications using diverse AI ecosystems. • Case studies: real-world applications in scientific research across regions.	• Apply AI fundamentals to solve discipline-specific problems [Usar]. • Design solutions that integrate multiple intelligent systems for complex tasks [Evaluar]. • Develop practical applications using Western and Chinese AI tools [Usar]. • Implement workflows that leverage specific strengths of each platform [Evaluar].
Readings : [**Mollick2023**], [**Google2024**], [**Alibaba2024**], [**OpenAI2024**]	

Unit 4: Global AI Ethics and Responsible Usage (12 hours)	
Competences Expected: AG-C01,AG-C11	
Topics	**Learning Outcomes**
• Understanding AI biases: Western and Eastern cultural perspectives. • Ethical frameworks: comparing EU AI Act, Chinese regulations, and global standards. • Academic integrity: proper citation and AI usage in coursework across platforms. • Privacy and data security: regional differences in AI data handling. • Cultural sensitivity: navigating AI outputs in global contexts. • Environmental impact: sustainability considerations of different AI systems. • Developing personal guidelines for ethical AI usage in international settings. • Case studies: ethical dilemmas in AI deployment across regions.	• Demonstrate professional responsibility in using AI technologies [Evaluar]. • Apply professional ethical principles in global AI contexts [Usar]. • Develop ethical guidelines to mitigate biases in AI systems [Evaluar]. • Evaluate ethical implications of AI deployment across different cultures [Usar].
Readings : [**UNESCO2023**], [**EU2024**], [**Bender2021**], [**ChinaAI2024**], [**Tencent2023**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BMA104. Advanced Differential and Integral Calculus (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	BMA104. Advanced Differential and Integral Calculus
2.2 Semester	:	3rd Semester.
2.3 Credits	:	5
2.4 Hours	:	4 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BMA102. Differential Calculus. (1st Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Advanced differential and integral calculus provides the rigorous foundations for understanding multidimensional phenomena in engineering and computer science. This course covers vector functions, multivariable calculus, differentiable transformations, multiple integrals, and integral calculus on manifolds, including the fundamental theorems of Green, Stokes, and Gauss. These tools are essential for modeling complex systems, optimizing multivariable functions, and solving problems in computer graphics, computer vision, and physical simulations.

5. GOALS

- Study vector functions of a real variable and the differential geometry of space curves.
- Apply differential calculus to functions of several variables: partial derivatives, gradient, tangent plane, optimization, and Taylor's theorem.
- Analyze differentiable transformations between vector spaces using the Jacobian matrix and the inverse and implicit function theorems.
- Evaluate multiple integrals in different coordinate systems and compute areas, volumes, and centers of mass.
- Compute line integrals and apply Green's Theorem and the concept of conservative fields.
- Compute surface integrals and apply Stokes' Theorem and the Divergence Theorem.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Assessment)

7. TOPICS

Unit 1: Vector Functions of a Real Variable (14 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Definition, domain, and range of vector functions of a real variable • Limits, continuity, and derivatives of vector functions • Arc length and natural parametrization of a curve • Tangent, normal, and binormal vectors; moving Frenet frame • Curvature, torsion, and Frenet-Serret formulas	• Identify limits, continuity, and derivatives of vector functions of a real variable [Familiarity] • Compute tangent, normal, and binormal vectors and reparametrize a curve by arc length [Usage] • Calculate curvature and torsion using the Frenet-Serret formulas [Assessment]
Readings : [**Apostol1997**], [**MarsdenTromba2011**]	

Unit 2: Multivariable Differential Calculus (18 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Limits and continuity of functions of several variables in $\mathbb{R}^n$ • Partial derivatives, directional derivatives, and the gradient vector • Tangent plane, total differential, and linear approximation • Extrema of multivariable functions and Lagrange multipliers • Taylor's theorem for functions of several variables	• Evaluate limits and determine continuity of functions of several variables [Familiarity] • Compute gradients, directional derivatives, and tangent planes for multivariable functions [Usage] • Apply Lagrange multipliers to solve constrained optimization problems in computing and data science [Usage] • Expand functions of several variables using Taylor's theorem and estimate error bounds [Assessment]
Readings : [**Apostol1997**], [**MarsdenTromba2011**]	

Unit 3: Multivariable Transformations (12 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Transformations from $\mathbb{R}^n$ to $\mathbb{R}^m$ and their differentiability • Jacobian matrix and Jacobian determinant • Change of variables in polar, cylindrical, and spherical coordinates • Inverse function theorem • Implicit function theorem in multivariable contexts	• Describe transformations between coordinate systems using the Jacobian matrix [Familiarity] • Compute the Jacobian and apply change of variables to evaluate multiple integrals [Usage] • Apply the inverse and implicit function theorems to analyze local invertibility [Assessment]
Readings : [**Apostol1997**], [**MarsdenTromba2011**]	

Unit 4: Multiple Integrals (18 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Double integrals over rectangles and general regions; iterated integrals (Fubini's theorem) • Double integrals in polar coordinates and change-of-variables formula • Triple integrals in Cartesian coordinates and applications to volume and mass • Triple integrals in cylindrical and spherical coordinates • Applications: area, volume, centers of mass, and moments of inertia	• Evaluate double and triple integrals over general regions using iterated integrals [Usage] • Apply coordinate changes in polar, cylindrical, and spherical systems to simplify multiple integrals [Usage] • Compute areas, volumes, and centers of mass using multiple integrals [Assessment]
Readings : [**Apostol1997**], [**MarsdenTromba2011**]	

Unit 5: Line Integrals and Vector Fields (10 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Vector fields: definition, visualization, divergence, and curl • Line integrals of scalar functions: arc length and mass • Line integrals of vector fields: work and circulation • Conservative fields, potential functions, and path independence • Green's Theorem and its applications in the plane	• Compute scalar and vector line integrals over oriented curves [Usage] • Determine whether a vector field is conservative and construct its potential function [Usage] • Apply Green's Theorem to evaluate line integrals via double integrals [Assessment]
Readings : [**Apostol1997**], [**MarsdenTromba2011**]	

Unit 6: Surface Integrals and Fundamental Theorems (12 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Parametric surfaces, tangent plane, and surface area • Surface integrals of scalar functions • Flux integrals: surface integrals of vector fields • Stokes' Theorem: relation between line integrals and surface integrals • Divergence Theorem (Gauss): relation between flux and volume integral	• Parametrize surfaces and evaluate scalar and flux surface integrals [Usage] • Apply Stokes' and Divergence Theorems to reduce complex integrals [Assessment] • Relate local differential properties (curl, divergence) to global behavior via integral theorems [Assessment]
Readings : [**Apostol1997**], [**MarsdenTromba2011**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

STA251. Probability Calculation (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	STA251. Probability Calculation
2.2 Semester	:	3rd Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BMA102. Differential Calculus. (1st Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Probability and statistics provide the mathematical foundation for reasoning under uncertainty, analyzing data, and building predictive models – skills that are essential across computing, artificial intelligence, computational mathematics, and engineering alike. This course develops a rigorous understanding of probability theory and its use in describing random phenomena, together with the fundamentals of descriptive statistics, statistical inference (estimation, hypothesis testing, and Bayesian reasoning), and regression modeling, so that students can summarize and analyze data, quantify uncertainty, and draw sound quantitative conclusions in their own field of application.

5. GOALS

- Understand the fundamentals of probability theory and descriptive statistics.
- Apply probability distributions and statistical inference methods to analyze data and quantify uncertainty.
- Develop skills to model random phenomena and build predictive statistical models such as linear regression.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

7. TOPICS

Unit 1: Numerical Summaries: Location, Spread, and Shape (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Measures of location: mean, median, mode, trimmed mean, and weighted mean • Measures of spread: variance, standard deviation, IQR, MAD, and range • Measures of shape: skewness and kurtosis; quantiles and percentiles • Robust summary statistics: resistance to outliers and heavy-tailed distributions	• Calculate and interpret measures of location, spread, and shape for a univariate dataset [Usage] • Select appropriate summary statistics based on the distributional properties of the data [Usage] • Compare classical and robust summary statistics and justify the choice for a given dataset [Assessment]
Readings : [**Devore2016**]	

Unit 2: Sample Spaces and Events (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Random experiments, outcomes, and sample spaces: finite, countable, and uncountable • Set-theoretic operations on events: union, intersection, complement, and De Morgan's laws • Sigma-algebras and measurable spaces as the foundation for rigorous probability • Borel sets and the Borel sigma-algebra on the real line	• Define sample spaces, events, and sigma-algebras for a given random experiment [Familiarity] • Apply set-theoretic operations to compute probabilities of compound events [Usage] • Construct appropriate sigma-algebras for measurable spaces arising in statistical problems [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 3: Probability Axioms and Properties (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Kolmogorov's three axioms of probability and their consequences • Derived properties: complement rule, addition rule, monotonicity, and continuity of probability • Classical, frequentist, and subjective (Bayesian) interpretations of probability • Counting methods: permutations, combinations, and the multiplication principle	• State Kolmogorov's axioms and derive fundamental probability properties [Familiarity] • Calculate probabilities of events using the addition rule and complement rule [Usage] • Apply counting methods to compute probabilities in finite sample spaces [Usage]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 4: Conditional Probability and Independence (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Definition and interpretation of conditional probability P(A|B) and its properties • Bayes' theorem and the law of total probability: derivation and applications • Independence of events: pairwise and mutual independence • Conditional independence and its role in probabilistic graphical models	• Explain conditional probability and independence and their relationship [Familiarity] • Apply Bayes' theorem and the law of total probability to solve probability problems [Usage] • Analyze whether events are independent or conditionally independent in a given probability model [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 5: Discrete Random Variables and Distributions (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Probability mass functions (PMF) and cumulative distribution functions (CDF) for discrete random variables • Key discrete distributions: Bernoulli, Binomial, Geometric, Negative Binomial, Poisson, Hypergeometric • Properties of discrete distributions: mean, variance, skewness, and kurtosis • Poisson approximation to the Binomial and limiting distribution arguments	• Identify and describe the key discrete probability distributions and their parameters [Familiarity] • Calculate probabilities, means, and variances using PMFs of standard discrete distributions [Usage] • Select an appropriate discrete distribution to model a given random phenomenon and justify the choice [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 6: Continuous Random Variables and Distributions (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Probability density functions (PDF) and CDFs for continuous random variables • Key continuous distributions: Uniform, Normal, Exponential, Gamma, Beta, Chi-squared, t, F • The Normal distribution: properties, standard normal, and its central role in statistical inference • The exponential family of distributions: canonical form and sufficient statistics	• Describe the properties and parameters of key continuous probability distributions [Familiarity] • Calculate probabilities and quantiles for standard continuous distributions [Usage] • Identify distributions belonging to the exponential family and derive their sufficient statistics [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 7: Joint, Marginal, and Conditional Distributions (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Joint PMFs and PDFs for two or more random variables • Marginal and conditional distributions derived from joint distributions • Covariance, correlation, and their properties; independence implies zero covariance (converse not always true) • The bivariate normal distribution and its conditional distributions	• Derive marginal and conditional distributions from a given joint distribution [Usage] • Calculate covariance and correlation between two random variables [Usage] • Analyze the conditional structure of the bivariate normal distribution [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 8: Limit Theorems: LLN and CLT (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Weak law of large numbers: convergence in probability and its implications for estimation • Central limit theorem: statement, conditions, and role in justifying normal approximations • Applications of the CLT: normal approximation to the binomial and sampling distributions • Strong law of large numbers: almost sure convergence • The delta method: asymptotic distribution of smooth functions of estimators	• State the LLN and CLT and explain the conditions under which they hold [Familiarity] • Apply the CLT to derive normal approximations and sampling distributions [Usage] • Apply the delta method to derive asymptotic distributions of nonlinear estimators [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 9: Point Estimation (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Desirable properties: unbiasedness, consistency, efficiency, and sufficiency • Method of moments estimation: derivation and asymptotic properties • Maximum likelihood estimation: derivation, properties, and invariance principle • Cramer-Rao lower bound and the concept of Fisher information	• Describe the key properties of estimators and explain the trade-offs between them [Familiarity] • Derive method of moments and MLE estimators for standard distributions [Usage] • Evaluate the efficiency of an estimator using the Cramer-Rao lower bound [Assessment]
Readings : [**Devore2016**]	

Unit 10: Interval Estimation and Confidence Intervals (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Constructing confidence intervals using pivotal quantities and exact distributions • Correct interpretation of confidence intervals: coverage probability and common misconceptions • Large-sample confidence intervals based on asymptotic normality • Duality between confidence intervals and hypothesis tests	• Construct confidence intervals for means, proportions, and variances using exact and large-sample methods [Usage] • Interpret confidence intervals correctly and identify common misinterpretations [Usage] • Explain the duality between confidence intervals and hypothesis tests and use it to draw inferences [Assessment]
Readings : [**Devore2016**]	

Unit 11: Hypothesis Testing Foundations (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Neyman-Pearson framework: null/alternative hypotheses, rejection regions, Type I and II errors • p-values: definition, correct interpretation, and common misuse; power and sample size • Common parametric tests: z-test, t-test, F-test, chi-squared test • Neyman-Pearson lemma and uniformly most powerful tests	• Explain the logic of hypothesis testing, Type I and II errors, and the meaning of a p-value [Familiarity] • Apply standard parametric tests and interpret the results in context [Usage] • Derive the most powerful test for a simple hypothesis using the Neyman-Pearson lemma [Assessment]
Readings : [**Devore2016**]	

Unit 12: Bayesian Inference (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Bayesian framework: prior, likelihood, posterior, and the role of Bayes' theorem • Prior distributions: informative, non-informative, and conjugate priors • Posterior inference: point estimates, credible intervals, and posterior predictive distributions • Bayesian model comparison: Bayes factors and posterior model probabilities	• Explain the Bayesian approach to inference and contrast it with the frequentist approach [Familiarity] • Derive the posterior distribution for standard conjugate models [Usage] • Apply Bayes factors to compare statistical models and interpret the evidence [Assessment]
Readings : [**Ross2014**], [**Devore2016**]	

Unit 13: Simple Linear Regression (4 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Ordinary least squares estimation: derivation of coefficients and their properties • Inference for regression coefficients: t-tests, confidence intervals, and p-values • Coefficient of determination R-squared and analysis of variance decomposition • Gauss-Markov theorem: BLUE property of OLS under classical assumptions	• Fit a simple linear regression model and interpret the estimated coefficients [Usage] • Conduct hypothesis tests and construct confidence intervals for regression parameters [Usage] • Verify the Gauss-Markov assumptions and assess their consequences when violated [Assessment]
Readings : [**Devore2016**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BEI201. Technical and professional English III (Elective)

2. GENERAL INFORMATION

2.1 Course	:	BEI201. Technical and professional English III
2.2 Semester	:	3rd Semester.
2.3 Credits	:	2
2.4 Hours	:	4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BEI102. Technical and professional English II. (2nd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

At this level, students engage with English as the primary medium for accessing and producing technical content in computing, engineering, and the sciences. Grammar structures are applied to reading scientific articles, writing technical reports, describing engineering processes, and participating in professional discussions. The course bridges everyday English fluency with the demands of technical communication.

5. GOALS

- Train students to read, understand, and produce technical communication in English.
- Provide linguistic tools for connecting ideas in descriptive and analytical writing about computing and engineering.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Technical Communication (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Review of Tenses: Present, Past, and Future in technical documentation. • Interrogative sentences with complex Wh- questions for system analysis. • Words with more than one meaning in technical contexts (Polysemy). • Expressions for describing professional activities and research interests.	• By the end of the unit, students use various tenses to exchange detailed technical information and formulate precise questions about computing systems and engineering processes.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 2: Comparing Architectures (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Use of "What's it like?": Describing system characteristics and interfaces. • Advanced comparative and superlative adjectives in benchmark analysis. • Collocations in computing: high-performance, low-level, fault-tolerant. • Reading: Architecture comparison reports (RISC vs. CISC, SQL vs. NoSQL).	• Students analyze and describe differences between computing architectures, frameworks, and systems using varied comparative structures with technical precision.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 3: Then and Now in Computing (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Perfect vs. Simple Past: Contrasting historic events with current state. • Time expressions: for, since, ever, never applied to technology evolution. • Adverbs of time and manner in scientific narrative. • Vocabulary: Generations of technology, legacy systems, current platforms.	• Students differentiate and correctly apply the present perfect and the simple past to narrate the evolution of computing technologies and contrast historical achievements with the current state of the field.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 4: Engineering Standards and Protocols (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Modals of obligation and permission: Have to, Must, Can in professional norms. • Giving technical recommendations: Should, Ought to — code reviews and design. • Vocabulary: IEEE/ACM standards, coding conventions, security policies. • Formal writing: Technical memos and engineering recommendations.	• Students express engineering rules, standards, and professional recommendations using appropriate modal verbs in formal technical writing and peer discussion.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 5: How Systems Are Built (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Passive Voice: Present and Simple Past to describe engineering processes. • Vocabulary: Manufacturing, deployment, integration, and testing. • Expressions for giving technical opinions and discussing design trade-offs. • Writing: Process description for a software or hardware system.	• Students use the passive voice to describe how systems are designed, built, tested, and deployed, participating in technical discussions about engineering processes.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 6: Debugging and Analysis (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Past Continuous vs. Simple Past in incident and failure reports. • Narrative and sequence connectors: while, when, suddenly, as a result. • Vocabulary: Bugs, exceptions, crashes, logs, root cause analysis. • Writing: Incident report for a software or network failure.	• Students narrate debugging sessions, system failures, and experimental incidents using narrative tenses and technical sequencing connectors fluently.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 7: Research and Innovation (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Future Forms: Will, Going to, Present Continuous for research projection. • Expressions of probability and certainty in scientific contexts. • Vocabulary: Innovation cycles, research proposals, technology forecasting. • Oral presentations: "The future of computing in my field".	• Students project research goals and discuss future directions in computing and engineering using various future structures with appropriate degrees of certainty.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 8: Technical English III (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Programming and Web Development terminology: functions, APIs, frameworks. • User Interface (UI/UX) descriptions and usability concepts. • Basic project management vocabulary: sprints, milestones, deliverables. • Writing: Simple technical specification or README document.	• By the end of the course, students use specialized software development vocabulary to describe logical and functional components and produce basic technical documentation in English.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS210. Algorithms and Data Structures (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS210. Algorithms and Data Structures
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS113. Objects-oriented programming II. (3^{rd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course provides a comprehensive study of fundamental data structures and algorithms with emphasis on implementation using advanced C++ features. Building on prior C++ knowledge, students will implement generic data structures using templates and traits, ensuring type flexibility and performance. The course covers linear structures (vectors, various linked lists), tree structures (binary, AVL, B-trees), heaps, hash tables, tries, and concurrent data structures. Special focus is placed on modern C++ techniques including variadic templates for generic operations and concurrent programming for thread-safe data structures.

5. GOALS

- Implement generic data structures using templates and traits.
- Master advanced tree structures and balanced trees.
- Understand and implement concurrent data structures.
- Apply variadic templates for generic operations on containers.
- Analyze algorithmic complexity of all implemented structures.
- Design efficient data structures for specific problem constraints.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Advanced Foundation for Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Template metaprogramming review and advanced traits. • Type traits and SFINAE for conditional compilation. • Variadic templates and parameter packs. • Perfect forwarding and universal references. • Lambda expressions with generic captures. • Policy-based design and CRTP (Curiously Recurring Template Pattern). • Concept-based constraints (C++20 concepts if applicable).	• Design type traits for customizing data structure behavior [Usar]. • Implement variadic templates for generic container operations [Usar]. • Apply policy-based design to create configurable data structures [Evaluar]. • Use perfect forwarding to implement efficient constructors [Usar].
Readings : [**Stroustrup2013**], [**Vandevoorde2017**]	

Unit 2: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Abstract Data Type (ADT) and operations on an ADT: – Dictionary operations (insert, delete, find) • Arrays: – Numeric vs non-numeric, character strings – Single (vector) vs multidimensional (matrix) • Records/Structs/Tuples and Objects • Linked lists (for historical reasons): – Single vs Double and Linear vs Circular • Stacks • Queues and deques: – Heap-based priority queue • Hash tables/maps: – Collision resolution and complexity (e.g., probing, chaining, rehash) • Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Sets	• For each ADT/Data-Structure in this unit: – Explain its definition, properties, representation(s), and associated ADT operations. – Explain step-by-step how the ADT operations associated with the data structure transform it. [Usar] • For each algorithm in this unit explain step-by-step how the algorithm operates [Usar] • Implement vector with customizable allocation policy using templates [Usar]. • Design generic linked list nodes with trait-based memory management [Usar]. • Compare time/space complexity of different linear structures [Evaluar]. • Apply variadic templates to initialize containers with multiple elements [Usar].
Readings : [**Cormen2009**], [**Stroustrup2013**]	

Unit 3: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Binary tree template implementation with iterator support. • Binary search tree with trait-based comparison operators. • Tree traversal algorithms (iterative and recursive). • Expression trees and their applications. • Template-based node structures with configurable payload.	• Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Evaluar] • Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Familiarizarse] • Implement generic binary tree with configurable traversal order [Usar]. • Design expression tree evaluator using visitor pattern [Usar]. • Analyze tree operation complexities for different balancing strategies [Evaluar].
Readings : [**Cormen2009**], [**Knuth1997**]	

Unit 4: Balanced and Multi-way Trees (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• AVL tree implementation with template-based balancing trait. • Red-black tree properties and operations. • B-Tree template implementation for disk-based storage. • 2-3 Tree and 2-3-4 Tree structures. • Digital trie (prefix tree) implementation. • Suffix trees and their applications in string processing. • Node splitting/merging strategies for B-trees.	• Implement self-balancing trees using policy-based design [Usar]. • Design B-tree for database index applications [Usar]. • Evaluate trade-offs between different balanced tree structures [Evaluar]. • Implement trie for efficient string prefix matching [Usar].
Readings : [**Cormen2009**], [**Knuth1997**], [**Sedgewick2011**]	

Unit 5: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Binary heap implementation using array representation. • Min-heap and max-heap variants. • Heapify algorithms and complexity analysis. • Priority queue ADT implementation. • Binomial heap and Fibonacci heap overview. • Heap applications: Dijkstra's algorithm, heap sort.	• Explain the heap property and the use of heaps as an implementation of a priority queue [Familiarizarse] • Implement generic heap with configurable comparison trait [Usar]. • Design priority queue supporting multiple priority update strategies [Usar]. • Analyze heap operations complexity for different heap types [Evaluar]. • Apply heap in graph algorithm implementations [Usar].
Readings : [**Cormen2009**], [**Sedgewick2011**]	

Unit 6: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Hash tables/maps: – Collision resolution and complexity (e.g., probing, chaining, rehash) • Hash function design and properties. • Separate chaining implementation with linked lists. • Open addressing: linear probing, quadratic probing, double hashing. • Perfect hashing and minimal perfect hashing. • Cuckoo hashing and hopscotch hashing. • Hash table resizing strategies and load factor management.	• Explain how collision avoidance and collision resolution is handled in hash tables [Familiarizarse] • Design generic hash table with configurable collision resolution [Usar]. • Implement custom hash functions for user-defined types [Usar]. • Evaluate performance of different hashing schemes [Evaluar]. • Apply hash tables in compiler symbol table implementation [Usar].
Readings : [**Cormen2009**], [**Knuth1997**]	

Unit 7: Concurrent Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Thread-safe data structure design principles. • Lock-based concurrent linked lists. • Lock-free and wait-free algorithms. • Concurrent hash tables with fine-grained locking. • Atomic operations and memory ordering. • Concurrent queues (bounded and unbounded). • Concurrent stacks and elimination techniques.	• Design thread-safe data structures using mutex and lock guards [Usar]. • Implement lock-free linked list using atomic operations [Usar]. • Analyze performance of concurrent vs sequential data structures [Evaluar]. • Apply concurrent queues in producer-consumer patterns [Usar].
Readings : [**Herlihy2012**], [**Williams2019**]	

Unit 8: Specialized Structures and Applications (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Persistent data structures. • Disjoint-set union (union-find) with path compression. • Bloom filters and their probabilistic properties. • Skip lists for probabilistic balanced search. • Segment trees and Fenwick trees for range queries. • Spatial data structures: k-d trees, quadtrees. • Cache-oblivious data structures.	• Implement persistent data structures with structural sharing [Usar]. • Design union-find with both union by rank and path compression [Usar]. • Apply specialized structures to solve domain-specific problems [Evaluar]. • Evaluate space-time tradeoffs in probabilistic data structures [Evaluar].
Readings : [**Okasaki1999**], [**Sedgewick2011**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS211. Theory of Computation (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS211. Theory of Computation
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS1D1. Discrete Structures. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Theory of Computation provides the mathematical foundations to understand what can be computed and how efficiently. It introduces formal languages, automata, and the limits of algorithmic solvability. This knowledge is fundamental for understanding compiler construction, formal verification, and the inherent complexity of computational problems.

5. GOALS

- Master the concepts of finite automata and regular expressions.
- Analyze context-free grammars and pushdown automata.
- Understand the Universal Turing Machine as a model of computation.
- Comprehend the boundaries of computability and the Halting Problem.
- Differentiate between complexity classes like P and NP.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Formal Languages and Automata (18 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal automata: – Finite State – Pushdown – Linear Bounded – Turing Machine • Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0)) • Deterministic and nondeterministic automata • Pumping Lemma proofs: – Proof of Finite State/Regular-Language limitation – Pushdown Automata/Context-Free-Language limitation	• For each formal automaton in this unit: – Explain its definition comparing its characteristics with this unit's other automata – Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input – Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] • Given a problem, develop an appropriate automaton that addresses the problem [Create] • Develop a regular expression for a given regular language expressed in natural language [Create] • Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply] • Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply]
Readings : [**Sipser2012**], [**Hopcroft2013**]	

Unit 2: Formal Languages and Automata (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal automata: – Finite State – Pushdown – Linear Bounded – Turing Machine • Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0) • Normal Forms: Chomsky and Greibach • Pumping Lemma proofs: – Proof of Finite State/Regular-Language limitation – Pushdown Automata/Context-Free-Language limitation • Relations among formal automata, languages, and grammars	• For each formal automaton in this unit: – Explain its definition comparing its characteristics with this unit's other automata – Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input – Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] • Given a problem, develop an appropriate automaton that addresses the problem [Create] • Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply] • Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply] • Construct CFGs for formal language descriptions [Evaluar]
Readings : [**Sipser2012**], [**Hopcroft2013**]	

Unit 3: Formal Languages and Automata (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal automata: – Finite State – Pushdown – Linear Bounded – Turing Machine • Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0) • Decidability, (un)computability, and halting • The Church-Turing thesis • Reducibility and reductions • Equivalent models of algorithmic computation: – Turing Machines and Variations (e.g., multi-tape, non-deterministic) – Lambda Calculus – Mu-Recursive Functions	• For each formal automaton in this unit: – Explain its definition comparing its characteristics with this unit's other automata – Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input – Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] • Explain a universal Turing Machine and its operation [Explain] • Present to an audience of co-workers and managers the impossibility of providing them a program that checks all other programs, including some seemingly simple ones, for infinite loops including an explanation of the Halting problem, why it has no algorithmic solution, and its significance for real-world algorithmic computation [Explain] • Explain the Church-Turing Thesis and its significance for algorithmic computation [Explain] • Apply arithmetization and diagonalization to prove the Halting Problem for Turing Machines is Undecidability [Apply] • Given a known undecidable language, apply a mapping reduction or computational history to prove that another language is undecidable [Apply]
Readings : [**Sipser2012**], [**Kozen2006**]	

Unit 4: Formal Languages and Automata (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0) • Relations among formal automata, languages, and grammars • Equivalent models of algorithmic computation: – Turing Machines and Variations (e.g., multi-tape, non-deterministic) – Lambda Calculus – Mu-Recursive Functions • Context-Sensitive Languages (Type-1) and applications	• For each formal model in this unit: – Explain its definition comparing its characteristics with the others in this unit – Explain example inputs that are and cannot be accepted by the language/grammar. [Explain] • Explain the Primitive and General Recursive functions (zero, successor, selection, primitive recursion, composition, and Mu), their significance, and Turing Machine implementations [Explain] • Explain how computation is performed in Lambda Calculus (e.g., Alpha conversion and Beta reduction) [Explain] • Explain Rice's theorem and its significance [Explain]
Readings : [**Sipser2012**], [**Kozen2006**]	

Unit 5: Complexity Analysis Framework (3 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Complexity Analysis Framework: – Best, average, and worst-case performance of an algorithm – Empirical and relative (Order of Growth) measurements – Input size and primitive operations – Time and space efficiency	• Prepare a presentation that explains to first year students the basic concepts of algorithmic complexity including best, average, and worst-case algorithm behavior, Big-O, Omega, and Theta notations, complexity classes, time and space tradeoffs, empirical measurement, and impact on practical problems [Explain] • Informally evaluate the foundational complexity class of simple algorithms [Evaluate]
Readings : [**Sipser2012**], [**Cormen2022**]	

Unit 6: Asymptotic Notation and Complexity Classes (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Asymptotic complexity analysis (average and worst-case bounds): – Big-O, Big-Omega, and Big-Theta formal notations – Foundational Complexity Classes and Representative Examples/Problems: ∗ $O(1)$ Constant (e.g., array access) ∗ $O(\log_2 n)$ Logarithmic (e.g., binary search) ∗ $O(n)$ Linear (e.g., linear search) ∗ $O(n \log_2 n)$ Log Linear (e.g., mergesort) ∗ $O(n^2)$ Quadratic (e.g., selection sort) ∗ $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) ∗ $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) ∗ $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)	• Using examples, explain each of the foundational complexity classes in this unit [Explain] • Explain to a non-technical audience the significance of tractable versus intractable algorithms using an intuitive explanation of Big-O complexity [Explain]
Readings : [**Sipser2012**], [**Cormen2022**]	

Unit 7: Complexity Analysis: Recursion, Amortization, and Tight Bounds (7 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Little-o, Little-Omega, and Little Theta notations • Formal recursive analysis	• Use recurrence relations to evaluate the time complexity of recursively defined algorithms [Apply] • Apply elementary recurrence relations using a form of the Master Theorem [Apply]
Readings : [**Sipser2012**], [**Cormen2022**]	

Unit 8: Computational Complexity Theory (6 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Tractability and intractability: – P, NP, and NP-Complete Complexity Classes – NP-Complete Problems (e.g., SAT, Knapsack, TSP) – Reductions • Turing Machine-based models of complexity: – Time complexity: ∗ P, NP, NP-C, and EXP classes ∗ Cook-Levin theorem – Space Complexity: ∗ NSpace and PSpace ∗ Savitch's theorem	• Explain the significance of NP-Completeness [Explain] • Explain how NP-Hard is a lower bound and NP is an upper bound for NP-Completeness [Explain] • Explain examples of NP-complete problems [Explain] • Explain the Cook-Levin Theorem and the NP-Completeness of SAT [Explain] • Explain the classes P and NP [Explain] • Prove that a problem is NP-Complete by reducing a classic known NP-C problem to it (e.g., 3SAT and Clique) [Create] • Explain the P-space class and its relation to the EXP class [Explain]
Readings : [**Sipser2012**], [**Cormen2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS221. Computer Systems Architecture (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS221. Computer Systems Architecture
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS1D1. Discrete Structures. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

A deep understanding of computer architecture is essential for developing high-performance software and understanding the hardware-software interface. This course explores how computers execute instructions, manage memory, and communicate with external devices. By studying the internal organization of processors and the hierarchy of memory systems, students gain the ability to optimize code for specific hardware architectures and understand the trade-offs in modern system design.

5. GOALS

- Understand the principles of digital logic and data representation.
- Understand the organization of the Central Processing Unit (CPU).
- Master the basics of assembly language and instruction set architecture.
- Analyze the memory hierarchy, including cache memory and virtual memory.
- Evaluate performance improvements through pipelining and parallelism.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Assembly Level Machine Organization (20 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• von Neumann machine architecture • Control unit: instruction fetch, decode, and execution • Introduction to SIMD vs MIMD and the Flynn taxonomy • Shared memory multiprocessors/multicore organization • Instruction set architecture (ISA) (e.g., x86, ARM and RISC-V): – Fixed vs variable-width instruction sets – Instruction formats – Data manipulation, control, I/O – Addressing modes – Machine language programming – Assembly language programming • Subroutine call and return mechanisms • I/O and interrupts • Heap, static, stack, and code segments	• Discuss how the classical von Neumann functional units are implemented in embedded systems, particularly on-chip and off-chip memory [Debate] • Describe how instructions are executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution [Describe] • Assess an example diagram with instruction-level parallelism and hazards to describe how they are managed in typical processor pipelines [Assess] • Discuss how instructions are represented at the machine level and in the context of a symbolic assembler [Debate] • Map an example of high-level language patterns into assembly/machine language notations [Map] • Contrast different instruction formats considering aspects such as addresses per instruction and variable-length vs fixed-length formats [Contrast] • Analyze a subroutine diagram to comment on how subroutine calls are handled at the assembly level [Analyze] • Describe basic concepts of interrupts and I/O operations [Describe] • Write a simple assembly language program for string/array processing and manipulation [Write]
Readings : [**Patterson2020**], [**Harris2012**]	

Unit 2: Memory Hierarchy (18 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Memory hierarchy: the importance of temporal and spatial locality • Main memory organization and operations • Persistent memory (e.g., SSD, standard disks) • Latency, cycle time, bandwidth, and interleaving • Cache memories: – Address mapping – Block size – Replacement and store policy – Prefetching • Multiprocessor cache coherence • Virtual memory (hardware support) • Fault handling and reliability • Reliability: – Error coding – Data compression – Data integrity • Processing In-Memory (PIM)	• Using a memory system diagram, identify the main types of memory technology (e.g., SRAM, DRAM) and their relative cost and performance [Analyze] • Measure the effect of memory latency on running time [Evaluate] • Enumerate the functions of a system with virtual memory management [Enumerate] • Compute average memory access time under various cache and memory configurations and mixes of instruction and data references [Compute]
Readings : [**Patterson2020**], [**Stallings2015**]	

Unit 3: Interfacing and Communication (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• I/O fundamentals: – Handshaking and buffering – Programmed I/O – Interrupt-driven I/O • Interrupt structures: vectored and prioritized, interrupt acknowledgment • I/O devices (e.g., mouse, keyboard, display, camera, sensors, actuators) • External storage, physical organization, and drives • Buses fundamentals: – Bus protocols – Arbitration – Direct-memory access (DMA)	• Analyze an interrupt control diagram to comment on how interrupts are used to implement I/O control and data transfers [Analyze] • Enumerate various types of buses in a computer system [Enumerate] • List the advantages of magnetic disks and contrast them with those of solid-state disks [List]
Readings : [**Patterson2020**], [**Tanenbaum2012**]	

Unit 4: Functional Organization (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Implementation of simple datapaths, including instruction pipelining, hazard detection, and resolution (e.g., stalls, forwarding) • Control unit: – Hardwired implementation – Microprogrammed realization • Instruction pipelining • Introduction to instruction-level parallelism (ILP)	• Compare alternative implementation of datapaths in modern computer architectures [Compare] • Produce a set of control signals for adding two integers using hardwired and microprogrammed implementations [Produce] • Discuss instruction-level parallelism using pipelining and significant hazards that may occur [Debate] • Design a complete processor, including datapath and control [Design] • Compute the average cycles per instruction for a given processor and memory system implementation [Compute]
Readings : [**Harris2012**], [**Patterson2020**]	

Unit 5: Heterogeneous Architectures (10 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• SIMD and MIMD architectures (e.g., General-Purpose GPUs, TPUs, and NPUs) • Heterogeneous memory systems: – Shared memory versus distributed memory – Volatile vs non-volatile memory – Coherence protocols • Domain-Specific Architectures (DSAs): – Machine Learning Accelerator – In-networking computing – Embedded systems for emerging applications – Neuromorphic computing – Edge computing devices • Packaging and integration solutions such as 3DIC and chiplets • Machine learning in architecture design: – AI algorithms for workload analysis – Optimization of architecture configurations for performance and power efficiency • Data-Oriented Design (DOD): organizing data in memory (array-of-structures vs. structure-of-arrays) to maximize cache locality and exploit SIMD vectorization; contrast and complementarity with the object-oriented paradigm.	• Analyze a system diagram with alternative parallel architectures, e.g., SIMD and MIMD, and identify the key differences [Evaluar] • Discuss what memory-management issues are found in multiprocessors that are not present in uniprocessors and how these issues might be resolved [Evaluar] • Indicate the differences between memory backplane, processor memory interconnect, and remote memory via networks, their implications for access latency, and their impact on program performance [Familiarizarse] • Discuss how you would determine when to use a domain-specific accelerator instead of a general-purpose CPU [Evaluar] • Enumerate key differences in architectural design principles between a vector and scalar-based processing unit [Familiarizarse] • List the advantages and disadvantages of a PIM architecture [Familiarizarse] • Restructure a collection of objects (AoS) into a DOD layout (SoA) and measure the performance impact [Usar].
Readings : [**Patterson2020**], [**Stallings2015**]	

Unit 6: Secure Processor Architectures (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Principles of Secure Hardware: – Security Risk Analysis, Asset Protection, and Threat Model – Cryptographic Acceleration with Hardware – Support for virtualization (e.g., OS isolation) • Roots of trust in hardware, Physically Unclonable Functions (PUF) • Hardware Random Number Generators • Memory protection extensions: – Runtime pointer bounds checking (e.g., buffer overflow) – Protection at the microarchitectural level – Protection at the ISA level • Trusted Execution Environment (TEE): – Trusted Computer Base Protections – Protecting virtual machines – Protecting containers – Trusted software modules (Enclaves) • Homomorphic encryption for privacy-preserving data processing	• Discuss principles of secure hardware, exploring a framework for risk analysis and asset protection [Debate] • Summarize how Physically Unclonable Functions (PUF) can be a unique device identifier in security applications [Summarize] • Distinguish a random number generator with dedicated hardware support from generators without hardware dedicated to generating entropy [Distinguish] • List the advantages and disadvantages of memory protection at the ISA level [List] • Describe key design issues of a trusted execution environment (TEE) to support virtual machines [Describe]
Readings : [**Patterson2020**], [**Anderson2020**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS271. Data Management (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS271. Data Management
2.2 Semester	:	4th Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	• CS112. Objects-oriented programming I. (2nd Sem) • CS1D1. Discrete Structures. (2nd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course introduces fundamental concepts and techniques in data management, focusing on relational database systems. Students will learn data modeling using entity-relationship diagrams, relational database design principles, normalization theory, and Structured Query Language (SQL). The course covers core database system components, transaction management, and basic query processing. Emphasis is placed on practical skills for designing, implementing, and querying databases, as well as understanding the role of data in organizations and the associated ethical considerations in data management.

5. GOALS

- Design and implement normalized relational databases.
- Formulate complex queries using SQL for data retrieval and manipulation.
- Apply data modeling techniques using entity-relationship diagrams.
- Understand database system architecture and transaction management.
- Recognize ethical and security considerations in data management.
- Utilize basic database administration tools and techniques.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Core Database System Concepts (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Purpose and advantages of database systems • Components of database systems • Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods) • Database architecture, data independence, and data abstraction • Transaction management • Normalization • Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce) • How to support CRUD-only applications • Distributed databases/cloud-based systems • Structured, semi-structured, and unstructured data • Use of a declarative query language • Systems supporting structured and/or stream content	• Identify at least four advantages that using a database system provides [Familiarizarse] • Enumerate the components of a (relational) database system [Familiarizarse] • Explain the advantages of database systems over file systems [Usar]. • Describe the three-schema architecture and data independence [Familiarizarse]. • Identify different types of database users and their roles [Familiarizarse].
Readings : [**Ramakrishnan2002**], [**Silberschatz2019**]	

Unit 2: Data Modeling (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Conceptual models (e.g., entity-relationship, UML diagrams) • Entity sets, relationship sets, and attributes • Cardinality constraints: one-to-one, one-to-many, many-to-many • Weak entity sets and identifying relationships • Extended E-R features: specialization, generalization, aggregation • UML class diagrams for data modeling	• Describe the components of the E-R (or some other non-relational) data model [Familiarizarse] • Model a given environment using a conceptual data model [Usar] • Design E-R diagrams for real-world scenarios [Usar]. • Convert E-R diagrams to relational schemas [Usar]. • Apply extended E-R features for complex modeling [Evaluar].
Readings : [**Ramakrishnan2002**], [**Elmasri2015**]	

Unit 3: Relational Databases (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Entity and referential integrity: Candidate key, superkeys • Relational database design • Relational algebra operations: select, project, join, union, etc. • Integrity constraints: domain, key, entity, referential • Relational calculus: tuple and domain calculus • Mapping E-R diagrams to relational schemas	• Describe the defining characteristics behind the relational data model [Familiarizarse] • Comment on the difference between a foreign key and a superkey [Usar] • Express queries using relational algebra [Usar]. • Design relational schemas with appropriate constraints [Usar]. • Map complex E-R diagrams to normalized relations [Evaluar].
Readings : [**Ramakrishnan2002**], [**Elmasri2015**]	

Unit 4: Query Construction (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• SQL Query Formation: – Interactive SQL execution – Programmatic execution of an SQL query • Data definition language: CREATE, ALTER, DROP • Data types and domain constraints reation and management • View definition and materialized views • Database schema design and implementation	• Compose SQL queries that incorporate select, project, join, union, intersection, set difference, and set division [Usar] • Define, in SQL, a relation schema, including all integrity constraints and delete/update triggers [Usar] • Create and modify database schemas using SQL DDL [Usar]. • Implement integrity constraints at schema level [Usar]. • Design and create appropriate indexes for performance [Evaluar].
Readings : [**Ramakrishnan2002**], [**Beaulieu2009**]	

Unit 5: Query Construction (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Complex SQL queries: nested subqueries, correlated subqueries • Aggregation functions and GROUP BY operations • Set operations: UNION, INTERSECT, EXCEPT • JOIN operations: inner, outer, natural, cross joins • Data manipulation: INSERT, UPDATE, DELETE • Transaction control: COMMIT, ROLLBACK, SAVEPOINT	• Determine when a nested SQL query is correlated or not [Usar] • Compose an SQL query to update a tuple in a relation [Usar] • Formulate complex queries with multiple joins and subqueries [Usar]. • Apply aggregation for data analysis and reporting [Usar]. • Implement data modification operations with transaction control [Evaluar].
Readings : [**Ramakrishnan2002**], [**Beaulieu2009**]	

Unit 6: Relational Databases (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Introduction to Functional dependency theory • Normalization Theory: – Decomposition of a schema; lossless-join, and dependency-preservation properties of a decomposition – Normal forms (BCNF) – Denormalization (for efficiency) • Functional dependencies and inference rules • Normal forms: 1NF, 2NF, 3NF, BCNF • Decomposition algorithms: lossless-join and dependency-preserving • Practical considerations in normalization	• Describe how functional dependency theory generalizes the notion of key [Familiarizarse] • Detect which normal form a given decomposition yields [Usar] • Identify functional dependencies in relational schemas [Usar]. • Normalize relations to BCNF [Evaluar]. • Evaluate decompositions for lossless-join and dependency preservation [Evaluar].
Readings : [**Ramakrishnan2002**], [**Elmasri2015**]	

Unit 7: DBMS Internals (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Transaction Management: – Isolation Levels – ACID – Serializability – Distributed Transactions • Concurrency Control: – 2-Phase Locking – Deadlocks handling strategies – Quorum-based consistency models • ACID properties: Atomicity, Consistency, Isolation, Durability • Serializability and conflict serializability • Lock-based concurrency control: two-phase locking • Deadlock detection and prevention • Isolation levels and their trade-offs	• Describe the four properties for a correct transaction manager [Familiarizarse] • Outline the principle of serializability [Familiarizarse] • Explain ACID properties and their importance [Usar]. • Analyze transaction schedules for serializability [Evaluar]. • Apply locking protocols for concurrency control [Usar].
Readings : [**Ramakrishnan2002**], [**Bernstein1987**]	

Unit 8: Data Security and Privacy (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Differences between data security and data privacy • Personally identifying information (PII) and its protection • Ethical considerations in ensuring the security and privacy of data • Database security: authentication, authorization, encryption • SQL injection attacks and prevention • Introduction to NoSQL databases and big data • Data governance and compliance	• Describe the differences in the goals for data security and data privacy [Familiarizarse] • Identify and mitigate risks associated with different approaches to protecting data [Usar] • Implement basic database security measures [Usar]. • Prevent SQL injection vulnerabilities [Evaluar]. • Discuss ethical considerations in data collection and use [Evaluar].
Readings : [**Ramakrishnan2002**], [**Elmasri2015**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS401. Research Methodology (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS401. Research Methodology
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	1 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS100. Introduction to Computer Science. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course aims to teach students how to conduct scientific research in the field of computer science. The course instructors will determine a specific study area for each student and provide relevant bibliography for analysis. Based on this and additional bibliographic sources researched by the student, the student should be able to construct a survey-type article on the assigned topic.

5. GOALS

- Learn how to initiate scientific research in computer science with methodological rigor.
- Know the main academic bibliographic sources: IEEE Xplore, ACM Digital Library, Google Scholar, and arXiv.
- Critically analyze existing proposals on a topic and relate them coherently in a literature review.
- Write technical documents in computer science using LaTeXwith IEEE and ACM conference templates.
- Reproduce existing results in a topic through implementation and experimentation.
- Build a survey-type article with taxonomy, synthesis, and comparison of existing approaches.
- The deliverables for this course are:

  Partial Progress: Mastery of the article topic and preliminary bibliography in LaTeXarticle format.

  Final: Completed survey-type article, optionally containing experimental results of the studied technique(s).

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Foundations of Scientific Research in Computer Science (8 hours)	
Competences Expected: AG-C05,AG-C08	
Topics	**Learning Outcomes**
• Types of research in computer science: empirical, theoretical, constructive, and evaluative. • The scientific method applied to research in computer science. • Main academic databases: IEEE Xplore, ACM Digital Library, Google Scholar, Semantic Scholar, and arXiv. • Bibliographic search strategies: Boolean operators, citation tracking, and systematic search. • Reference managers: Zotero, Mendeley, and BibTeX integration.	• Identify the types of research in computer science and their methodological differences. [Familiarizarse] • Use the main academic databases to find relevant literature on a given topic. [Usar] • Organize bibliographic references using reference managers and BibTeX format. [Usar]
Readings : [**ieee**], [**acm**]	

Unit 2: Critical Reading and Analysis of Scientific Literature (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Structure of a scientific article: abstract, introduction, related work, methodology, results, and conclusions. • Techniques for efficient reading of scientific articles (three-pass reading). • Critical evaluation: identifying contributions, limitations, and open problems. • Building a reading matrix and comparative tables of related works. • Classification and taxonomy design of existing approaches on a topic.	• Read and critically evaluate scientific articles identifying their contributions and limitations. [Usar] • Build an organized reading matrix that synthesizes multiple works on a given topic. [Usar] • Design a preliminary taxonomy of approaches for an assigned research area. [Evaluar]
Readings : [**ieee**], [**acm**], [**citeseer**]	

Unit 3: Research Ethics and Academic Integrity (6 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Academic integrity: plagiarism, self-plagiarism, data fabrication, and falsification. • Ethics in computer science research: privacy, algorithmic bias, and human subjects studies. • Open access, licenses (Creative Commons, GPL), and intellectual property in software. • Good citation and authorship attribution practices in collaborative work. • The peer review process: role of the reviewer and responsibilities of the author.	• Distinguish forms of academic misconduct and apply correct citation practices. [Usar] • Analyze ethical dilemmas in computational research applying integrity principles. [Evaluar] • Explain intellectual property rights and relevant licenses in computer science. [Familiarizarse]
Readings : [**ieee**], [**acm**]	

Unit 4: Technical Writing and Scientific Writing Tools (10 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• LaTeX structure: preamble, sections, equations, figures, tables, and cross-references. • Conference and journal templates: IEEE and ACM. • Academic writing style in computer science: clarity, precision, and active voice. • Writing each section of the article: abstract, introduction, related work, methodology, results, and conclusions. • Document version control with Git for LaTeXprojects. • Bibliography management with BibTeX and JabRef.	• Write a complete technical article in LaTeXusing IEEE or ACM templates. [Usar] • Apply academic writing conventions of the computer science community. [Usar] • Evaluate the writing quality of an article based on clarity and structure criteria. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 5: Systematic Literature Review and Survey Articles (10 hours)	
Competences Expected: AG-C04,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Types of reviews: narrative, systematic, mapping study, and meta-analysis. • Systematic literature review (SLR) protocol: research questions, search strategy, and selection criteria. • PRISMA guidelines: flowchart for selection of primary studies. • Synthesis techniques: narrative synthesis, vote counting, and thematic analysis. • Taxonomy design and classification schemes for a survey. • Structure and writing of a survey-type article.	• Design and execute a systematic literature review protocol for a computer science topic. [Usar] • Apply PRISMA guidelines to document and report the study selection process. [Usar] • Write a survey article with taxonomy, critical synthesis, and comparison of approaches. [Evaluar]
Readings : [**ieee**], [**acm**], [**citeseer**]	

Unit 6: Presentation and Dissemination of Research Results (6 hours)	
Competences Expected: AG-C02,AG-C04,AG-C05	
Topics	**Learning Outcomes**
• Oral presentation techniques for scientific audiences: structure, time management, and Q&A sessions. • Poster design for research conferences and symposia. • Responding to reviewers: addressing comments constructively and with evidence. • Research project planning: schedule, milestones, and deliverables. • Collaboration and co-authorship practices in scientific research.	• Prepare and deliver an oral presentation of research results to an academic audience. [Usar] • Develop a work plan for a research project with schedule and milestones. [Usar] • Respond constructively to reviewer comments in the publication process. [Evaluar]
Readings : [**ieee**], [**acm**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

MA202. Numerical Methods (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA202. Numerical Methods
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BMA103. Integral Calculus. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Numerical methods are a fundamental tool shared by computer science and engineering disciplines for approximating solutions to mathematical problems that cannot be solved analytically. This course provides an introduction to the most common numerical methods, including error analysis, solution of nonlinear equations, interpolation, numerical integration, and solution of differential equations, in scientific and engineering contexts.

5. GOALS

- Understand the importance of numerical methods in solving scientific and engineering problems.
- Apply different numerical methods to approximate solutions to mathematical problems.
- Analyze the accuracy and efficiency of the numerical methods used.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Error Analysis and Floating-Point Arithmetic (6 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• IEEE 754 floating-point representation: machine epsilon, overflow, underflow, and rounding modes • Rounding errors, catastrophic cancellation, and loss of significance • Forward and backward error analysis; condition number of a problem • Numerical stability: stable vs. unstable algorithms; Wilkinson's backward error analysis • Error propagation in arithmetic operations and function evaluation	• Explain the IEEE 754 floating-point standard and identify the sources of rounding and cancellation errors [Familiarity] • Estimate the condition number of a problem and predict the accuracy of a numerical result [Usage] • Analyze an algorithm for numerical stability using forward and backward error bounds [Assessment]
Readings : [**ChapraCanale2015**], [**BurdenFaires2010**]	

Unit 2: Root-Finding for Nonlinear Equations (10 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Bisection method: interval bracketing, stopping criteria, and error bound • Newton-Raphson method: derivation, quadratic convergence, and failure cases • Secant method as an alternative to Newton's method without an explicit derivative • Order of convergence and efficiency comparison among iterative methods • Extension of Newton's method to systems of nonlinear equations via the Jacobian matrix, the basis of solvers used in machine learning	• Compare the convergence guarantees of the bisection, Newton-Raphson, and secant methods [Familiarity] • Implement the Newton-Raphson method to approximate roots of a nonlinear function [Usage] • Analyze the order of convergence of an iterative method and justify its choice based on computational cost [Assessment]
Readings : [**ChapraCanale2015**], [**BurdenFaires2010**]	

Unit 3: Approximation Theory and Interpolation (10 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Polynomial interpolation: Lagrange, Newton divided differences, and Runge's phenomenon • Spline interpolation: cubic splines, B-splines, and piecewise polynomial methods • Best approximation in normed spaces: Chebyshev (minimax) and least squares • Chebyshev polynomials: properties, orthogonality, and spectral convergence • Trigonometric approximation and the fast Fourier transform (FFT)	• Explain Runge's phenomenon and justify the choice of Chebyshev nodes to mitigate it [Familiarity] • Construct cubic spline and Chebyshev interpolants for given data and estimate the interpolation error [Usage] • Apply the FFT to efficiently compute trigonometric approximations of a sampled function [Assessment]
Readings : [**ChapraCanale2015**], [**BurdenFaires2010**]	

Unit 4: Numerical Integration and Quadrature (10 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Newton-Cotes rules: trapezoidal, Simpson's, and composite rules; error analysis • Adaptive quadrature and automatic error control • Gaussian quadrature: optimal nodes and weights, orthogonal polynomial connection • Numerical treatment of improper and singular integrals • Monte Carlo integration and quasi-Monte Carlo methods for high-dimensional integrals	• Derive the error formula for composite Simpson's rule and identify its order of accuracy [Familiarity] • Select and apply an appropriate quadrature rule (Gaussian, adaptive) based on integrand regularity [Usage] • Apply Monte Carlo integration to estimate high-dimensional integrals and quantify the statistical error [Assessment]
Readings : [**ChapraCanale2015**], [**BurdenFaires2010**]	

Unit 5: Numerical Methods for Differential Equations (12 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Runge-Kutta methods: Euler, RK4, and embedded methods for error control • Linear multistep methods: Adams-Bashforth, Adams-Moulton, and BDF methods • Stability analysis: zero-stability, absolute stability regions, and stiff ODEs • Finite difference methods for parabolic and elliptic PDEs: stability and convergence	• Compare explicit and implicit ODE solvers and explain their relative stability for stiff problems [Familiarity] • Implement a Runge-Kutta method with adaptive step control and apply it to a system of ODEs [Usage] • Analyze the stability and convergence of a finite difference scheme for a parabolic PDE [Assessment]
Readings : [**ChapraCanale2015**], [**BurdenFaires2010**]	

Unit 6: Numerical Linear Algebra (12 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Gaussian elimination with partial pivoting, LU factorization, and complexity analysis • QR factorization via Householder and Gram-Schmidt; least squares problems • Eigenvalue algorithms: power iteration, QR algorithm, and the Lanczos method • Singular value decomposition: computation, truncation, and applications (PCA, pseudoinverse) • Iterative methods for large sparse systems: CG, GMRES, and preconditioning	• Compare direct and iterative solvers and select the appropriate method for a given matrix structure [Familiarity] • Apply QR factorization and SVD to solve least squares problems and compute low-rank approximations [Usage] • Analyze the numerical stability and computational cost of eigenvalue algorithms [Assessment]
Readings : [**BurdenFaires2010**], [**TrefethenBau1997**]	

Unit 7: Finite Element Method (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Weak (variational) formulation of boundary value problems and Sobolev spaces • Galerkin method: finite-dimensional approximation spaces and the stiffness matrix • Triangular and quadrilateral finite element spaces: Lagrange elements and conformity • A priori error estimates: Céa's lemma and interpolation error bounds • A posteriori error estimates and adaptive mesh refinement	• Derive the weak formulation of an elliptic BVP and show its equivalence to the strong form [Familiarity] • Assemble the global stiffness matrix and load vector for a linear finite element discretization [Usage] • Estimate the H^1 error of a finite element solution using Céa's lemma and interpolation theory [Assessment]
Readings : [**ReddyFEM2019**], [**BurdenFaires2010**]	

Unit 8: Optimization Algorithms (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Gradient descent and line search methods: Armijo-Wolfe conditions and convergence rates • Newton's method and quasi-Newton methods (BFGS, L-BFGS) • Constrained optimization: KKT conditions, penalty methods, and sequential quadratic programming • Convex optimization algorithms: interior-point methods and ADMM • Stochastic gradient descent (SGD), variance reduction, and Adam optimizer	• Explain the convergence guarantees of gradient descent for smooth convex functions [Familiarity] • Apply Newton and quasi-Newton methods to unconstrained optimization problems [Usage] • Formulate a constrained optimization problem as a KKT system and apply an interior-point method [Assessment]
Readings : [**NocedalWright2006**]	

Unit 9: Parallel and High-Performance Computing (4 hours)	
Competences Expected: AG-C11	
Topics	**Learning Outcomes**
• Parallel architectures: shared memory (OpenMP), distributed memory (MPI), and GPU (CUDA) • Performance models: Amdahl's law, roofline model, and communication complexity • Parallel algorithms for dense and sparse linear algebra (ScaLAPACK, PETSc) • Domain decomposition and parallel PDE solvers • Automatic differentiation (forward and reverse mode) and its role in scientific ML	• Explain Amdahl's law and identify the bottlenecks limiting parallel speedup in a given algorithm [Familiarity] • Implement a parallel linear algebra routine using MPI or OpenMP and measure parallel efficiency [Usage] • Apply automatic differentiation to compute gradients of complex scientific functions for optimization [Assessment]
Readings : [**PachecoMalensek2022**]	

Unit 10: Graph Theory (4 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Graphs: definitions, isomorphism, degree sequences, trees, and spanning trees • Connectivity, Menger's theorem, and network flows (max-flow min-cut) • Matchings (Hall's theorem), graph colorings, and the chromatic polynomial • Planar graphs, Euler's formula, Kuratowski's theorem, and the four-color theorem • Adjacency and Laplacian matrices, eigenvalues, and expander graphs	• Identify structural properties of graphs (connectivity, planarity, bipartiteness) and apply Euler's formula [Familiarity] • Apply Hall's marriage theorem and network flow algorithms to matching and routing problems [Usage] • Analyze a graph's spectrum to bound its chromatic number and connectivity properties [Assessment]
Readings : [**West2001**]	

Unit 11: Time Series Analysis (2 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Stationarity, autocovariance function, and the autocorrelation function (ACF) • ARMA models: identification, estimation (Yule-Walker, MLE), and diagnostics • Spectral density, the periodogram, and Wiener-Khinchin theorem • State-space models and the Kalman filter • ARIMA models, seasonal adjustment, and multi-step forecasting	• Identify stationarity and determine the order of an ARMA model from ACF and PACF plots [Familiarity] • Fit ARIMA models to time series data, validate residuals, and produce forecasts [Usage] • Apply the Kalman filter to estimate hidden states in a linear Gaussian state-space model [Assessment]
Readings : [**ShumwayStoffer2017**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BEI202. Technical and professional English IV (Elective)

2. GENERAL INFORMATION

2.1 Course	:	BEI202. Technical and professional English IV
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BEI201. Technical and professional English III. (3^{rd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

At the intermediate level, students apply English to the full range of technical communication required in computing and engineering practice: reading research papers, writing structured technical documents, analyzing case studies, and engaging in professional dialogue. Grammar structures are taught through authentic computing and engineering contexts, reinforcing both linguistic competence and disciplinary knowledge.

5. GOALS

- Develop the ability to write and read structured technical documents in English.
- Bring the student to a more fluent and natural expression in technical and scientific English.
- Master vocabulary for computing, systems engineering, and applied sciences.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Best Practices and Ethics (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Modals of obligation and advice: Should, Must, Have to in engineering codes. • Prohibition and absence of obligation in software licensing and data privacy. • Vocabulary: Professional ethics, intellectual property, responsible AI. • Writing: Formal technical recommendation and policy documents.	• By the end of the unit, students correctly use modal auxiliaries to express engineering standards, ethical obligations, and professional recommendations in formal technical environments.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 2: Failures and Root Causes (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Past Perfect: Structure and use in root cause analysis and post-mortems. • Contrast between Simple Past, Continuous, and Perfect in failure timelines. • Vocabulary: System outages, security breaches, cascading failures. • Expressions for reporting information: reported speech in incident briefings.	• Students reconstruct complex failure timelines and root cause analyses by establishing clear chronological sequences through the use of past tense contrasts.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 3: If-Then Logic (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Conditionals Type 0, 1, and 2 applied to computing logic and engineering decisions. • Sentences with "If" and "Unless" for system constraints and requirements. • Vocabulary: Hypothetical scenarios, risk analysis, and optimization trade-offs. • Expressions of what-if design thinking: "If we used X, the system would..."	• Students pose real and hypothetical technical scenarios and their consequences, using conditional structures to reason about design choices and engineering decisions.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 4: Ongoing Research (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Perfect Continuous for describing research in progress. • Contrast between Simple and Continuous Present Perfect in scientific writing. • Vocabulary: Continuous integration, agile development, iterative research. • Word formation: Prefixes and suffixes in scientific and technical terminology.	• Students describe research activities and development processes that began in the past and continue into the present, emphasizing duration and iterative progress.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 5: Manufacturing and Deployment (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Passive Voice in all basic tenses for technical process descriptions. • Verbs with two objects in passive: data is sent to the server by the client. • Vocabulary: Hardware manufacturing, cloud deployment, CI/CD pipelines. • Writing: Technical description of a deployment or production process.	• Students use the passive voice across multiple tenses to describe technical processes where the action, system, or result is the focus rather than the agent.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 6: Tech Roadmaps (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Future Continuous and Future Perfect for project timelines and milestones. • Adverbs of probability: perhaps, probably, definitely — in tech forecasting. • Vocabulary: Product roadmaps, release planning, emerging research areas. • Oral presentations: "The state of AI / quantum computing in 5 years".	• Students project technical states, completed future milestones, and research outcomes, discussing computing and engineering roadmaps with advanced vocabulary.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 7: Technical Teams (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Indirect questions and Tag questions in technical interviews and peer reviews. • Common Phrasal Verbs in engineering and research environments. • Vocabulary: Collaboration tools, code review practices, team roles. • Writing: Short technical blog posts and engineering team updates.	• Students use polite inquiry forms and confirmation strategies in technical interviews and team discussions, improving their fluency and naturalness in professional engineering contexts.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

Unit 8: Technical English IV: Networks and Security (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Networking terminology: protocols, topologies, OSI model, routing. • Cybersecurity vocabulary: vulnerabilities, threats, encryption, authentication. • Reading and writing network incident and security audit reports. • Project: Security assessment plan for a computing infrastructure.	• By the end of the course, students command networking and cybersecurity technical terminology to read specialized documentation and communicate effectively with IT and security professionals in English.
Readings : [**Soars022S**], [**Cambridge06**], [**MacGrew99**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS212. Analysis and Design of Algorithms (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS212. Analysis and Design of Algorithms
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	• CS210. Algorithms and Data Structures. (4^{th} Sem) • CS211. Theory of Computation. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Algorithms are the heart of computer science. This course provides the mathematical and analytical tools necessary to evaluate the efficiency of algorithms in terms of time and space. Students will explore fundamental data structures, complexity analysis, and various design paradigms—such as brute force, greedy, and dynamic programming. Understanding these techniques is essential for building software that is not only functional but also scalable and resource-efficient.

5. GOALS

- Master the use of asymptotic notation to analyze algorithm performance.
- Apply foundational data structures to solve computational problems.
- Use algorithmic strategies like Greedy, Divide and Conquer, and Dynamic Programming.
- Analyze the complexity of algorithms in best, average, and worst cases.
- Develop the ability to select the most appropriate algorithm for a given problem.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Foundational Data Structures (6 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Abstract Data Type (ADT) and operations on an ADT: – Dictionary operations (insert, delete, find) • Arrays: – Numeric vs non-numeric, character strings – Single (vector) vs multidimensional (matrix) • Records/Structs/Tuples and Objects • Linked lists (for historical reasons): – Single vs Double and Linear vs Circular • Stacks • Queues and deques: – Heap-based priority queue • Hash tables/maps: – Collision resolution and complexity (e.g., probing, chaining, rehash) • Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Sets	• For each ADT/Data-Structure in this unit: – Explain its definition, properties, representation(s), and associated ADT operations. – Explain step-by-step how the ADT operations associated with the data structure transform it. [Explain] • Explain how collision avoidance and collision resolution is handled in hash tables [Explain] • Explain the heap property and the use of heaps as an implementation of a priority queue [Explain]

Readings : [**Cormen2022**], [**Kleinberg2005**]

Unit 2: Foundational Algorithms (6 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Graphs (e.g., [un]directed, [a]cyclic, [un]connected, and [un]weighted): – Graph representation: adjacency list vs matrix • Search algorithms: – $O(n)$ complexity (e.g., linear/sequential array/list search) – $O(\log_2 n)$ complexity (e.g., binary search) – $O(\log_b n)$ complexity (e.g., uninformed depth/breadth-first tree search) • Sorting algorithms (e.g., stable, unstable): – $O(n^2)$ complexity (e.g., insertion, selection) – $O(n \log n)$ complexity (e.g., quicksort, merge, timsort) • Graph algorithms: – Shortest path (e.g., Dijkstra's, Floyd's) – Minimal spanning tree (e.g., Prim's, Kruskal's) • Sorting algorithms: – $O(n \log n)$ complexity heapsort – Pseudo $O(n)$ complexity (e.g., bucket, counting, radix) • Graph algorithms: – Transitive closure (e.g., Warshall's) – Topological sort • Matching: – Efficient string matching (e.g., Boyer-Moore, Knuth-Morris-Pratt) – Longest common subsequence matching – Regular expression matching	• For each algorithm in this unit explain step-by-step how the algorithm operates [Explain] • For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply] • Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create] • Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain] • For each of the algorithms and algorithmic approaches in the KA Core topics: – Explain a prototypical example of the algorithm, and – Explain step-by-step how the algorithm operates. [Explain]

Readings : [**Cormen2022**], [**Kleinberg2005**]

Unit 3: Advanced Algorithms (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Cryptography algorithms (e.g., SHA-256) • Parallel algorithms • Consensus algorithms (e.g., Blockchain): – Proof of work vs proof of stake • Quantum computing algorithms: – Oracle-based (e.g., Deutsch-Jozsa, Bernstein-Vazirani, Simon) – Superpolynomial speed-up via QFT (e.g., Shor's) – Polynomial speed-up via amplitude amplification (e.g., Grover's) • Fast-Fourier Transform (FFT) algorithm • Differential evolution algorithm	• An appreciation of quantum computation and its application to certain problems [Explain]
Readings : [**Cormen2022**], [**Kleinberg2005**]	

Unit 4: Algorithmic Strategies (24 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Paradigms: – Brute-Force (e.g., linear search, selection sort, traveling salesperson, knapsack) – Decrease-and-Conquer: ∗ By a Constant (e.g., insertion sort, topological sort) ∗ By a Constant Factor (e.g., binary search) ∗ By a Variable Size (e.g., Euclid's) – Divide-and-Conquer (e.g., binary search, quicksort, mergesort, Strassen's) – Greedy (e.g., Dijkstra's, Kruskal's, Knapsack) – Transform-and-Conquer: ∗ Instance simplification (e.g., find duplicates via list presort) ∗ Representation change (e.g., heapsort) ∗ Problem reduction (e.g., least-common-multiple, linear programming) ∗ Dynamic programming (e.g., Floyd's, Marshall, Bellman-Ford) – Space vs time tradeoffs (e.g., hashing) • Handling exponential growth (e.g., heuristic A*, branch-and-bound, backtracking) • Iteration vs recursion (e.g., factorial, tree search) • Paradigms: – Approximation algorithms – Iterative improvement (e.g., Ford-Fulkerson, simplex) – Randomized/Stochastic algorithms (e.g., max-cut, balls and bins) • Quantum computing	• For each of the paradigms in this unit: – Explain its definitional characteristics – Explain an example that demonstrates the paradigm including how this example satisfies the paradigm's characteristics. [Explain] • For each of the algorithms in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain the paradigm used by the algorithm and how it exemplifies this paradigm [Explain] • Given an algorithm, explain the paradigm used by the algorithm and how it exemplifies this paradigm [Explain] • Give a real-world problem, evaluate appropriate algorithmic paradigms and algorithms from these paradigms that address the problem including evaluating the tradeoffs among the paradigms and algorithms selected [Evaluate] • Give examples of iterative and recursive algorithms that solve the same problem, explain the benefits and disadvantages of each approach [Explain] • Evaluate whether a greedy approach leads to an optimal solution [Evaluate] • Explain various approaches for addressing computational problems whose algorithmic solutions are exponential [Explain]
Readings : [**Cormen2022**], [**Dasgupta2006**]	

Unit 5: Complexity Analysis Framework (6 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Complexity Analysis Framework: – Best, average, and worst-case performance of an algorithm – Empirical and relative (Order of Growth) measurements – Input size and primitive operations – Time and space efficiency • Empirical measurements of performance • Time and space tradeoffs in algorithms	• For each algorithm in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain its runtime complexity class and why it belongs to this class [Explain] • Develop empirical studies to determine and validate hypotheses about the runtime complexity of various algorithms by running algorithms on input of various sizes and comparing actual performance to the theoretical analysis [Create] • Explain examples that illustrate time-space tradeoffs of algorithms [Explain] • Explain how tree balance affects the efficiency of binary search tree operations [Explain]

Readings : [**Cormen2022**], [**Sipser2012**]

Unit 6: Asymptotic Notation and Complexity Classes (6 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Asymptotic complexity analysis (average and worst-case bounds): – Big-O, Big-Omega, and Big-Theta formal notations – Foundational Complexity Classes and Representative Examples/Problems: ∗ $O(1)$ Constant (e.g., array access) ∗ $O(\log_2 n)$ Logarithmic (e.g., binary search) ∗ $O(n)$ Linear (e.g., linear search) ∗ $O(n \log_2 n)$ Log Linear (e.g., mergesort) ∗ $O(n^2)$ Quadratic (e.g., selection sort) ∗ $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) ∗ $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) ∗ $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)	• For each foundational complexity class in this unit, explain an algorithm that demonstrates the associated runtime complexity [Explain] • Apply Big-O notation to give upper case bounds on time/space complexity of algorithms [Apply]

Readings : [**Cormen2022**], [**Sipser2012**]

Unit 7: Complexity Analysis: Recursion, Amortization, and Tight Bounds (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Little-o, Little-Omega, and Little Theta notations • Formal recursive analysis • Amortized analysis	• Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior [Evaluate] • Use recurrence relations to evaluate the time complexity of recursively defined algorithms [Apply] • Apply elementary recurrence relations using a form of the Master Theorem [Apply]
Readings : [**Cormen2022**], [**Sipser2012**]	

Unit 8: Computational Complexity Theory (6 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Tractability and intractability: – P, NP, and NP-Complete Complexity Classes – NP-Complete Problems (e.g., SAT, Knapsack, TSP) – Reductions	• Explain the significance of NP-Completeness [Explain] • Explain examples of NP-complete problems [Explain] • Explain the classes P and NP [Explain]
Readings : [**Sipser2012**], [**Cormen2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS261. Artificial Intelligence (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS261. Artificial Intelligence
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	STA251. Probability Calculation. (3^{rd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Artificial Intelligence (AI) is a fundamental field of computer science that seeks to create systems capable of performing tasks that typically require human intelligence. This course provides an introduction to the techniques used to build such systems, including search algorithms, probabilistic reasoning, and machine learning. Students will learn to model complex problems and implement intelligent agents, while considering the social and ethical implications of these technologies.

5. GOALS

- Understand the fundamental concepts and history of Artificial Intelligence.
- Implement and evaluate search algorithms for problem-solving.
- Master the basics of machine learning, including supervised and unsupervised learning.
- Apply probabilistic reasoning to handle uncertainty in intelligent systems.
- Analyze the ethical challenges and social impact of AI.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Introduction to AI: Concepts and Intelligent Behavior (4 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Overview of AI problems, Examples of successful recent AI applications • Definitions of agents with examples (e.g., reactive, deliberative) • What is intelligent behavior?: – The Turing test and its flaws – Multimodal input and output – Simulation of intelligent behavior – Rational versus non-rational reasoning • Overview of AI Applications, growth, and impact (economic, societal, ethics) • Philosophical issues • History of AI	• Describe the Turing test and the "Chinese Room" thought experiment [Explain] • Differentiate between optimal reasoning/behavior and human-like reasoning/behavior [Evaluate] • Differentiate the terms: AI, machine learning, and deep learning [Explain]
Readings : [**Russell2020**], [**Luger2008**]	

Unit 2: Problem Characteristics and Agent Nature (4 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Problem characteristics: – Fully versus partially observable – Single versus multi-agent – Deterministic versus stochastic – Static versus dynamic – Discrete versus continuous • Nature of agents: – Autonomous, semi-autonomous, mixed-initiative autonomy – Reflexive, goal-based, and utility-based – Decision making under uncertainty and with incomplete information – The importance of perception and environmental interactions – Learning-based agents – Embodied agents: ∗ sensors, dynamics, effectors • Practice identifying problem characteristics in example environments • Additional depth on nature of agents with examples • Additional depth on AI Applications, Growth, and Impact (economic, societal, ethics, security)	• Enumerate the characteristics of a specific problem [Apply]
Readings : [**Russell2020**], [**Luger2008**]	

Unit 3: State and Uninformed Search (5 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• State space representation of a problem: – Specifying states, goals, and operators – Factoring states into representations (hypothesis spaces) – Problem solving by graph search: ∗ e.g., Graphs as a space, and tree traversals as exploration of that space ∗ Dynamic construction of the graph (not given upfront) • Uninformed graph search for problem solving: – Breadth-first search – Depth-first search: ∗ With iterative deepening – Uniform cost search • Space and time complexities of graph search algorithms • Bidirectional search • Understanding the search space: – Constructing search trees – Dynamic search spaces – Combinatorial explosion of search space – Search space topology (e.g., ridges, saddle points, local minima)	• Design the state space representation for a puzzle (e.g., N-queens or 3-jug problem) [Apply] • Select and implement an appropriate uninformed search algorithm for a problem (e.g., tic-tac-toe), and characterize its time and space complexities [Apply]
Readings : [**Russell2020**], [**Luger2008**]	

Unit 4: Heuristic Search and CSP (6 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Heuristic graph search for problem solving: – Heuristic construction and admissibility – Hill-climbing – Local minima and the search landscape: ∗ Local vs global solutions – Greedy best-first search – A* search • Beam search • Implementation of A* search • Constraint satisfaction • Variations on A* (IDA*, SMA*, RBFS)	• Select and implement an appropriate informed search algorithm for a problem after designing a helpful heuristic function (e.g., a robot navigating a 2D gridworld) [Apply] • Evaluate whether a heuristic for a given problem is admissible/can guarantee an optimal solution [Evaluate] • Design and implement A*/beam search to solve a problem, and compare it against other search algorithms in terms of the solution cost, number of nodes expanded, etc [Apply] • Compare and contrast various heuristic searches vis-a-vis applicability to a given problem [Evaluate] • Model a logic or Sudoku puzzle as a constraint satisfaction problem, solve it with backtrack search, and determine how much arc consistency can reduce the search space [Apply]
Readings : [**Russell2020**], [**Luger2008**]	

Unit 5: Adversarial Search and Stochastic Optimization (5 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Two-player adversarial games: – Minimax search – Alpha-beta pruning: ∗ Ply cutoff • Local search • Tabu search • Two-player adversarial games: – The horizon effect – Opening playbooks/endgame solutions – What it means to ”solve” a game (e.g., checkers) • Implementation of minimax search, beam search • Expectimax search (MDP-solving) and chance nodes • Stochastic search: – Simulated annealing – Genetic algorithms – Monte-Carlo tree search	• Apply minimax search in a two-player adversarial game (e.g., connect four), using heuristic evaluation at a particular depth to compute the scores to back up [Apply] • Apply minimax search with alpha-beta pruning to prune search space in a two-player adversarial game (e.g., connect four) [Apply] • Compare and contrast genetic algorithms with classic search techniques, explaining when it is most appropriate to use a genetic algorithm to learn a model versus other forms of optimization (e.g., gradient descent) [Evaluate] • Design and implement a genetic algorithm solution to a problem [Apply] • Design and implement a simulated annealing schedule to avoid local minima in a problem [Apply]
Readings : [**Russell2020**], [**Luger2008**]	

Unit 6: Machine Learning Fundamentals (7 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Definition and examples of a broad variety of machine learning tasks: – Supervised learning: ∗ Classification ∗ Regression – Reinforcement learning – Unsupervised learning: ∗ Clustering • Fundamental ideas: – No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. – Sources of error and undecidability in machine learning • A simple statistical-based supervised learning such as linear regression or decision trees: – Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly • The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): – The bias (underfitting) - variance (overfitting) tradeoff	• Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] • Differentiate the terms of AI, machine learning, and deep learning [Evaluate] • Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] • Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] • Explain how machine learning works as an optimization/search process [Explain]
Readings : [**Russell2020**], [**Alpaydin2020**]	

Unit 7: Machine Learning Data, Representations and Evaluation (7 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Working with Data: – Data preprocessing: ∗ Importance and pitfalls of preprocessing choices – Handling missing values (imputing, flag-as-missing): ∗ Implications of imputing vs flag-as-missing – Encoding categorical variables, encoding real-valued data – Normalization/standardization – Emphasis on real data, not textbook examples • Representations: – Hypothesis spaces and complexity – Simple basis feature expansion, such as squaring univariate features – Learned feature representations • Machine learning evaluation: – Separation of train, validation, and test sets – Performance metrics for classifiers – Estimation of test performance on held-out data – Tuning the parameters of a machine learning model with a validation set – Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions	• Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] • Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] • Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] • Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]
Readings : [**Russell2020**], [**Alpaydin2020**]	

Unit 8: Neural Networks and Ethics (6 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Basic neural networks: – Fundamentals of understanding how neural networks work and their training process, without details of the calculations – Basic introduction to generative neural networks (e.g., large language models) • Ethics for Machine Learning: – Focus on real data, real scenarios, and case studies – Dataset/algorithmic/evaluation bias and unintended consequences	• Describe the neural network training process and resulting learned representations [Explain]
Readings : [**Russell2020**], [**Alpaydin2020**]	

Unit 9: Machine Learning Formulation and Evaluation (3 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: – Objective function – Gradient descent – Regularization to avoid overfitting (mathematical formulation) • Ensembles of models: – Simple weighted majority combination • Performance evaluation: – Other metrics for classification (e.g., error, precision, recall) – Performance metrics for regressors – Confusion matrix – Cross-validation: ∗ Parameter tuning (grid/random search, via cross-validation)	• Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare] • Evaluate the performance of a simple learning system on a real-world dataset [Evaluate] • Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain] • Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain] • Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]
Readings : [**Goodfellow2016**]	

Unit 10: Deep Learning, RL and Ethics (3 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Deep learning: – Deep feed-forward networks (intuition only, no mathematics) – Convolutional neural networks (intuition only, no mathematics) – Visualization of learned feature representations from deep nets – Other architectures (generative NN, recurrent NN, transformers, etc.) • Overview of reinforcement learning methods • Two or more applications of machine learning algorithms: – E.g., medicine and health, economics, vision, natural language, robotics, game play • Ethics for Machine Learning: – Continued focus on real data, real scenarios, and case studies – Privacy – Fairness – Intellectual property – Explainability	• Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] • Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]
Readings : [**Goodfellow2016**]	

Unit 11: Knowledge Representation and Probabilistic Reasoning Foundations (6 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Types of representations: – Symbolic, logical: ∗ Creating a representation from a natural language problem statement – Learned subsymbolic representations – Graphical models (e.g., naive Bayes, Bayesian network) • Review of probabilistic reasoning, Bayes theorem • Bayesian reasoning: – Bayesian inference • Random variables and probability distributions: – Axioms of probability – Probabilistic inference – Bayes' Rule (derivation) – Bayesian inference (more complex examples)	• Given a natural language problem statement, encode it as a symbolic or logical representation [Apply] • Compute a probabilistic inference in a real-world problem using Bayes' theorem to determine the probability of a hypothesis given evidence [Apply] • Apply Bayes' rule to determine the probability of a hypothesis given evidence [Apply] • Compute the probability of outcomes and test whether outcomes are independent [Apply]
Readings : [**Russell2020**]	

Unit 12: Independence, Markov Models and Decision Making (6 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Independence • Conditional Independence • Markov chains and Markov models • Utility and decision making	• Explain how we can make decisions under uncertainty, using concepts such as Bayes theorem and utility [Explain]
Readings : [**Russell2020**]	

Unit 13: AI Applications and Societal Impact (4 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• At least one application of AI to a specific problem and field, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose at least one for the CS Core): – Formulating and evaluating a specific application as an AI problem: ∗ How to deal with underspecified or ill-posed problems – Data availability/scarcity and cleanliness: ∗ Basic data cleaning and preprocessing ∗ Data set bias – Algorithmic bias – Evaluation bias – Assessment of societal implications of the application • Deployed deep generative models: – High-level overview of deep image generative models (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion, etc.), their uses, and their shortcomings/pitfalls. – High-level overview of large language models (e.g., as of 2023, ChatGPT, Bard, etc.), their uses, and their shortcomings/pitfalls. • Overview of societal impact of AI: – Ethics – Fairness – Trust/explainability – Privacy and usage of training data – Human autonomy and oversight/regulations/legal requirements – Sustainability	• Given a real-world application domain and problem, formulate an AI solution to it, identifying proper data/input, preprocessing, representations, AI techniques, and evaluation metrics/methodology [Apply]
Readings : [**Russell2020**], [**Alpaydin2020**]	

Unit 14: Additional AI Applications and Societal Impact (4 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• One or more additional applications of AI to a broad set of problems and diverse fields, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose a different area from that chosen for the CS Core): – Formulating and evaluating a specific application as an AI problem: ∗ How to deal with underspecified or ill-posed problems – Data availability/scarcity and cleanliness: ∗ Basic data cleaning and preprocessing ∗ Data set bias – Algorithmic bias – Evaluation bias – Assessment of societal implications of the application • Additional depth on deployed deep generative models: – Introduction to how deep image generative models work, (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion) including discussion of attention – Introduction to how large language models work, (e.g., as of 2023, ChatGPT, Bard) including discussion of attention – Idea of foundational models, how to use them, and the benefits/issues with training them from big data • Analysis and discussion of the societal impact of AI: – Ethics – Fairness – Trust/explainability – Privacy and usage of training data – Human autonomy and oversight/regulations/legal requirements – Sustainability	• Analyze the societal impact of one or more specific real-world AI applications, identifying issues regarding ethics, fairness, bias, trust, and explainability [Evaluate] • Describe some of the failure modes of current deep generative models for language or images, and how this could affect their use in an application [Describe]
Readings : [**Russell2020**], [**Alpaydin2020**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS272. Databases II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS272. Databases II
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS271. Data Management. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course is a continuation of Data Management I, focusing on the internal mechanisms of database management systems (DBMS) and modern database technologies. It covers query optimization, indexing structures, and transactional integrity in depth. Additionally, it introduces students to non-relational models (NoSQL) and distributed architectures, which are essential for handling large-scale data in contemporary cloud environments.

5. GOALS

- Understand the internal storage structures and indexing techniques.
- Analyze and optimize query execution plans.
- Comprehend advanced transaction protocols and recovery mechanisms.
- Evaluate and implement solutions using NoSQL database models.
- Understand the principles of distributed and parallel databases.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Query Processing (12 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Page structures • Index structures: – B+ trees – Hash indices: static and dynamic – Index creation in SQL • File structures: – Heap files – Hash files • Algorithms for query operators: – External Sorting – Selection – Projection; with and without duplicate elimination – Natural Joins: Nested loop, Sort-merge, Hash join – Analysis of algorithm efficiency • Query transformations • Query optimization: – Access paths – Query plan construction – Selectivity estimation – Index-only plans • Parallel Query Processing (e.g., parallel scan, parallel join, parallel aggregation) • Database tuning/performance: – Index selection – Impact of indices on query performance – Denormalization	• Describe the purpose and organization of both B+ tree and hash index structures [Describe] • Compose an SQL command to create an index (any kind) [Compose] • Specify the steps for the various query operator algorithms: external sorting, projection with duplicate elimination, sort-merge join, hash-join, block nested-loop join [Analyze] • Derive the run-time (in I/O requests) for each of the above algorithms [Analyze] • Transform a query in relational algebra to its equivalent appropriate for a left-deep, pipelined execution [Redesign] • Compute selectivity estimates for a given selection and/or join operation [Compute] • Describe how to modify an index structure to facilitate an index-only operation for a given relation [Describe] • For a given scenario decide on which indices to support for the efficient execution of a set of queries [Assess] • Describe how DBMSs leverage parallelism to speed up query processing by dividing the work across multiple processors or nodes [Describe]
Readings : [**Silberschatz2019**], [**Ramakrishnan2003**]	

Unit 2: DBMS Internals (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• DB Buffer Management • Transaction Management: – Isolation Levels – ACID – Serializability – Distributed Transactions • Concurrency Control: – 2-Phase Locking – Deadlocks handling strategies – Quorum-based consistency models • Recovery Manager: – Relation with Buffer Manager • Concurrency Control: – Optimistic concurrency control – Timestamp concurrency control • Recovery Manager: – Write-Ahead logging – ARIES recovery system (Analysis, REDO, UNDO)	• Describe how a DBMS manages its Buffer Pool [Describe] • Describe the four properties for a correct transaction manager [Describe] • Outline the principle of serializability [Outline]
Readings : [**Silberschatz2019**], [**GarciaMolina2008**]	

Unit 3: Distributed Databases/Cloud Computing (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Distributed DBMS: – Distributed data storage – Distributed query processing – Distributed transaction model – Homogeneous and heterogeneous solutions – Client-server distributed databases • Parallel DBMS: – Parallel DBMS architectures: shared memory, shared disk, shared nothing – Speedup and scale-up, e.g., use of the MapReduce processing model – Data replication and weak consistency models	• Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe] • Analyze the tradeoffs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze] • Describe data replication strategies and weak consistency models in distributed database systems [Describe]
Readings : [**Silberschatz2019**], [**Ozsu2020**]	

Unit 4: NoSQL Systems (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Why NoSQL? (e.g., Impedance mismatch between Application [CRUD] and RDBMS) • Key-Value and Document data model • Storage systems (e.g., Key-Value systems, Data Lakes) • Distribution Models (Sharding and Replication) • Graph Databases • Consistency Models (Update and Read, Quorum consistency, CAP theorem) • Processing model (e.g., Map-Reduce, multi-stage map-reduce, incremental map-reduce) • Case Studies: Cloud storage system (e.g., S3); Graph databases; "When not to use NoSQL"	• Develop a use case for the use of NoSQL over RDBMS [Create] • Describe the defining characteristics behind Key-Value and Document-based data models [Describe]
Readings : [**Sadalage2012**], [**kleppmann2017designing**]	

Unit 5: Semi-structured and Unstructured Databases (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Vectorized unstructured data (text, video, audio, etc.) and vector storage: – TF-IDF Vectorizer with ngram – Word2Vec – Array database or array data type handling • Semi-structured databases (e.g., JSON): – Storage: ∗ Encoding and compression of nested data types – Indexing: ∗ Btree, skip index, Bloom filter ∗ Inverted index and bitmap compression ∗ Space filling curve indexing for semi-structured geo-data – Query processing for OLTP and OLAP use cases: ∗ Insert, Select, update/delete tradeoffs ∗ Case studies on Postgres/JSON, MongoDB, and Snowflake/JSON	• Describe vectorized representations of unstructured data (e.g., TF-IDF, Word2Vec) and their storage models [Describe] • Apply indexing techniques (e.g., B-tree, inverted index, Bloom filter) to semi-structured data stored in JSON format [Apply] • Analyze query processing tradeoffs for semi-structured databases in OLTP and OLAP use cases [Analyze]
Readings : [**kleppmann2017designing**], [**manning2008introduction**], [**mikolov2013efficient**], [**agrawal2023toward**], [**mongodb_documentation**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS291. Software Engineering I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS291. Software Engineering I
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	• CS113. Objects-oriented programming II. (3^{rd} Sem) • CS271. Data Management. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Software engineering is a collaborative discipline that applies structured processes to build high-quality systems. This first course focuses on the foundational aspects of the software lifecycle, from initial requirements and human-centric design to systematic testing. By integrating teamwork principles and rigorous development fundamentals, students will learn to create software that is both technically sound and user-oriented.

5. GOALS

- Understand and apply various software process models and configuration management.
- Master requirements elicitation and modeling using standardized notations.
- Develop effective teamwork and collaboration skills for software development.
- Design robust software architectures based on engineering principles.
- Implement systematic testing strategies and understand software quality.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Version Control and CI/CD (12 hours)

Competences Expected: AG-C03,AG-C09

Topics	Learning Outcomes
• Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant. • Release management. • Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security • Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Collaborative review workflow as a quality gate – Mandatory pull-request review before merging – Code ownership via CODEOWNERS-style rules – Conventional commit message conventions – Branch-protection rules enforcing checks before merge • Dependency and supply-chain vulnerability scanning integrated into CI pipelines	• Describe the difference between centralized and distributed software configuration management [Describe] • Describe how version control can be used to help manage software release management [Describe] • Identify configuration items and use a source code control tool in a small team-based project [Analyze] • Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] • Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] • Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] • Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Readings : [**Sommerville2015**], [**Pressman2019**]

Unit 2: Requirements Engineering (14 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Describe functional requirements using, for example, use cases or user stories. – Using at least one method of documenting and structuring functional requirements. – Understanding how the method supports design and implementation. – Strengths and weaknesses of using a specific approach. • Properties of requirements including consistency, validity, completeness, and feasibility. • Requirements elicitation. – Sources of requirements, for example, users, administrators, or support personnel. – Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis. • Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context • Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics • Communicating and/or formalizing requirement specifications.	• Compare different methods of eliciting requirements along multiple axes [Compare] • Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze] • Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze] • Collect a set of requirements for a simple software system [Analyze] • Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze] • Identify the functional and non-functional requirements in a set of requirements [Analyze]
Readings : [**Sommerville2015**], [**Wiegers2013**]	

Unit 3: Team Collaboration and Dynamics (14 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture • Common causes of team conflict, and approaches for conflict resolution. • Cooperative programming: – Pair programming or Swarming – Code review – Collaboration through version control • Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics – Advantages of teamwork – Risks and complexity of such collaboration • Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule	• Follow effective team communication practices [Analyze] • Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate] • Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze] • Collaborate effectively in cooperative development/programming [Analyze] • Propose and delegate necessary roles and responsibilities in a software development team [Propose] • Compose and follow an agenda for a team meeting [Compose] • Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]
Readings : [**Wigdor2011**], [**Johnson2013**]	

Unit 4: Principles and Software Architecture (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• System design principles. System Reliability – Levels of abstraction (e.g., architectural design and detailed design) – Separation of concerns – Information hiding – Coupling and cohesion • Software architecture. System Reliability – Design paradigms ∗ Top-down functional decomposition/layered design ∗ Data-oriented architecture ∗ Object-oriented analysis and design ∗ Event-driven design – Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) – Identifying component boundaries and dependencies • Programming in the large vs programming in the small. System Reliability • Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering	• Identify the standard software architecture of a given high-level design [Analyze] • Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] • Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] • Identify the dependencies among a set of software components in an architectural design [Analyze]
Readings : [**Gamma1994**], [**Pressman2019**]	

Unit 5: Verification and validation Concepts (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation concepts – Verification: Are we building the thing right? – Validation: Did we build the right thing? • Why testing matters: Does the component remain functional as the code evolves? • Testing objectives – Usability – Reliability – Conformance to specification – Performance – Security	• Explain why testing is important [Explain] • Distinguish between program validation and verification [Distinguish] • Describe different objectives of testing [Describe]
Readings : [**Sommerville2015**], [**Pressman2019**]	

Unit 6: Testing Tools and Analysis (4 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation tooling and automation – Static analysis – Code coverage – Fuzzing – Dynamic analysis and fault containment (sanitizers, etc.) – Fault logging and fault tracking • Test planning and generation – Fault estimation and testing termination including defect seeding – Use of random and pseudo random numbers in testing • Testing asynchronous, parallel, and concurrent systems • Verification and validation of non-code artifacts (documentation, training materials) • Behavior-driven development (BDD) – Expressing acceptance criteria as Gherkin Given/When/Then scenarios – Executing BDD scenarios as automated end-to-end tests • Mutation testing and static code-quality analysis – Mutation testing to assess test-suite effectiveness – Static detection of code smells and excessive cyclomatic complexity	• Describe and compare different tools for verification and validation [Describe] • Automate the use of different tools in a small software project [Design] • Explain how and when random numbers should be used in testing [Explain] • Describe approaches for fault estimation [Describe] • Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] • Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] • Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] • Describe techniques for the verification and validation of non-code artifacts [Describe] • Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] • Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]
Readings : [**Sommerville2015**], [**Pressman2019**]	

Unit 7: Performance Testing and Benchmarking (4 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Performance testing and benchmarking – Throughput and latency – Degradation under load (stress testing, FIFO vs LIFO handling of requests) – Speedup and scaling ∗ Amdahl's law ∗ Gustafson's law ∗ Soft and weak scaling – Identifying and measuring figures of merits – Common performance bottlenecks ∗ Compute-bound ∗ Memory-bandwidth bound ∗ Latency-bound – Statistical methods and best practices for benchmarking ∗ Estimation of uncertainty ∗ Confidence intervals – Analysis and presentation (graphs, etc.) – Timing techniques	• Describe throughput and latency and provide examples of each [Describe] • Explain speedup and the different forms of scaling and how they are computed [Explain] • Describe common performance bottlenecks [Describe] • Describe statistical methods and best practices for benchmarking software [Describe] • Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] • Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]
Readings : [**Bondi2015**], [**Gregg2020**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS2S1. Operating systems (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS2S1. Operating systems
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS221. Computer Systems Architecture. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The operating system is a collection of services needed to safely interface the hardware with applications. This course explores the mechanisms and policies needed to virtualize computation, memory, and I/O. Students will understand core concepts such as concurrency, persistence, and trust boundaries, which are fundamental for developing robust and efficient software systems.

5. GOALS

- Understand the role and purpose of modern operating systems.
- Master process scheduling and the lifecycle of execution units.
- Analyze memory management, including physical and virtual memory virtualization.
- Evaluate storage systems and file management techniques.
- Comprehend the basics of security and protection within an OS environment.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Role and Purpose of Operating Systems (8 hours)

Competences Expected: AG-C03,AG-C09

Topics	**Learning Outcomes**
• Operating systems mediate between general purpose hardware and application-specific software. • Universal operating system functions (e.g., process, user and device interfaces, persistence of data) • Extended and/or specialized operating system functions (e.g., embedded systems, server types such as file, web, multimedia, boot loaders and boot security) • Design issues (e.g., efficiency, robustness, flexibility, portability, security, compatibility, power, safety, tradeoffs between error checking and performance, flexibility and performance, and security and performance). See also: Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering • Influences of security, networking, multimedia, parallel and distributed computing • Overarching concern of security/protection: Neglecting to consider security at every layer creates an opportunity to inappropriately access resources. – Unauthorized access to files on an unencrypted drive can be achieved by moving the media to another computer. – Operating systems enforced security can be defeated by infiltrating the boot layer before the operating system is loaded. – Process isolation can be subverted by inadequate authorization checking at API boundaries. – Vulnerabilities in system firmware can provide attack vectors that bypass the operating system entirely. – Improper isolation of virtual machine memory, computing, and hardware can expose the host system to attacks from guest systems. – The operating system may need to mitigate exploitation of hardware and firmware vulnerabilities, leading to potential performance reductions (e.g., Spectre and Meltdown mitigations). • Exposure of operating systems functions in shells and systems programming. See also: Shell Scripting	• Understand the objectives and functions of modern operating systems [Explain] • Evaluate the design issues in different usage scenarios (e.g., real time OS, mobile, server) [Evaluate] • Understand the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve [Explain] • Understand how evolution and stability are desirable and mutually antagonistic in operating systems function [Explain]
Readings : [**Silberschatz2018**], [**Tanenbaum2014**]	

Unit 2: Shell Scripting (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Error/exception handling • Piping • System commands: – Interface with operating systems • Environment variables • File abstraction and operators • Data structures, such as arrays and lists. • Regular expressions • Programs and processes • Workflow	• Create and execute automated scripts to manage various system tasks [Create] • Solve various text processing problems through scripting [Solve]
Readings : [**Robbins2005**], [**Silberschatz2018**]	

Unit 3: Scheduling (16 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Preemptive and non-preemptive scheduling • Schedulers and policies (e.g., first come, first serve, shortest job first, priority, round robin, multilevel). See also: Resource Management • Concepts of Symmetric Multi-Processor (SMP) scheduling and cache coherence. See also: Memory Hierarchy • Timers (e.g., building many timers out of finite hardware timers). See also: Assembly Level Machine Organization • Fairness and starvation • Subtopics of operating systems such as energy-aware scheduling and real-time scheduling. See also: Performance and Energy Efficiency, Embedded Characteristics and Architectures, Real-Time, Memory and Verification, Systems, Programming and Sensors, Mobile Platforms: Capabilities and Asynchronous Computing, Mobile Platforms: Architecture and Constraints, Mobile Platforms: Development Fundamentals • Cooperative scheduling, such as Linux futexes and userland scheduling.	• Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes [Compare] • Explain the relationships between scheduling algorithms and application domains [Explain] • Explain the distinctions among types of processor scheduler such as short-term, medium-term, long-term, and I/O [Explain] • Evaluate a problem or solution to determine appropriateness for asymmetric and/or symmetric multiprocessing [Evaluate] • Evaluate a problem or solution to determine appropriateness as a process vs threads [Evaluate] • List some contexts benefitting from preemption and deadline scheduling [List] • Explain the ways that the logic embodied in scheduling algorithms are applicable to other operating systems mechanisms, such as first come first serve or priority to disk I/O, network scheduling, project scheduling, and problems beyond computing [Explain]
Readings : [**Silberschatz2018**], [**Stallings2017**]	

Unit 4: Memory Management (16 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Review of physical memory, address translation and memory management hardware See also: Memory Hierarchy, Discrete Mathematics • Impact of memory hierarchy including cache concept, cache lookup, and per-CPU caching on operating system mechanisms and policy See also: Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Logical and physical addressing, address space virtualization. See also: Memory Hierarchy, Discrete Mathematics • Concepts of paging, page replacement, thrashing and allocation of pages and frames • Allocation/deallocation/storage techniques (algorithms and data structure) performance and flexibility – Arenas, slab allocators, free lists, size classes, heterogeneously sized pages (huge pages) • Memory caching and cache coherence and the effect of flushing the cache to avoid speculative execution vulnerabilities. See also: Functional Organization, Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Security mechanisms and concepts in memory management including sandboxing, protection, isolation, and relevant vectors of attack. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles • Virtual memory: leveraging virtual memory hardware for OS services and efficiency	• Explain memory hierarchy and cost-performance tradeoffs [Explain] • Summarize the principles of virtual memory as applied to caching and paging [Summarize] • Evaluate the tradeoffs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed [Evaluate] • Describe the reason for and use of cache memory (performance and proximity, how caches complicate isolation and virtual machine abstraction) [Describe] • Code/Develop efficient programs that consider the effects of page replacement and frame allocation on the performance of a process and the system in which it executes [Create] • Explain how hardware is utilized for efficient virtualization [Explain]
Readings : [**Silberschatz2018**], [**ArpaciDusseau2018**]	

Unit 5: File Systems API and Implementation (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Concept of a file including data, metadata, operations, and access-mode • File system mounting • File access control • File sharing • Basic file allocation methods, including linked allocation table • File system structures comprising file allocation including various directory structures and methods for uniquely identifying files (e.g., name, identified or metadata storage location) • Allocation/deallocation/storage techniques (algorithms and data structure) impact on performance and flexibility (i.e., internal and external fragmentation and compaction) • Free space management such as using bit tables vs linking • Implementation of directories to segment and track file location	• Explain the choices to be made in designing file systems [Explain] • Evaluate different approaches to file organization, recognizing the strengths and weaknesses of each [Evaluate] • Apply software constructs appropriately given knowledge of the file system implementation [Apply]
Readings : [**Silberschatz2018**], [**Stallings2017**]	

Unit 6: Protection and Safety (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Overview of operating system security mechanisms. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles • Attacks and antagonism (scheduling, etc.). See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles • Review of major vulnerabilities in real operating systems. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles • Operating systems mitigation strategies such as backups. See also: System Reliability • Policy/mechanism separation. See also: Risk and Data Management, Security Governance, Management and Policy • Security methods and devices. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles – Rings of protection (history from Multics to virtualized x86) – x86_64 rings -1 and -2 (hypervisor and ME/PSP) • Protection, access control, and authentication. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Protocols, Applications and Advanced Cryptography, Cryptographic Algorithms, Cryptography Fundamentals	• Understand the requirement for protection and security mechanisms in operating systems [Explain] • List and describe the attack vectors that leverage OS vulnerabilities [List] • Understand the mechanisms available in an OS to control access to resources [Explain] • Summarize the features and limitations of an operating system that impact protection and security [Summarize]
Readings : [**Silberschatz2018**], [**Tanenbaum2014**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

FG106. Theater (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	FG106. Theater
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	1 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Theater as a comprehensive performing art invites the student to recognize themselves as a creative and social human being. Through theatrical practice, the student develops their physical, verbal, and emotional expression; strengthens their self-confidence and personal security; and cultivates communication, empathy, and teamwork skills. This course provides practical tools for artistic expression that contribute to the comprehensive education of the university student, stimulating their sensitivity, creativity, and imagination. At the same time, it integrates them into their academic community through shared experiential activities, opening paths of self-discovery and authentic communication with others.

5. GOALS

- Develop the student's body awareness and non-verbal communication skills through theatrical expression techniques.
- Foster creativity, spontaneity, and teamwork through dramatic play and improvisation.
- Learn about the historical trajectory of theater from its origins to its contemporary manifestations, including Peruvian theater.
- Value theatrical art as a tool for social integration, self-discovery, and expression of personal and collective identity.
- Apply the theatrical techniques learned in the creation and public presentation of a group stage proposal.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Art, Creativity, and Theater (6 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• What is Art? An experiential and personal experience. • The master key: creativity. • The importance of theater in personal and professional development. • Utility and approach of theatrical art. • Theater as a pedagogical tool and personal development instrument.	• Recognize the relevance of art and creativity in personal and social development [Usar]. • Connect the student with their group, valuing the importance of human communication and the social collective [Usar]. • Identify basic notions of theater and its function in university life [Usar].
Readings : [**Majorana**], [**PAVIS**], [**Miravalles1998**]	

Unit 2: Play: The Actor's Craft (6 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• I play, therefore I exist. Play as a universal language. • Child's play and dramatic play. • Group integration games and creativity games. • Spontaneous improvisation. • The theatrical sequence and body warm-up.	• Recognize play as a fundamental tool of theater [Usar]. • Internalize and revalue play as creative learning [Usar]. • Approach the theatrical experience spontaneously and naturally [Usar].
Readings : [**PAVIS**], [**Miravalles1998**]	

Unit 3: Body Expression and Dramatic Use of Objects (9 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Body awareness: the body as an instrument. • Awareness of the stage space. • Awareness of time and rhythm. • Creation of individual and collective sequences: body, space, and time. • The dramatic use of objects: theatrical play. • Theatrical performances using objects.	• Experiment with new forms of body expression and communication [Usar]. • Develop mechanisms for controlling and managing the body in stage space [Usar]. • Cultivate imagination and the ability to respond to auditory, rhythmic, and visual stimuli [Usar]. • Explore emotional states and collective atmospheres through movement [Usar].
Readings : [**PAVIS**], [**Grotowski1970**]	

Unit 4: Non-Verbal Communication in Theater (9 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Relaxation, concentration, and breathing. • Disinhibition and interaction with the group. • Improvisation: techniques and exercises. • Balance, weight, time, and body rhythm. • Movement analysis: types and qualities. • Stage presence and the actor's energy. • Dance and theatrical choreography. • Expressive gesture and the use of masks.	• Practice non-verbal communication skills [Usar]. • Engage in games and exercises of individual and group body language [Usar]. • Express emotions and feelings freely and creatively through original performances [Usar]. • Distinguish different types of acting and their technical foundations [Usar].
Readings : [**PAVIS**], [**Stanislavski1983**], [**Grotowski1970**]	

Unit 5: Traces of Theater Through Time (6 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• The origin of theater: ritual and performance in Greece and Rome. • Medieval theater and Commedia dell'Arte. • Spanish Golden Age theater and English Elizabethan theater. • Romanticism, Naturalism, and Realism: Ibsen and Chekhov. • Brecht's epic theater and Stanislavski's psychological theater. • Theater of the absurd, contemporary theater, and total theater. • Theater in Peru: Yuyachkani, La Tarumba, Pataclaun, and others.	• Understand the influence that society has exerted on theater throughout history [Usar]. • Appreciate the value and contribution of fundamental playwrights and theater directors [Usar]. • Analyze the social and historical context of theatrical art in different eras [Usar]. • Reflect on Peruvian theater and its contemporary expressions [Usar].
Readings : [**PAVIS**], [**Balta2001**]	

Unit 6: Theatrical Production (12 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Theatrical appreciation: attendance and critical analysis of plays. • The stage space: types of stages and their dramatic use. • The dramatic text: reading, analysis, and adaptation. • Character construction: technique and psychology. • Directing and staging: basic fundamentals. • Production elements: costumes, makeup, set design, and props. • Public presentation of group theatrical exercises.	• Employ theatrical creation as a manifestation of ideas and feelings before society [Usar]. • Apply the techniques practiced in a concrete and meaningful stage proposal [Usar]. • Exchange experiences and present brief group theatrical performances before an audience [Usar].
Readings : [**PAVIS**], [**Miravalles1998**], [**Brook1969**], [**Balta2001**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

BEI203. Technical and professional English V (Elective)

2. GENERAL INFORMATION

2.1 Course	:	BEI203. Technical and professional English V
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BEI202. Technical and professional English IV. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

A fundamental part of the integral formation of a professional is the ability to communicate in a foreign language in addition to the native language itself. In the engineering and computing career, technical English is the most critical tool for accessing the forefront of knowledge and research, writing scientific papers, participating in international conferences, and collaborating in global teams. This course brings students to the level of autonomous technical communicators capable of producing and critically evaluating academic and professional content.

5. GOALS

- Increase the ability and fluency to communicate complex technical ideas in English.
- Ensure that students interact with emphasis on scientific argumentation and advanced technical writing.
- Master vocabulary for software engineering, data science, and systems research.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Scientific Argumentation (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Advanced review of auxiliary verbs and verb tenses in academic writing. • Negative questions and emphatic answers in scientific debate. • Word formation: Prefixes and suffixes common in computing and science. • Correcting misconceptions: Refuting claims in technical discussions.	• By the end of the unit, students use auxiliaries and tenses precisely to construct scientific arguments, challenge erroneous technical claims, and engage in structured academic debate.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 2: Research Methodology (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Simple vs. Present Continuous: Distinguishing general facts from ongoing experiments. • State verbs and action verbs in experimental descriptions. • Vocabulary: Research design, experimental variables, data collection methods. • Writing: Description of a research method or experimental procedure.	• Students differentiate between stative and progressive uses to accurately describe scientific laws, ongoing experiments, and research methodologies in computing and engineering.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 3: Case Studies (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Narrative tenses: Simple Past, Past Continuous, and Past Perfect in case study analysis. • Structure of "Used to" and "Would" for describing legacy systems and past practices. • Vocabulary: Historical computing cases, paradigm shifts, engineering lessons learned. • Writing: Structured case study report on a computing incident or innovation.	• Students analyze and narrate sequences of technical events and historical computing cases fluently, applying narrative tenses correctly in professional case study reports.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 4: Research Proposals (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Predictions and plans: Future tense review for academic proposal writing. • Future Continuous and Future Perfect in project scope and outcomes. • Vocabulary: Research objectives, expected contributions, scope and limitations. • Writing: Short research proposal or technical project abstract.	• Students project research goals and anticipated outcomes, and make evidence-based predictions about the evolution of technology using appropriate future forms in formal academic writing.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 5: Technical Documentation (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Verb patterns (Verbs followed by -ing or infinitive) in instructional writing. • Expressions with "Like" for analogies and technical comparisons. • Vocabulary: API documentation, user manuals, technical specifications. • Debates: Standards for open-source documentation and reproducibility.	• Students use complex verb patterns to write clear, precise technical documentation and use analogical reasoning to explain technical concepts to diverse audiences.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 6: Long-term Projects (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Present Perfect Continuous vs. Simple Present Perfect in research progress reports. • Modal verbs of deduction (Present and Past) for interpreting experimental results. • Vocabulary: Long-running systems, sustained research programs, startup development. • Oral presentations: "Progress report on a computing research project".	• Students discuss long-term research and development activities, make deductions about technical outcomes, and present progress reports on computing projects with appropriate academic register.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 7: Advanced Technical Writing (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Relative clauses (Defining vs. Non-defining) in technical paper writing. • Advanced passive voice in formal scientific and engineering prose. • Vocabulary: Peer review, impact factors, citation practices, open access. • Writing: Abstract and introduction section of a computing research paper.	• Students write precise, well-structured technical and scientific texts using relative clauses and advanced passive constructions to add depth and clarity to research writing.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

Unit 8: English for Software Engineering (12 hours)	
Competences Expected: AG-C03,AG-C04	
Topics	**Learning Outcomes**
• Agile Methodologies: Scrum, Kanban, and XP terminology in English. • QA (Quality Assurance) and Testing: Test plans, bug reports, acceptance criteria. • Software requirements documentation: user stories, use cases, specifications. • Simulation: Technical team meeting (Sprint planning and retrospective).	• By the end of the course, students command software engineering and project management terminology, can produce professional technical documents, and actively participate in international technical meetings and reviews in English.
Readings : [**Soars023S**], [**Soars023W**], [**Soars023T**], [**Cambridge06**], [**MacGrew99**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS231. Networking and Communication (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS231. Networking and Communication
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS2S1. Operating systems . (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Networking and communication are foundational to modern computing systems, distributed applications, and the global internet. This course provides a comprehensive introduction to the full CS2023 Networking and Communication Body of Knowledge—from physical transmission and data link protocols, through reliable transport and routing, to application services, security, mobility, and emerging paradigms such as SDN and quantum networking. Students will develop both theoretical understanding and practical skills applicable to the design and analysis of modern networked systems.

5. GOALS

- Understand the layered architecture of the TCP/IP and OSI models, including the role and responsibilities of each layer.
- Explain single-hop communication mechanisms including MAC protocols, encoding, Ethernet, WiFi, and LAN topologies.
- Master the principles of reliable data transfer, flow control, and congestion control.
- Analyze routing paradigms, forwarding mechanisms, and IP scalability in local and wide area networks.
- Comprehend fundamental application-layer protocols of the Internet (DNS, HTTP, TCP, UDP) and implement socket-based applications.
- Understand network security threats and the countermeasures used to protect networked systems.
- Describe principles of mobile networking including cellular networks, wireless LANs, and device-to-device communication.
- Identify emerging networking paradigms such as SDN, network virtualization, and quantum networking.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Fundamentals of Networks and Communications (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Importance of networking in contemporary computing, and associated challenges. See also: Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation • Organization of the internet (e.g., users, Internet Service Providers, autonomous systems, content providers, content delivery networks) • Switching techniques (e.g., circuit and packet) • Layers and their roles (application, transport, network, datalink, and physical) • Layering principles (e.g., encapsulation and hourglass model). See also: Digital Systems and Logic, Sequential and Parallel Processing, State Machines • Network elements (e.g., routers, switches, hubs, access points, and hosts) • Basic queueing concepts (e.g., relationship with latency, congestion, service levels, etc.)	• Articulate the organization of the internet [Articulate] • List and define the appropriate network terminology [List] • Describe the layered structure of a typical networked architecture [Describe] • Identify the different types of complexity in a network (edges, core, etc.) [Analyze]
Readings : [**Kurose2021**], [**Tanenbaum2021**], [**PetersonDavie2022**]	

Unit 2: Single Hop Communication (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Introduction to modulation, bandwidth, and communication media • Encoding and Framing • Medium Access Control (MAC) (e.g., random access and scheduled access) • Ethernet and WiFi • Switching (e.g., spanning trees, VLANS). • Local Area Network Topologies (e.g., data center, campus networks).	• Describe some basic aspects of modulation, bandwidth, and communication media [Describe] • Describe in detail a MAC protocol [Describe] • Demonstrate understanding of encoding and framing solution tradeoffs [Demonstrate] • Describe details of the implementation of Ethernet [Describe] • Describe how switching works [Describe] • Describe one kind of a LAN topology [Describe]
Readings : [**Tanenbaum2021**], [**PetersonDavie2022**]	

Unit 3: Networked Applications (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Naming and address schemes (e.g., DNS, and Uniform Resource Identifiers) • Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination • Diversity of networked application demands (e.g., latency, bandwidth, and loss tolerance). See also: Communication, Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context, Digital Equity and Accessibility, Social Impact and AI • Coverage of application-layer protocols (e.g., HTTP) • Interactions with TCP, UDP, and Socket APIs. See also: Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities	• Define the principles of naming, addressing, resource location [Define] • Analyze the needs of specific networked application demands [Analyze] • Describe the details of one application layer protocol [Describe] • Implement a simple client-server socket-based application [Implement]
Readings : [**Kurose2021**], [**PetersonDavie2022**]	

Unit 4: Reliability Support (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Unreliable delivery (e.g., UDP) • Principles of reliability (e.g., delivery without loss, duplication, or out of order). See also: System Reliability • Error control (e.g., retransmission, error correction) • Flow control (e.g., stop and wait, window based) • Congestion control (e.g., implicit and explicit congestion notification) • TCP and performance issues (e.g., Tahoe, Reno, Vegas, Cubic)	• Describe the operation of reliable delivery protocols [Describe] • List the factors that affect the performance of reliable delivery protocols [List] • Describe some TCP reliability design issues [Describe] • Design and implement a simple reliable protocol [Design]
Readings : [**Kurose2021**], [**Tanenbaum2021**]	

Unit 5: Routing and Forwarding (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Routing paradigms and hierarchy (e.g., intra/inter domain, centralized and decentralized, source routing, virtual circuits, QoS) • Forwarding methods (e.g., forwarding tables and matching algorithms) • IP and Scalability issues (e.g., NAT, CIDR, BGP, different versions of IP)	• Describe various routing paradigms and hierarchies [Describe] • Describe how packets are forwarded in an IP network [Describe] • Describe how the Internet tackles scalability challenges [Describe]
Readings : [**Kurose2021**], [**Tanenbaum2021**], [**PetersonDavie2022**]	

Unit 6: Network Security (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• General intro about security (Threats, vulnerabilities, and countermeasures). See also: Computer Crimes and Attacks, Security Laws, Policies and Responsibility, AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering • Network specific threats and attack types (e.g., denial of service, spoofing, sniffing and traffic redirection, attacker-in-the-middle, message integrity attacks, routing attacks, ransomware, and traffic analysis). See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering • Countermeasures: – Cryptography (e.g. SSL, TLS, symmetric/asymmetric) – Architectures for secure networks (e.g., secure channels, secure routing protocols, secure DNS, VPNs, DMZ, Zero Trust Network Access, hyper network security, anonymous communication protocols, isolation) – Network monitoring, intrusion detection, firewalls, spoofing and DoS protection, honeypots, tracebacks, BGP Sec, RPKI	• Describe some of the threat models of network security [Describe] • Describe specific network-based countermeasures [Describe] • Analyze various aspects of network security from a case study [Analyze]
Readings : [**Stallings2022**], [**Kurose2021**]	

Unit 7: Mobility Support (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Principles of cellular communication (e.g., 4G, 5G) • Principles of Wireless LANs (mainly 802.11) • Device to device communication (e.g., IoT communication) • Multi-hop wireless networks (e.g., ad hoc networks, opportunistic, delay tolerant)	• Describe some aspects of cellular communication such as registration [Describe] • Describe how 802.11 supports mobile users [Describe] • Describe practical uses of device-to-device communication, as well as multihop [Describe] • Describe one type of mobile network such as ad hoc [Describe]
Readings : [**Tanenbaum2021**], [**PetersonDavie2022**]	

Unit 8: Emerging Topics (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Middleboxes (e.g., advances in usage of AI, intent-based networking, filtering, deep packet inspection, load balancing, NAT, CDN) • Network Virtualization (e.g., SDN, Data Center Networks) • Quantum Networking (e.g., Intro to the domain, teleportation, security, Quantum Internet) • Satellite, mmWave, Visible Light	• Describe the value of advances in middleboxes in networks [Describe] • Describe the importance of Software Defined Networks [Describe] • Describe some of the added value achieved by using Quantum Networking [Describe]
Readings : [**PetersonDavie2022**], [**Tanenbaum2021**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS311. Algorithms for Complex Problems (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS311. Algorithms for Complex Problems
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS212. Analysis and Design of Algorithms. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Competitive Programming trains students to solve complex algorithmic problems under time and resource constraints. This course bridges the gap between theoretical algorithm design and high-performance implementation. It covers advanced techniques in dynamic programming, graph theory, and mathematics, fostering the analytical thinking required for top-tier software engineering interviews and international programming contests like the ICPC.

5. GOALS

- Master advanced competitive programming techniques and paradigms.
- Develop the ability to quickly analyze the complexity of potential solutions.
- Implement efficient data structures for range queries and dynamic updates.
- Apply specialized mathematical and geometric algorithms to solve contests problems.
- Optimize code for execution speed and memory usage in a competitive environment.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Algorithmic Strategies (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Complete Search: Recursive Backtracking and Bitmasking. • Greedy Strategies: Choice property and optimal substructure. • Divide and Conquer: Binary Search on the answer and Ternary Search. • Dynamic Programming: State reduction, Digit DP, and DP on trees.	• Apply search pruning techniques to solve NP-hard problems in small constraints [Usar] • Identify the most efficient paradigm for a given problem statement [Evaluar]
Readings : [**Halim2020**], [**Laaksonen2020**]	

Unit 2: Foundational Algorithms (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Fenwick Trees (Binary Indexed Trees). • Segment Trees: Lazy Propagation and Persistent Segment Trees. • Square Root Decomposition and Mo's Algorithm. • Sparse Tables for RMQ.	• Implement range query structures for real-time data updates [Evaluar] • Analyze the trade-offs between different range query data structures [Usar]
Readings : [**Halim2020**], [**Skiena2020**]	

Unit 3: Foundational Algorithms (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Network Flow: Max Flow Min Cut (Dinic's and Edmonds-Karp). • Strongly Connected Components (Tarjan's and Kosaraju's). • Bipartite Matching and Heavy-Light Decomposition. • Centroid Decomposition on trees.	• For each algorithm in this unit explain step-by-step how the algorithm operates [Explain] • For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply] • Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create] • Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain] • For each of the algorithms and algorithmic approaches in the KA Core topics: – Explain a prototypical example of the algorithm, and – Explain step-by-step how the algorithm operates. [Explain]
Readings : [**Laaksonen2020**], [**Halim2020**]	

Unit 4: Foundational Algorithms (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Number Theory: Sieve of Eratosthenes, Extended Euclidean, and Modular Inverse. • Combinatorics: Inclusion-Exclusion and Burnside's Lemma. • Computational Geometry: Cross product, Convex Hull (Monotone Chain), and Line intersection. • Fast Fourier Transform (FFT) for polynomial multiplication.	• Solve combinatorial problems using modular arithmetic [Usar] • Implement geometric primitives for spatial relationship analysis [Evaluar]
Readings : [**Skiena2020**], [**Laaksonen2020**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS312. Advanced Data Structures (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS312. Advanced Data Structures
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS212. Analysis and Design of Algorithms. (5^{th} Sem)

3. PROFESSORS

Lecturer

- Ernesto Cuadros-Vargas <ecuadrosv@uni.edu.pe>
  - PhD in Computer Science, ICMC-USP, Brazil, 2004.
  - MSc in Computer Science, ICMC-USP, Brazil, 1998.

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Algorithms and data structures are a fundamental part of Computer Science that allow us to organize information efficiently. For any professional in the field, a solid foundation in this area is crucial. This course focuses on advanced structures such as Multidimensional Access Methods, Spatio-Temporal Access Methods, and Metric Access Methods, which are essential for high-performance applications in search engines, geographic information systems (GIS), and big data processing.

5. GOALS

- Understand, design, and implement innovative data structures for multidimensional data.
- Apply similarity search techniques and metric indexing to complex data types.
- Analyze the ”Curse of Dimensionality” and its impact on search efficiency.
- Research and present contemporary indexing methods for Big Data in English.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Static and dynamic implementation techniques. • Data abstraction and encapsulation. • Abstract Data Types (ADT). • Compile-time vs. Runtime polymorphism. • Generic data structures and C++ Templates. • Reusable components and design patterns for data structures.	• Evaluate the impact of static and dynamic techniques on data structure design [Evaluar]. • Implement generic and reusable data structures using advanced programming paradigms [Usar]. • Analyze performance trade-offs between different implementation strategies [Evaluar].
Readings : [**Cuadros2004Implementing**], [**Knuth2007TAOCP-V-I**], [**Gamma94**], [**VanJosGreg:Templates:2018**]	

Unit 2: Multidimensional Access Methods (Part I) (8 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Introduction to multidimensional data. • Vector space representation. • The Curse of Dimensionality: theoretical foundations. • Impact of high dimensionality on traditional indexing. • Real-world applications in search engines and pattern recognition.	• Explain the significance of multidimensional data representation [Familiarizarse]. • Analyze the complexity and performance degradation in high-dimensional spaces [Usar]. • Discuss real-world scenarios where dimensionality impact is critical [Usar].
Readings : [**Gaede98multidimensional**], [**Samet2006SAM-MAM**]	

Unit 3: Multidimensional Access Methods (Part II) (20 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Spatial data structures: Quadtree and Octree. • Point Access Methods: Kd-Tree. • Region Access Methods: R-Tree (Guttman), R+ tree, R* tree. • Variations of R-trees and their relation to paging and block size. • Advanced structures: X-tree and SS-tree.	• Implement spatial data structures for high-volume data indexing [Usar]. • Evaluate the limitations of tree-based spatial structures in specific domains [Evaluar]. • Implement range search and Nearest Neighbor (k-NN) strategies [Usar].
Readings : [**Samet2006SAM-MAM**]	

Unit 4: Metric Access Methods (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Metric Spaces: definitions and properties. • Similarity search vs. Exact search. • Vantage Point Tree (VP-Tree). • Slim-Tree and M-Tree. • Pivot-based indexing.	• Model similarity search problems using metric space properties [Usar]. • Compare metric access methods against multidimensional methods for non-vector data [Evaluar].
Readings : [**Samet2006SAM-MAM**], [**Zezula07**]	

Unit 5: Technical and Professional Communication (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Seminar on recent research papers (SIGMOD, VLDB, ICDE). • Indexing for Big Data and streaming data. • Compact data structures and succinct representations. • Future trends in similarity search.	• Research new methods for indexing large volumes of complex data [Usar]. • Present and lead technical discussions in English on state-of-the-art methods [Evaluar]. • Identify potential thesis topics within the advanced data structures domain [Familiarizarse].
Readings : [**Navarro:CompactDS:2016**], [**Traina00SlimTree**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.

- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS342. Compilers (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS342. Compilers
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS211. Theory of Computation. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course provides a deep understanding of how high-level languages are transformed into executable machine code. The study of compilers allows students to understand language semantics, memory management, and code optimization, which are fundamental skills for designing domain-specific languages and high-performance software engineering.

5. GOALS

- Understand the phases of a compiler and construction tools.
- Implement lexical and syntactic analyzers based on grammars.
- Generate and optimize intermediate code for a specific architecture.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Program Analysis and Analyzers (20 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Relevant program representations, such as basic blocks, control-flow graphs, def-use chains, and static single assignment • Undecidability and consequences for program analysis • Flow-insensitive analysis, such as type-checking and scalable pointer and alias analysis • Flow-sensitive analysis, such as forward and backward dataflow analyses • Path-sensitive analysis, such as software model checking and software verification • Tools and frameworks for implementing analyzers • Role of static analysis in program optimization and data dependency analysis during exploitation of concurrency • Role of program analysis in (partial) verification and bug-finding • Parallelization: – Analysis for auto-parallelization – Analysis for detecting concurrency bugs	• Explain the difference between dataflow graph and control flow graph [Explain] • Explain why non-trivial sound program analyses must be approximate [Explain] • Argue why an analysis is correct (sound and terminating) [Argue] • Explain why potential aliasing limits sound program analysis and how alias analysis can help [Explain] • Use the results of a program analysis for program optimization and/or partial program correctness [Use]
Readings : [**Aho2006**], [**Appel2004**]	

Unit 2: Language Translation and Execution (24 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Execution models for JIT (Just-In-Time), compiler, interpreter • Use of intermediate code, e.g., bytecode • Limitations and benefits of JIT, compiler, and interpreter • Cross compilers/transpilers • BNF and extended BNF representation of context-free grammar • Parse tree using a simple sentence such as arithmetic expression or if-then-else statement • Execution as native code or within a virtual machine • Language translation pipeline: syntax analysis, parsing, optional type-checking, translation/code generation and optimization, linking, loading, execution • Run-time representation of core language constructs such as objects (method tables) and functions that can be passed as parameters to and returned from functions (closures) • Secure compiler development	• Explain and understand the differences between compiled, JIT, and interpreted language implementations, including the benefits and limitations of each [Explain] • Differentiate syntax and parsing from semantics and evaluation [Differentiate] • Use BNF and extended BNF to specify the syntax of simple constructs such as if-then-else, type declaration and iterative constructs for known languages such as C++ or Python [Use] • Illustrate the parse tree using a simple sentence/arithmetic expression [Apply] • Illustrate translation of syntax diagrams to BNF/extended BNF for simple constructs such as if-then-else, type declaration, iterative constructs, etc [Apply] • Illustrate ambiguity in parsing using nested if-then-else/arithmetic expression and show resolution using precedence order [Apply] • Discuss the benefits and limitations of garbage collection, including the notion of reachability [Debate]
Readings : [**Aho2006**], [**Appel2004**]	

Unit 3: Run-time Behavior and Systems (20 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Process models using stacks and heaps to allocate and deallocate activation records and recovering environments using frame pointers and return addresses during a procedure call including parameter-passing examples • Schematics of code lookup using hash tables for methods in implementations of object-oriented programs • Data layout for objects and activation records • Object allocation in heap • Implementing virtual entities and virtual methods; virtual method tables and their application • Run-time behavior of object-oriented programs • Compare and contrast allocation of memory during information exchange using parameter-passing and non-local variables (using chain of static links). • Dynamic memory management approaches and techniques: malloc/free, garbage collection (mark-sweep, copying, reference counting), regions (also known as arenas or zones) • Just-in-time compilation and dynamic recompilation • Interface to operating system (e.g., for program initialization) • Interoperability between programming languages including parameter-passing mechanisms and data representation: – Big endian, little endian – Data layout of composite data types such as arrays • Other common features of virtual machines, such as class loading, threads, and security checking • Sandboxing	• Discuss benefits and limitations of automatic memory management [Debate] • Explain the use of metadata in run-time representations of objects and activation records, such as class pointers, array lengths, return addresses, and frame pointers [Explain] • Compare and contrast static allocation vs stack-based allocation vs heap-based allocation of data elements [Compare] • Explain why some data elements cannot be automatically deallocated at the end of a procedure/method call (need for garbage collection) [Explain] • Discuss advantages, disadvantages, and difficulties of just-in-time and dynamic recompilation [Debate] • Discuss the use of sandboxing in mobile code [Debate] • Identify the services provided by modern language run-time systems [Analyze]
Readings : [**Aho2006**], [**Cooper2011**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI263. Introduction to Machine Learning (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	AI263. Introduction to Machine Learning
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS261. Artificial Intelligence. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course introduces the fundamentals of machine learning, covering classical and modern algorithms for classification, regression, and clustering problems. It focuses on practical implementation using scikit-learn and TensorFlow, with applications in computer vision and natural language processing.

5. GOALS

- Understand the mathematical principles behind ML algorithms.
- Implement complete ML pipelines with Python.
- Evaluate and optimize models using standard metrics.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: ML Fundamentals (15 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Supervised vs unsupervised learning • Overfitting and regularization • Cross-validation and learning curves	• Explain the bias-variance tradeoff [Familiarizarse] • Implement k-fold cross-validation [Usar]
Readings : [**Bishop2006**], [**Goodfellow2016**]	

Unit 2: Linear Models (15 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Linear and logistic regression • Support Vector Machines (SVM) • Linear discriminant analysis	• Program linear models with scikit-learn [Usar] • Interpret regression coefficients [Evaluar]
Readings : [**Hastie2009**], [**Murphy2012**]	

Unit 3: Ensembles and Trees (15 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Decision trees and random forests • Gradient Boosting (XGBoost, LightGBM) • Stacking and voting	• Optimize hyperparameters with GridSearch [Usar] • Visualize decision trees [Evaluar]
Readings : [**Hastie2009**], [**Geron2022**]	

Unit 4: Basic Neural Networks (15 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Multilayer perceptrons (MLP) • Backpropagation and optimizers • Introduction to Keras/TensorFlow	• Build simple neural networks [Usar] • Monitor training with TensorBoard [Evaluar]
Readings : [**Goodfellow2016**], [**Chollet2021**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

FI201. Computational Physics (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	FI201. Computational Physics
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	BFI101. Physics I. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course applies the principles of physics to computational problems, with an emphasis on light, wave propagation, collisions, and energy transfer. These concepts are essential in areas such as computer graphics, physical simulations, and video game development.

5. GOALS

- Understand the physical principles relevant to computing.
- Apply these principles to solve specific computational problems.
- Implement physics-based algorithms for simulations and computer graphics.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Optics and Light Propagation (10 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Nature of light. • Reflection and refraction. • Lenses and mirrors. • Interference and diffraction. • Illumination models (e.g., Phong, Blinn-Phong).	• Describe the properties of light and its propagation. [Familiarizarse] • Apply the laws of reflection and refraction. [Usar] • Implement illumination models in computer graphics. [Evaluar]
Readings : [**YoungFreedman2018**], [**Hecht2017**]	

Unit 2: Collisions and Energy Transfer (8 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Impulse and linear momentum. • Elastic and inelastic collisions. • Conservation of energy in collisions. • Deformation of elastic meshes (e.g., mass-spring model).	• Apply the principles of conservation of linear momentum and energy in collisions. [Familiarizarse] • Model the deformation of elastic meshes due to impact. [Usar] • Implement collision simulations in a computational environment. [Evaluar]
Readings : [**YoungFreedman2018**], [**Taylor2005**]	

Unit 3: Rigid Body Mechanics (8 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Rotation of rigid bodies. • Moment of inertia. • Torque and rotational kinetic energy.	• Describe the rotation of rigid bodies. [Familiarizarse] • Calculate the moment of inertia. [Usar] • Apply the equations of rotational dynamics. [Evaluar]
Readings : [**YoungFreedman2018**], [**Taylor2005**]	

Unit 4: Fluid Dynamics (6 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Basic principles of fluid dynamics. • Viscosity. • Laminar and turbulent flow.	• Describe the properties of fluids. [Familiarizarse] • Explain the concepts of viscosity and laminar/turbulent flow. [Usar] • Solve simple fluid dynamics problems. [Evaluar]
Readings : [**YoungFreedman2018**]	

Unit 5: Thermodynamics (6 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Laws of thermodynamics. • Heat transfer.	• State the laws of thermodynamics. [Familiarizarse] • Describe the mechanisms of heat transfer. [Usar]
Readings : [**YoungFreedman2018**]	

Unit 6: Physical Simulation (10 hours)	
Competences Expected: AG-C08,AG-C11	
Topics	**Learning Outcomes**
• Numerical methods for physical simulation. • Verlet integration. • Collision detection. • Particle systems. • Constraints and solvers.	• Implement basic numerical methods for physical simulation. [Familiarizarse] • Use Verlet integration to simulate motion. [Usar] • Implement collision detection algorithms. [Evaluar]
Readings : [**Taylor2005**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS251. Computer Graphics (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS251. Computer Graphics
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	MA202. Numerical Methods. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Computer Graphics is a fundamental area of Computer Science that focuses on the mathematical and computational techniques for generating and manipulating images. This course provides students with the foundations of 2D and 3D modeling, rendering pipelines, and the physical principles of light and color, enabling them to develop high-performance visual applications, simulations, and interactive systems.

5. GOALS

- Understand the graphics pipeline and coordinate system transformations.
- Implement rendering algorithms for realistic and real-time visualization.
- Apply mathematical modeling to create and manipulate 3D objects.

6. COMPETENCES

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Usage)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Fundamental Concepts of Graphics and Interactive Techniques (16 hours)	
Competences Expected: AG-C07,AG-C08	
Topics	**Learning Outcomes**
• Overview of computer graphics pipeline and interactive systems • Basic rendering concepts: rasterization, ray tracing, shading models • Color theory and color spaces • Coordinate systems and transformations • Basic interaction techniques: input devices, event handling • Advanced rendering concepts: global illumination, physically based rendering • Graphics hardware and acceleration techniques • Human perception and visual cognition	• Explain the basic stages of the computer graphics pipeline [Explain] • Describe basic rendering techniques such as rasterization and ray tracing [Describe] • Apply color theory to design visually effective graphics [Apply] • Implement coordinate transformations for 2D and 3D graphics [Implement] • Design basic interactive applications using common input devices and event handling [Design] • Compare advanced rendering techniques such as global illumination and physically based rendering [Compare] • Analyze the role of graphics hardware in accelerating rendering tasks [Analyze] • Explain how human perception influences graphics and interaction design [Explain]
Readings : [**Marschner2021**], [**Hearn2014**]	

Unit 2: Applied Rendering and Techniques (16 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Rasterization algorithms: scan conversion, antialiasing • Ray tracing: acceleration structures, sampling, shading • Texture mapping and filtering • Shading models: Lambertian, Phong, BRDF • Advanced techniques: shadow mapping, environment mapping, deferred rendering	• Implement rasterization algorithms for primitive rendering [Implement] • Implement basic ray tracing with acceleration structures [Implement] • Apply texture mapping and filtering techniques to enhance visual realism [Apply] • Compare different shading models and their applications [Compare] • Implement advanced rendering techniques such as shadow mapping or deferred rendering [Implement]
Readings : [**Marschner2021**], [**Hearn2014**], [**Angel2014**]	

Unit 3: Geometric Modeling (16 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Geometric representations: polygons, curves, surfaces, implicit forms • Modeling techniques: splines, subdivision surfaces, constructive solid geometry • Procedural modeling and generation • Geometric algorithms: intersection, distance, decomposition	• Describe different geometric representations and their trade-offs [Describe] • Implement modeling techniques such as spline curves or subdivision surfaces [Implement] • Create procedural models using algorithms or rules [Create] • Apply geometric algorithms to solve problems like intersection testing or mesh processing [Apply]
Readings : [**Marschner2021**], [**Angel2014**]	

Unit 4: Visualization (16 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Principles of data visualization: perception, cognition, and effective encoding • Types of visualizations: scientific, information, geographical • Visualization techniques for multivariate, temporal, and hierarchical data • Interactive visualization and exploration • Evaluation of visualization effectiveness	• Apply principles of perception and cognition to design effective visualizations [Apply] • Select appropriate visualization techniques for different data types and tasks [Evaluate] • Implement interactive visualizations for data exploration [Implement] • Evaluate the effectiveness of a visualization using empirical methods [Evaluate]
Readings : [**Marschner2021**]	

Unit 5: Shading and Advanced Rendering (12 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Advanced shading: physically based shading, subsurface scattering • Global illumination: radiosity, photon mapping, path tracing • Real-time rendering techniques: level of detail, culling, occlusion • GPU programming and shaders	• Implement advanced shading models for realistic materials [Implement] • Compare global illumination algorithms and their trade-offs [Compare] • Optimize real-time rendering using techniques like level of detail and culling [Create] • Develop GPU shaders for custom rendering effects [Create]
Readings : [**Marschner2021**], [**Angel2014**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS292. Software Engineering II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS292. Software Engineering II
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS291. Software Engineering I. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Software Engineering II builds upon the foundational principles of the first course, shifting focus towards advanced design, software evolution, and the management of large-scale projects. It explores the complexities of maintaining systems, ensuring software quality through rigorous validation, and the application of professional teamwork and project management techniques in collaborative environments.

5. GOALS

- Apply advanced software design principles and patterns to solve complex architectural problems.
- Master software configuration management and collaborative development practices.
- Understand the principles of software evolution, refactoring, and maintenance.
- Implement systematic testing strategies and understand software reliability.
- Develop professional skills for team-based software development and stakeholder management.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Team Collaboration and Dynamics (14 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture • Common causes of team conflict, and approaches for conflict resolution. • Cooperative programming: – Pair programming or Swarming – Code review – Collaboration through version control • Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics – Advantages of teamwork – Risks and complexity of such collaboration • Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule	• Follow effective team communication practices [Analyze] • Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate] • Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze] • Collaborate effectively in cooperative development/programming [Analyze] • Propose and delegate necessary roles and responsibilities in a software development team [Propose] • Compose and follow an agenda for a team meeting [Compose] • Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]
Readings : [**Sommerville2015**], [**Pressman2019**]	

Unit 2: Principles and Software Architecture (16 hours)

Competences Expected: AG-C03,AG-C09

Topics	**Learning Outcomes**
• System design principles. System Reliability – Levels of abstraction (e.g., architectural design and detailed design) – Separation of concerns – Information hiding – Coupling and cohesion • Software architecture. System Reliability – Design paradigms * Top-down functional decomposition/layered design * Data-oriented architecture * Object-oriented analysis and design * Event-driven design – Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) – Identifying component boundaries and dependencies • Programming in the large vs programming in the small. System Reliability • Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering	• Identify the standard software architecture of a given high-level design [Analyze] • Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] • Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] • Identify the dependencies among a set of software components in an architectural design [Analyze]

Readings : [**Gamma1994**], [**Pressman2019**]

Unit 3: Coding Practices (14 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Practical small-scale testing Software Development Practices – Unit testing – Test-driven development - This is particularly valuable for students psychologically, as it is far easier to engage constructively with the challenge of identifying challenging inputs for a given API (edge cases, corner cases) a priori. If they implement first, the instinct is often to avoid trying to crash their new creation, while a test-first approach gives them the intellectual satisfaction of spotting the problem cases and then watching as more tests pass during the development process. • Documentation Software Development Practices – Interface documentation - describe interface requirements, potentially including (formal or informal) contracts, pre and post conditions, invariants. – Implementation documentation should focus on tricky and non-obvious pieces of code, whether because the code is using advanced language features, or the behavior of the code is complex. (Do not add comments that re-state common/obvious operations and simple language features.) ∗ Clarify dataflow, computation, etc., focusing on what the code is. ∗ Identify subtle/tricky pieces of code and refactor to be self-explanatory if possible or provide appropriate comments to clarify. • Coding style Software Development Practices – Style guides – Commenting – Naming • ”Best Practices” for coding: techniques, idioms/patterns, mechanisms for building quality programs Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security – Defensive coding practices – Secure coding practices and principles – Using exception handling mechanisms to make programs more robust, fault-tolerant • Debugging Software Development Practices • Logging • Use of libraries and frameworks developed by others	• Write appropriate unit tests for a small component (several functions, a single type, etc.) [Write] • Write appropriate interface and (if needed) implementation comments for a small component [Write] • Describe techniques, coding idioms and mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness [Describe] • Write robust code using exception handling mechanisms [Write] • Describe secure coding and defensive coding practices [Describe] • Select and use a defined coding standard in a small software project [Use] • Compare and contrast integration strategies including top-down, bottom-up, and sandwich integration [Compare] • Describe the process of analyzing and implementing changes to code base developed for a specific project [Describe] • Describe the process of analyzing and implementing changes to a large existing code base [Describe]

Unit 4: Verification and validation Concepts (10 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation concepts – Verification: Are we building the thing right? – Validation: Did we build the right thing? • Why testing matters: Does the component remain functional as the code evolves? • Testing objectives – Usability – Reliability – Conformance to specification – Performance – Security	• Explain why testing is important [Explain] • Distinguish between program validation and verification [Distinguish] • Describe different objectives of testing [Describe]
Readings : [**Pressman2019**]	

Unit 5: Testing Tools and Analysis (4 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation tooling and automation – Static analysis – Code coverage – Fuzzing – Dynamic analysis and fault containment (sanitizers, etc.) – Fault logging and fault tracking • Test planning and generation – Fault estimation and testing termination including defect seeding – Use of random and pseudo random numbers in testing • Testing asynchronous, parallel, and concurrent systems • Verification and validation of non-code artifacts (documentation, training materials) • Behavior-driven development (BDD) – Expressing acceptance criteria as Gherkin Given/When/Then scenarios – Executing BDD scenarios as automated end-to-end tests • Mutation testing and static code-quality analysis – Mutation testing to assess test-suite effectiveness – Static detection of code smells and excessive cyclomatic complexity	• Describe and compare different tools for verification and validation [Describe] • Automate the use of different tools in a small software project [Design] • Explain how and when random numbers should be used in testing [Explain] • Describe approaches for fault estimation [Describe] • Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] • Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] • Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] • Describe techniques for the verification and validation of non-code artifacts [Describe] • Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] • Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]
Readings : [**Sommerville2015**], [**Pressman2019**]	

Unit 6: Performance Testing and Benchmarking (4 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Performance testing and benchmarking – Throughput and latency – Degradation under load (stress testing, FIFO vs LIFO handling of requests) – Speedup and scaling ∗ Amdahl's law ∗ Gustafson's law ∗ Soft and weak scaling – Identifying and measuring figures of merits – Common performance bottlenecks ∗ Compute-bound ∗ Memory-bandwidth bound ∗ Latency-bound – Statistical methods and best practices for benchmarking ∗ Estimation of uncertainty ∗ Confidence intervals – Analysis and presentation (graphs, etc.) – Timing techniques	• Describe throughput and latency and provide examples of each [Describe] • Explain speedup and the different forms of scaling and how they are computed [Explain] • Describe common performance bottlenecks [Describe] • Describe statistical methods and best practices for benchmarking software [Describe] • Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] • Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]
Readings : [**Bondi2015**], [**Gregg2020**]	

Unit 7: Reliability Concepts (10 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures • Identifying reliability requirements for different kinds of software • Software failures caused by defects/bugs, and so for high reliability the goal is to have minimum defects - by injecting fewer defects (better training, education, planning), and by removing most of the injected defects (testing, code review, etc.) • Software reliability, system reliability and failure behavior • Defect injection and removal cycle, and different approaches for defect removal • Compare the "error budget" approach to reliability with the "error-free" approach and identify domains where each is relevant.	• Describe how to determine the level of reliability required by a software system [Describe] • Explain the problems that exist in achieving very high levels of reliability [Explain] • Understand approaches to minimizing faults that can be applied at each stage of the software lifecycle [Explain]
Readings : [**Sommerville2015**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS2H1. User Experience (UX) (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS2H1. User Experience (UX)
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS291. Software Engineering I. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course focuses on the design, evaluation, and implementation of interactive computing systems for human use. Beyond classical HCI principles, it addresses the emerging challenges of interacting with AI systems, intelligent agents, and natural language processing software. Students will develop competencies in user-centered design, accessibility, ethical accountability, and the evaluation of both traditional interfaces and AI-driven conversational systems, acquiring skills essential for designing responsible and human-centered technology.

5. GOALS

- Understand human cognitive, physical, and emotional characteristics relevant to interactive system design.
- Apply ethical accountability and value-sensitive design principles to HCI practice, including algorithmic fairness and bias.
- Apply accessibility standards and inclusive design principles for diverse populations and abilities.
- Execute rigorous usability evaluation methods to iterate and improve user interfaces.
- Design interactive systems using user-centered and iterative design processes across multiple platforms and domains.
- Analyze the societal, ethical, and professional dimensions of interactive technology.
- Understand agent architectures and apply principles of human-agent interaction including AI assistants and autonomous systems.
- Apply natural language processing techniques to design and evaluate human-software interfaces based on language models.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Understanding the User: Individual goals and interactions with others (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Human capabilities and limitations: perception, cognition, motor skills • User characteristics: demographics, expertise, cultural background • User goals, tasks, and contexts of use • Social and organizational aspects of interaction • Cognitive models and theories relevant to HCI • Emotional and affective aspects of interaction • User research methods: interviews, surveys, observations	• Describe human capabilities and limitations that affect interaction design [Describe] • Analyze how user characteristics influence interaction with computing systems [Analyze] • Identify user goals and tasks in specific contexts of use [Analyze] • Explain social and organizational factors that impact technology adoption and use [Explain] • Apply cognitive models to predict and explain user behavior [Apply] • Consider emotional and affective factors in interaction design [Analyze] • Conduct user research using appropriate methods such as interviews or observations [Evaluate]
Readings : [**Norman2013**], [**Johnson2020**]	

Unit 2: Accountability and Responsibility in Design (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Ethical considerations in HCI: fairness, transparency, accountability • Design for values and value-sensitive design • Responsible innovation and design ethics • Algorithmic accountability and bias in interactive systems • Professional responsibility in HCI practice • Legal and regulatory considerations for interactive systems	• Identify ethical issues in HCI design and implementation [Analyze] • Apply value-sensitive design approaches to technology development [Apply] • Discuss responsible innovation practices in HCI [Debate] • Analyze algorithmic accountability and bias in interactive systems [Analyze] • Describe professional responsibilities in HCI practice [Describe] • Explain legal and regulatory considerations relevant to interactive systems [Explain]
Readings : [**Shneiderman2022**], [**Norman2013**]	

Unit 3: Accessibility and Inclusive Design (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Accessibility principles and guidelines • Disabilities and assistive technologies • Inclusive design and universal design principles • Evaluation methods for accessibility • Designing for diverse abilities and contexts • Accessibility standards and compliance	• Apply accessibility principles to interactive system design [Apply] • Describe how different disabilities affect interaction with technology [Describe] • Design inclusive systems using universal design principles [Design] • Conduct accessibility evaluations of interactive systems [Evaluate] • Create designs that accommodate diverse abilities and usage contexts [Create] • Ensure compliance with accessibility standards and regulations [Evaluate]
Readings : [**Johnson2020**]	

Unit 4: Evaluating the Design (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Usability principles and heuristics • Evaluation methods: heuristic evaluation, cognitive walkthrough • User testing: planning, conducting, analyzing • Advanced evaluation methods: controlled experiments, field studies • UX metrics and measurement • Analytical evaluation methods: modeling and simulation	• Apply usability principles and heuristics to evaluate designs [Apply] • Conduct heuristic evaluations and cognitive walkthroughs [Evaluate] • Plan, conduct, and analyze user testing sessions [Plan] • Design and conduct controlled experiments for HCI research [Design] • Measure user experience using appropriate metrics [Evaluate] • Use analytical methods to model and predict user performance [Use]
Readings : [**Norman2013**], [**Johnson2020**]	

Unit 5: System Design (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Design process: user-centered design, iterative design • Interaction design: concepts, principles, patterns • Prototyping techniques: low-fidelity, high-fidelity, interactive • Information architecture and navigation design • Advanced interaction techniques: gesture, voice, multimodal • Design for specific domains: mobile, web, ubiquitous computing • Collaborative and social computing systems	• Apply user-centered design processes to develop interactive systems [Apply] • Design interactions using established principles and patterns [Design] • Create prototypes at different fidelity levels for design exploration [Create] • Structure information and navigation for effective user interaction [Structure] • Implement advanced interaction techniques such as gesture or voice interfaces [Implement] • Design interactive systems for specific domains and platforms [Design] • Design collaborative systems that support group interaction [Design]
Readings : [**Johnson2020**], [**Norman2013**]	

Unit 6: Society, Ethics, and the Profession (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Social impact of interactive technologies • Ethical issues in HCI research and practice • Professional practice in HCI • Cultural and cross-cultural considerations in HCI • Sustainability and HCI • Future trends and challenges in HCI	• Analyze the social impact of interactive technologies on individuals and society [Analyze] • Address ethical issues in HCI research and design projects [Evaluate] • Describe professional practices and career paths in HCI [Describe] • Design systems that account for cultural differences and contexts [Design] • Incorporate sustainability considerations into HCI design [Integrate] • Discuss emerging trends and future challenges in HCI [Debate]
Readings : [**Shneiderman2022**], [**Norman2013**]	

Unit 7: Agents and Cognitive Systems (8 hours)

Competences Expected: AG-C03,AG-C09

Topics	**Learning Outcomes**
• Agent architectures (e.g., reactive, layered, cognitive) • Agent theory (including mathematical formalisms) • Rationality, Game Theory: – Decision-theoretic agents – Markov decision processes (MDP) – Bandit algorithms • Software agents, personal assistants, and information access: – Collaborative agents – Information-gathering agents – Believable agents (synthetic characters, modeling emotions in agents) • Learning agents • Cognitive systems: – Cognitive architectures (e.g., ACT-R, SOAR, ICARUS, FORR) – Capabilities (e.g., perception, decision making, prediction, knowledge maintenance) – Knowledge representation, organization, utilization, acquisition, and refinement – Applications and evaluation of cognitive systems • Multi-agent systems: – Collaborating agents – Agent teams – Competitive agents (e.g., auctions, voting) – Swarm systems and biologically inspired models – Multi-agent learning • Human-agent interaction: – Communication methodologies (verbal and non-verbal) – Practical issues – Applications: ∗ Trading agents, supply chain management ∗ Ethical issues of AI interactions with humans ∗ Regulation and legal requirements of AI systems for interacting with humans	• Characterize and contrast the standard agent architectures [Evaluate] • Describe the applications of agent theory to domains such as software agents, personal assistants, and believable agents, and discuss associated ethical implications [Describe] • Describe the primary paradigms used by learning agents [Describe] • Demonstrate using appropriate examples how multi-agent systems support agent interaction [Demonstrate] • Construct an intelligent agent using a well-established cognitive architecture (ACT-R, SOAR) for solving a specific problem [Apply]

Readings : [**RussellNorvig2021**], [**Shneiderman2022**]

Unit 8: Natural Language Processing (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Deterministic and stochastic grammars • Parsing algorithms: – CFGs and chart parsers (e.g., CYK) – Probabilistic CFGs and weighted CYK • Representing meaning/Semantics: – Logic-based knowledge representations – Semantic roles – Temporal representations – Beliefs, desires, and intentions • Corpus-based methods • N-grams and HMMs • Smoothing and backoff • Examples of use: POS tagging and morphology • Information retrieval: – Vector space model: ∗ TF & IDF – Precision and recall • Information extraction • Language translation • Text classification, categorization: – Bag of words model • Deep learning for NLP: – RNNs – Transformers – Multi-modal embeddings (e.g., images + text) – Generative language models	• Define and contrast deterministic and stochastic grammars, providing examples to show the adequacy of each [Define] • Simulate, apply, or implement classic and stochastic algorithms for parsing natural language [Apply] • Identify the challenges of representing meaning [Analyze] • List the advantages of using standard corpora. Identify examples of current corpora for a variety of NLP tasks [List] • Identify techniques for information retrieval, language translation, and text classification [Analyze] • Implement a TF/IDF transform, use it to extract features from a corpus, and train an off-the-shelf machine learning algorithm using those features to do text classification [Apply]
Readings : [**JurafskyMartin2024**], [**RussellNorvig2021**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI264. Deep Learning (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	AI264. Deep Learning
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	AI263. Introduction to Machine Learning. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course covers modern deep learning fundamentals, including convolutional networks, transformer architectures, and training techniques for advanced AI systems.

5. GOALS

- Implement deep neural networks using modern frameworks (PyTorch/TensorFlow)
- Design architectures for specific domains (vision, language, etc.)
- Analyze ethical limitations of deep models

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

7. TOPICS

Unit 1: Neural Network Training and Optimization (10 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Modern backpropagation: computational graphs and automatic differentiation • Advanced optimizers: momentum, RMSProp, Adam, and learning-rate schedules	• Implement a deep neural network from scratch, including its backpropagation pass [Apply] • Select and tune an optimizer and its learning-rate schedule for training stability [Apply]
Readings : [**goodfellow2016dl**]	

Unit 2: Regularization Techniques for Deep Networks (5 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Dropout regularization • Batch normalization	• Diagnose overfitting in a training run using regularization diagnostics [Analyze] • Apply dropout and batch normalization to improve a model's generalization [Apply]
Readings : [**goodfellow2016dl**]	

Unit 3: Convolutional and Residual Network Architectures (9 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Deep convolutional neural network architectures • Residual connections and ResNets	• Design a deep CNN architecture for a given vision task [Apply] • Adapt a residual architecture to a specific vision domain [Apply]
Readings : [**he2016resnet**]	

Unit 4: Transformer Architectures and Generative Models (16 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Transformers and the attention mechanism • Generative models: variational autoencoders (VAEs) and generative adversarial networks (GANs)	• Train a transformer model on a real dataset [Apply] • Analyze and contrast VAE and GAN generative-model architectures [Analyze]
Readings : [**vaswani2017attention**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

FG211. Professional Ethics (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	FG211. Professional Ethics
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	1 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course adapts professional ethical principles to Artificial Intelligence, integrating ACM/IEEE codes with AI-specific challenges. Combines regulatory frameworks (GDPR) with current societal impact cases.

5. GOALS

- Apply ACM/IEEE ethics codes to AI problems.
- Analyze compliance with regulations (GDPR) in autonomous systems.
- Design ethical protocols for AI projects.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Ethical Foundations in AI (10 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Comparison: ACM vs IEEE ethics codes for AI. • Algorithmic transparency and fairness principles.	• Explain differences between ethical frameworks [Familiarizarse]. • Evaluate compliance in real cases [Evaluar].
Readings : [**ACM-Ethics2018**], [**IEEE-Ethics2020**]	

Unit 2: Privacy and Regulations (14 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• GDPR applied to ML: right to explanation. • Ethical auditing of datasets (e.g., ImageNet).	• Implement GDPR checklists for models [Usar]. • Identify violations in case studies [Evaluar].
Readings : [**GDPR2018**], [**AI-Ethics2021**]	

Unit 3: Bias and Fairness (18 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Fairness metrics (demographic parity, equality of opportunity). • Tools: IBM Fairness 360, Google What-If.	• Measure biases in models using Python [Usar]. • Propose mitigation strategies [Evaluar].
Readings : [**ACM-Cases2020**], [**Mehrabi2021**]	

Unit 4: Accountability in Autonomous Systems (18 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Legal attribution in AI failures (e.g., self-driving cars). • ISO standards for trustworthy AI.	• Draft liability clauses [Evaluar]. • Analyze legal conflicts [Familiarizarse].
Readings : [**Bostrom2014**], [**IEEE-Cases2019**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

FG350. Leadership and Performance (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	FG350. Leadership and Performance
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	2 HT;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	FG106. Theater. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Contemporary organizations demand from their members the effective exercise of leadership: to assume with effectiveness and a spirit of service the challenges entrusted to them, in the pursuit of a more just and reconciled society. This challenge involves forming people with a sound knowledge of themselves, with the ability to judge reality objectively and to propose guidelines that positively modify the environment. The course combines theoretical foundations with the experiential development of competencies, training the student to lead teams with integrity, personal coherence, and genuine service to others.

5. GOALS

- Understand the theoretical foundations of leadership and its ethical dimension, integrating them into a process of self-knowledge.
- Identify and develop the personal competencies and virtues necessary for the effective exercise of leadership.
- Acquire practical skills for the management, motivation, and direction of high-performance teams.
- Contribute to the construction of a more just society through leadership with a sense of service and social responsibility.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

7. TOPICS

Unit 1: Foundations of Leadership (15 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Definition and concept of leadership. • Main leadership theories: transactional, transformational, and situational. • Integral vision of the human being: bodily, affective, and spiritual dimensions. • Motives of human action: extrinsic, intrinsic, and transcendent. • The practice of virtue in the exercise of leadership. • The leader as a model: ethics and personal coherence. • Leadership styles and their application in different contexts.	• Analyze and understand the theoretical bases of the exercise of leadership [Familiarizarse]. • Assume the correct attitude to put leadership into practice [Familiarizarse]. • Initiate a process of self-knowledge aimed at discovering leadership traits in oneself [Familiarizarse]. • Recognize the ethical dimension of leadership and its impact on the personal and social environment [Familiarizarse].
Readings : [**Cardona**], [**Ferreiro**], [**Dianine**], [**DSouza**]	

Unit 2: Formation of Virtues and Competencies (15 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• Theory of competencies and its application to leadership. • Recognition and inventory of personal competencies. • Personal development plan: diagnosis, goals, and follow-up. • Mental models and their influence on behavior. • Needs and emotional profiles. • Motivational vices: identification and overcoming. • Emotional intelligence applied to leadership. • Decision-making under pressure and ambiguity.	• Know and develop leadership competencies focused on effectiveness and service [Familiarizarse]. • Recognize the personal and group tendencies necessary for the exercise of leadership [Familiarizarse]. • Prepare a personal development plan for leadership competencies [Usar].
Readings : [**Wilkinson**], [**Huete**], [**Cardona**], [**Chinchilla**], [**Goleman**]	

Unit 3: Management and Direction of Teams (18 hours)	
Competences Expected: AG-C02,AG-C04	
Topics	**Learning Outcomes**
• The personal relationship with the team: trust and communication. • Integral leadership: mission, vision, and shared values. • Delegation and accompaniment: the art of motivating and delegating. • Coaching and mentoring as development tools. • Conflict resolution and group dynamics. • The five dysfunctions of a team and how to overcome them. • Foundations of unity: cohesion and organizational culture. • Servant leadership and its impact on the organization.	• Develop skills for effective and cohesive teamwork [Familiarizarse]. • Apply situational leadership techniques to lead teams in different contexts [Usar]. • Design strategies for conflict resolution and improvement of group performance [Usar].
Readings : [**Goleman**], [**CardonaP**], [**Hersey**], [**Hunsaker**], [**Hawkins**], [**Lencioni**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS281. Computing in Society (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS281. Computing in Society
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	2 HT;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course explores the multifaceted relationship between computing technologies and society. It provides a historical perspective on the evolution of the field while equipping students with ethical analysis frameworks to navigate contemporary challenges. Topics include intellectual property, privacy, professional responsibility, sustainability, and the socio-economic impact of automation. Students will develop the communication skills and ethical sensitivity required for professional practice in a diverse and globalized environment.

5. GOALS

- Apply ethical analysis to professional dilemmas in computing.
- Understand the legal frameworks for Intellectual Property and Privacy.
- Evaluate the environmental sustainability and social impact of IT.
- Communicate effectively regarding professional and ethical issues.

6. COMPETENCES

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Usage)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

7. TOPICS

Unit 1: Foundations and Early History of Computing (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• The history of computing: hardware, software, and human/organizational. • The role of history in the present including within different social contexts, and the relevance of this history on the future. • Age I (Pre-digital): Ancient analog computing (Stonehenge, Antikythera mechanism, Salisbury Cathedral clock, etc.), human-calculated number tables, Euclid, Lovelace, Babbage, Gödel, Church, Turing, pre-electronic (electro-mechanical and mechanical) hardware. • Age II (Early modern computing): ENIAC, UNIVAC, Bombes (Bletchley Park and codebreakers), computer companies (e.g., IBM), mainframes, etc.	• Understand the relevance and impact of computing history on recent events, present context, and possible future outcomes, from more than one cultural perspective [Assess] • Discuss how perspectives held today have been shaped by history, and that alternative perspectives exist (e.g., fears of AI replacing human workers vs AI augmenting human work, various views on Moore's Law) [Debate] • Identify formative and consequential trends in the history of the computing field [Analyze] • Identify the contributions of pioneering individuals or organizations (research labs, computer companies, government offices) in the computing field [Analyze]
Readings : [**Quinn2019**], [**Baase2017**]	

Unit 2: Social Impact and AI (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Social implications (e.g., political and cultural ideologies) in a hyper-networked world where the capabilities and impact of social media, artificial intelligence, and computing in general are rapidly evolving. • Impact of computing applications (e.g., social media, artificial intelligence applications) on individual well-being, and safety of all kinds (e.g., physical, emotional, economic). • Consequences of involving computing technologies, particularly artificial intelligence, biometric technologies, and algorithmic decision-making systems, in civic life (e.g., facial recognition technology, biometric tags, resource distribution algorithms, policing software) and how human agency and oversight is crucial. • Growth and control of the internet, data, computing, and artificial intelligence.	• Describe the different ways that computer technology (networks, mobile computing, artificial intelligence) mediates social interaction at the personal and collective levels [Describe] • Interpret the social context of a given design and its implementation [Interpret] • Analyze the efficacy of a given design and implementation using empirical data [Analyze] • Understand the implications of technology use (e.g., social media) for different identities, cultures, and communities [Explain] • Describe the internet's role in facilitating communication between citizens, governments, and each other [Describe] • Analyze the effects of reliance on computing in the implementation of democracy (e.g., delivery of social services, electronic voting) [Analyze] • Describe the impact of a lack of appropriate representation of people from historically minoritized populations in the computing profession (e.g., industry culture, product diversity) [Describe]
Readings : [**Quinn2019**]	

Unit 3: Ethical Foundations and Frameworks (4 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Avoiding fallacies and misrepresentation in argumentation. • Ethical theories and decision-making (philosophical and social frameworks, e.g. [1]). • Recognition of the role culture plays in our understanding, adoption, design, and use of computing technology. • Why ethics is important in computing, and how ethics is similar to, and different from, laws and social norms. • Introduction to ethical frameworks (e.g., consequentialism such as utilitarianism, non-consequentialism such as duty, rights, or justice, agent-centered such as virtue or feminism, contractarianism, ethics of care) and their use for analyzing an ethical dilemma.	• Describe how a given cultural context impacts decision making [Describe] • Express the use of example and analogy in ethical argument [Analyze] • Analyze (and avoid) basic logical fallacies in an argument [Analyze] • Analyze an argument to identify premises and conclusion [Analyze] • Evaluate how and why ethics is so important in computing and how it relates to cultural norms, values, and law [Evaluate] • Justify a decision made on ethical grounds [Justify] • Discuss the advantages and disadvantages of traditional ethical frameworks [Debate] • Analyze ethical dilemmas related to the creation and use of technology from multiple perspectives using ethical frameworks [Analyze]
Readings : [**Quinn2019**], [**Baase2017**]	

Unit 4: Professional Values, Nature and Continuing Development (4 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Community values and the laws by which we live. • The nature of being a professional including care, attention, discipline, fiduciary responsibility, and mentoring. • Keeping up to date as a computing professional in terms of familiarity, tools, skills, legal and professional frameworks as well as the ability and responsibility to self-assess and progress in the computing field.	• Identify ethical issues that arise in software design, development practices, and software deployment [Analyze] • Discuss how to address ethical issues in specific situations [Debate] • Describe the mechanisms that typically exist for a professional to keep up to date in ethical matters [Describe]
Readings : [**Baase2017**]	

Unit 5: Intellectual Property Foundations (4 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Intellectual property rights. • Legal foundations for intellectual property protection. • Philosophical foundations of intellectual property. • Forms of intellectual property (e.g., copyrights, patents, trade secrets, trademarks) and the rights they protect.	• Defend legal and ethical uses of copyrighted materials [Defend] • Discuss the philosophical bases of intellectual property in an appropriate context (e.g., country) [Debate] • Contrast the protections and obligations of copyright, patent, trade secret, and trademarks [Contrast] • Describe the rationale for the legal protection of intellectual property in the appropriate context (e.g., country) [Describe]
Readings : [**Quinn2019**]	

Unit 6: Digital IP Rights and Copyright (6 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Intangible digital intellectual property (IDIP). • Limitations on copyright protections, including fair use and the first sale doctrine. • Intellectual property laws and treaties that impact the enforcement of copyrights.	• Describe and critique legislation and precedent aimed at digital copyright infringements [Describe] • Identify contemporary examples of intangible digital intellectual property [Analyze] • Discuss whether a use of copyrighted material is likely to be fair use [Debate] • Analyze the use of copyrighted work under the concepts of fair use and the first sale doctrine [Analyze] • Discuss the nature of anti-circumvention laws in the context of copyright protection [Debate]
Readings : [**Tavani2013**]	

Unit 7: Data, Surveillance and Privacy Technology (4 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Privacy implications of widespread data collection including but not limited to transactional databases, data warehouses, surveillance systems, cloud computing, and artificial intelligence. • Conceptions of anonymity, pseudonymity, and identity. • Technology-based solutions for privacy protection (e.g., end-to-end encryption and differential privacy).	• Evaluate solutions to privacy threats in transactional databases and data warehouses [Evaluate] • Describe the role of data collection in the implementation of pervasive surveillance systems (e.g., RFID, face recognition, toll collection, mobile computing) [Describe] • Distinguish the concepts and goals of anonymity and pseudonymity [Distinguish] • Describe the ramifications of technology-based privacy protections, including differential privacy and end-to-end encryption [Describe]
Readings : [**Baase2017**]	

Unit 8: Environmental Footprint of Computing Systems (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Environmental, social, and cultural impacts of implementation decisions (e.g., sustainability goals, algorithmic bias/outcomes, economic viability, and resource consumption). • Local/regional/global social and environmental impacts of computing systems and their use (e.g., carbon footprints, resource usage, e-waste) due to hardware (e.g., e-waste, data centers, rare element and resource utilization, recycling) and software (e.g., cloud-based services, blockchain, AI model training and use). This includes everyday use of hardware (cheap hardware replaced frequently) and software (web-browsing, email, and other services with hidden/remote computational demands).	• Assess the environmental impacts of a given project's deployment (e.g., energy consumption, contribution to e-waste, impact of manufacturing) [Assess] • Describe global social and environmental impacts of computer use and disposal [Describe] • Describe the environmental impacts of design choices within the field of computing that relate to algorithm design, operating system design, networking design, database design, etc [Describe] • Assess computing applications in respect to environmental issues (e.g., energy, pollution, resource usage, recycling and reuse, food management and production) [Assess]
Readings : [**Quinn2019**]	

Unit 9: Economic Models and Strategies (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Economic models: regulated and unregulated, monopolies, network effects, and open market; knowledge and attention economies. • Pricing and deployment strategies: planned obsolescence, subscriptions, freemium, software licensing, open-source, free software, adware. • Impacts of differences in access to computing resources, and the effect of skilled labor supply and demand on the quality of computing products.	• Summarize the social effects of economic models (e.g., the knowledge and attention economies) [Summarize] • Describe the differences and similarities of competing strategies (e.g., subscription vs freemium vs free) [Describe] • Discuss examples of digital divides [Debate]
Readings : [**Baase2017**]	

Unit 10: Computer Crimes and Attacks (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Computer crimes, legal redress for computer criminals and impact on victims and society. • Social engineering, computing-enabled fraud, identity theft and recovery from these. • Cyber terrorism, criminal hacking, and hacktivism. • Malware, viruses, worms. • Attacks on critical infrastructure such as electrical grids and pipelines.	• List classic examples of computer crimes and social engineering incidents with societal impact [List] • Identify issues with laws that apply to computer crimes [Analyze] • Describe the motivation and ramifications of cyber terrorism, data theft, hacktivism, ransomware, and other attacks [Describe] • Examine the ethical and legal issues surrounding the misuse of access and various breaches of security [Examine]
Readings : [**Baase2017**]	

Unit 11: Identity and Diversity in Computing (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• How identity impacts and is impacted by computing technologies and environments (academic and professional). • The benefits of diverse development teams and the impacts of teams that are not diverse.	• Define and distinguish equity, equality, diversity, inclusion, and accessibility [Define] • Identify examples of the benefits that diverse teams can bring to software products, and how a lack of diversity has costs [Analyze] • Assess the impact of power and privilege in the computing profession as it relates to culture, industry, products, and society [Assess]
Readings : [**Quinn2019**]	

Unit 12: Technical and Professional Communication (2 hours)	
Competences Expected: AG-C01,AG-C02	
Topics	**Learning Outcomes**
• Oral, written, and electronic team and group communication. • Technical communication materials (e.g., source code, and documentation, tutorials, reference materials, API documentation). • Communicating with different stakeholders such as customers, leadership, or the public. • Tradeoffs in competing factors that affect communication channels and choices.	• Understand the importance of writing concise and accurate technical documents following well-defined standards for format and for including appropriate tables, figures, and references [Produce] • Analyze written technical documentation for technical accuracy, concision, lack of ambiguity, and awareness of audience [Analyze] • Compose and deliver an audience-aware, accessible, and organized formal presentation [Compose] • Identify and describe qualities of effective communication (e.g., virtual, face-to-face, intragroup, shared documents) [Analyze] • Choose an appropriate way to communicate delicate ethical concerns [Assess]
Readings : [**Quinn2019**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS3I1. Computer Security (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3I1. Computer Security
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS231. Networking and Communication. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

In an increasingly digital world, security is no longer an optional feature but a fundamental requirement. This course provides a comprehensive introduction to the principles and practices of cybersecurity, covering everything from theoretical foundations to secure coding and digital forensics. Students will gain the skills necessary to design, implement, and govern secure systems while understanding the ethical and professional responsibilities involved in protecting information and infrastructure.

5. GOALS

- Understand the fundamental principles and mathematical bases of security.
- Apply secure coding techniques and cryptographic methods to software.
- Analyze system vulnerabilities and perform forensic investigations.
- Evaluate security governance and professional ethics in cybersecurity.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Security Mindset and Principles (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Developing a security mindset incorporating cross-cutting concepts: confidentiality, integrity, availability, risk assessment, systems thinking, adversarial thinking, human-centered thinking. • Vulnerabilities, threats, attack surfaces, and attack vectors Protection and Safety. • Denial of Service (DoS) and Distributed Denial of Service (DDoS) Protection and Safety. • Principles and practices of protection, e.g., least privilege, open design, fail-safe defaults, defense in depth, and zero trust; and how they can be implemented Principles of Operating System, Protection and Safety, Coding Practices, Large-Scale Construction and Process, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Optimization considerations between security, privacy, performance, and other design goals System Design, Software Development Practices, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. • Concepts of trust and trustworthiness.	• Evaluate a system for possible attacks that can be launched by an adversary [Evaluate] • Design and develop approaches to protect a system from a set of identified threats [Design] • Describe how harm to user privacy can be avoided [Describe] • Develop a system that incorporates various principles of security and privacy [Create]
Readings : [**Pfleeger2015**], [**Stallings2017**]	

Unit 2: Cryptography Fundamentals (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Differences between algorithmic, applied, and mathematical views of cryptography. • Mathematical preliminaries: modular arithmetic, Euclidean algorithm, probabilistic independence, linear algebra basics, number theory, finite fields, complexity, asymptotic analysis Discrete Mathematics, Linear Algebra. • Basic cryptography: symmetric key and public key cryptography Foundational Data Structures, Foundational Algorithms, Advanced Algorithms, Discrete Mathematics. • Classical cryptosystems, such as shift, substitution, transposition ciphers, code books, and machines Discrete Mathematics. • Kerckhoff's principle and use of vetted libraries Coding Practices, Large-Scale Construction and Process.	• Explain the role of cryptography in supporting security and privacy [Explain] • Discuss the risks of inventing one's own cryptographic methods [Debate] • Discuss the importance of prime numbers in cryptography and explain their use in cryptographic algorithms [Debate] • Implement and cryptanalyze classical ciphers [Implement]
Readings : [**Stallings2017**]	

Unit 3: Injection and Input Validation (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Common vulnerabilities and weaknesses. • SQL injection and other injection attacks. • Cross-site scripting techniques and mitigations. • Input validation and data sanitization Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.	• Identify underlying problems in given examples of an enumeration of common weaknesses and explain how they can be circumvented [Analyze] • Apply input validation and data sanitization techniques to enhance security of a program [Apply] • Evaluate and prevent SQL injections attacks on a database application [Evaluate] • Evaluate and prevent cross-site scripting attacks against a website [Evaluate] • Construct a program that tests for all input handling errors [Create]
Readings : [**Pfleeger2015**]	

Unit 4: Malware Analysis and Advanced Security (6 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Offensive security techniques as a defense. • AI-assisted malware detection techniques. • Ransomware: creation, prevention, and mitigation. • Secure use of third-party components Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. • Malware: varieties, creation, reverse engineering, and defense against them Language Translation and Execution, Systems Execution and Memory Model. • Assurance: testing (including fuzzing and penetration testing), verification, and validation Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. • Static and dynamic analyses Program Analysis and Analyzers, Protection and Safety, Evaluation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. • Secure compilers and secure code generation Language Translation and Execution, Run-time Behavior and Systems.	• Describe different kinds of malicious software [Describe] • Explain the risks of misusing interfaces with third-party code and how to correctly use third-party code [Explain] • Discuss the need to update software to fix security vulnerabilities and the lifecycle management of the fix [Debate] • Apply static and dynamic tools to identify programming faults [Apply] • Evaluate a system for the existence of malware and remove it [Evaluate] • Implement preventive techniques to reduce the occurrence of ransomware [Implement]
Readings : [**SikorskiHonig2012**], [**Eilam2005**], [**NIST800-53**]	

Unit 5: Threat Analysis and Security Engineering (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Security engineering goals: building systems that remain dependable despite errors, accidents, or malicious adversaries Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Problem analysis and situational analysis to address system security Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. • Security analysis, covering security requirements analysis; security controls analysis; threat analysis; and vulnerability analysis Program Analysis and Analyzers, Evaluation. • Security attack domains and attack surfaces, e.g., communications and networking, hardware, physical, social engineering, software, and supply chain Network Security. • Security attack modes, techniques, and tactics, e.g., authentication abuse; brute force; buffer manipulation; code injection; content insertion; denial of service; eavesdropping; function bypass; impersonation; integrity attack; interception; phishing; protocol analysis; privilege abuse; spoofing; and traffic injection Network Security, Protection and Safety, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.	• Create a threat model for a system or system design [Create] • Apply situational analysis to develop secure solutions under a specified scenario [Apply] • Identify and mitigate security vulnerabilities and weaknesses in a system [Analyze]
Readings : [**Anderson2020**]	

Unit 6: Digital Forensics Techniques (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Basic principles and methodologies for digital forensics. • System design for forensics. • Forensics in different situations: operating systems, file systems, application forensics, web forensics, network forensics, mobile device forensics, use of database auditing Network Security. • Attacks on forensics and preventing such attacks.	• Explain what a digital investigation is and how it can be implemented Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Explain] • Design and implement software to support forensics [Design] • Extract data from a hard drive to comply with the law Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Extract] • Recover data based on a given search term from an imaged system [Analyze] • Reconstruct data and events from an application history, or a web artifact, or a cloud database, or a mobile device. Mobile Platforms: Capabilities and Asynchronous Computing, Mobile Platforms: Architecture and Constraints, Mobile Platforms: Development Fundamentals, Cloud Architecture and Data Management, Frontend, Frameworks and Web Standards [Apply] • Capture and analyze network traffic. Network Security [Analyze] • Develop approaches to address the challenges associated with mobile device forensics [Create] • Apply forensics tools to investigate security breaches [Apply] • Identify and mitigate anti-forensic methods [Analyze]
Readings : [**Pfleeger2015**]	

Unit 7: Security Governance, Management and Policy (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Protecting critical assets from threats. • Security governance: organizational objectives and general risk assessment. • Security management: achieve and maintain appropriate levels of confidentiality, integrity, availability, accountability, authenticity, and reliability Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. • Security policy: organizational policies, issue-specific policies, system-specific policies.	• Describe critical assets and how they can be protected [Describe] • Differentiate between security governance, management, and controls, giving examples of each [Differentiate] • Describe a technical control and implement it to mitigate specific threats [Describe]
Readings : [**Anderson2020**]	

Unit 8: Security Ethics, Law and Society (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Societal impacts on breakdowns in security and privacy Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Applicability of laws and regulations on security and privacy Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Professional ethical considerations when designing secure systems and maintaining privacy; ethical hacking Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Security and privacy implications of malicious AI/machine learning actors, e.g., identifying deep fakes Machine Learning Formulation and Evaluation, Deep Learning, RL and Ethics, Introduction to AI: Concepts and Intelligent Behavior, Problem Characteristics and Agent Nature, Machine Learning Fundamentals, Machine Learning Data, Representations and Evaluation, Neural Networks and Ethics, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Societal impacts of Internet of Things (IoT) devices and other emerging technologies on security and privacy Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.	• Calculate the impact of a breakdown in security of a given system [Calculate] • Construct a system that conforms to security laws [Create] • Apply a set of privacy regulations to design a system that protects privacy [Apply] • Evaluate the legal ramifications of a system not corresponding to applicable laws and regulations [Evaluate]
Readings : [**Anderson2020**], [**Pfleeger2015**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through

problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS3P1. Parallel and Distributed Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3P1. Parallel and Distributed Computing
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	• CS212. Analysis and Design of Algorithms. (5^{th} Sem) • CS231. Networking and Communication. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

With the end of frequency scaling in single-core processors, parallel and distributed computing has become essential for high-performance software development. This course introduces students to the programming models, communication patterns, and coordination mechanisms required to exploit multi-core systems and distributed clusters. Students will learn to design scalable algorithms and evaluate their performance under different computational constraints.

5. GOALS

- Design and implement parallel programs using shared and distributed memory.
- Master communication and synchronization protocols in distributed systems.
- Evaluate performance and scalability through formal metrics.
- Apply parallel algorithms to solve complex computational problems.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Programs: Declarative Parallelism (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Parallelism – Declarative parallelism: Determining which actions can, or must not, be performed in parallel, at the level of instructions, functions, closures, compound actions, sessions, tasks, and services is the core idea underlying PDC algorithms; failing to do so is the primary source of errors. See also: Algorithms – Defining order: for example, using "happens-before" relationships or series-parallel directed acyclic graphs (DAGs) representing programs. – Independence: determining when order does not matter, in terms of commutativity, dependencies, preconditions. – Guaranteeing order between otherwise parallel actions when necessary, including locks, safe publication; and enforcing communication - sending a message happens before receiving it; and relaxing it when not necessary.	• Graphically represent (as a Directed Acyclic Graph - DAG) how to parallelize a compound numeric expression; for example, $a = (b + c) * (d + e)$ [Design] • Explain why consistency and fault tolerance concepts do not arise in purely sequential programs [Explain]
Readings : [**Pacheco2021**]	

Unit 2: Programs: Starting Activities (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Starting activities – Options allowing actions to be performed (eventually) at locations range from hardwired to configuration scripts; also establishing communication and resource management; these are expressed differently in languages and contexts, typically relying on automated provisioning and management by platforms. See also: Resource Management – Procedural: Allowing multiple actions to start at a given program point; for example, starting new threads, possibly delimiting their scope or organizing them into hierarchical groups. – Reactive: Enabling upon event occurrence by installing an event handler, with less control over when actions start or finish, and may apply even on single processors. – Dependent: Enabling upon completion of others; for example, sequencing sets of parallel actions. See also: Coordination. – Granularity: The execution cost of action bodies should outweigh the overhead of organizing them.	• Write a service that creates a thread (or other form of procedural activation) to return a requested web page to each new client [Write]
Readings : [**Pacheco2021**]	

Unit 3: Programs: Execution Properties (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Execution Properties – Non-deterministic execution of actions without order. – Consistency: Ensuring agreement among parties on values and predicates when necessary to avoid races, maintain safety and atomicity, or reach consensus. – Fault tolerance: Handling failures in parts or communication, including (Byzantine) misbehavior due to unreliable parties and protocols, when necessary to maintain progress or availability. See also: System Reliability. – Trade-offs are a focus of evaluation. See also: Evaluation.	• Write a function that counts events, such as network packet receptions, efficiently [Write]
Readings : [**VanSteen2023**]	

Unit 4: Programs: Distribution (2 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Distribution – Defining locations as devices that execute actions, including hardware components, remote hosts; may also include external uncontrolled devices, hosts, and users. See also: Interfacing and Communication – A device may time-share or emulate multiple parallel actions using fewer processors through scheduling and virtualization. See also: Scheduling – Naming or identifying locations (e.g., device IDs) and actions as parts (e.g., thread IDs). – Activities across locations may communicate through media. See also: Communication	• Write a filter/map/reduce program in multiple styles [Write]
Readings : [**VanSteen2023**]	

Unit 5: Programs: Implementation Mappings (2 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• One or more of the following mappings and mechanisms across layered systems: – Instruction-level and data parallelism in the CPU. See also: Functional Organization. – SIMD and heterogeneous data parallelism. See also: Heterogeneous Architectures. – Scheduled concurrency on multi-core, tasks, actors. See also: Scheduling. – Clusters, clouds; elastic provisioning. See also: Common Aspects: Platforms, APIs and Constraints, Common Aspects: Platform Languages and Web Programming Patterns. – Networked distributed systems. See also: Networked Applications. – Emerging technologies such as quantum computing and molecular computing.	• Explain the trade-offs between different mapping strategies in terms of performance, cost, and implementation complexity [Explain]
Readings : [**Pacheco2021**], [**Hwu2022**]	

Unit 6: GPU Programming (4 hours)

Competences Expected: AG-C09

Topics	**Learning Outcomes**
• General-Purpose GPU Computing (GPGPU) – GPU architecture overview: streaming multiprocessors (SMs), CUDA cores, and warp execution model. – SIMT (Single Instruction, Multiple Threads) execution model and its differences from CPU SIMD. – Host-device interaction: data transfer between CPU (host) and GPU (device) memory via PCIe. – Use cases: scientific computing, machine learning, image processing, and large-scale simulation. • CUDA Programming Model – CUDA execution hierarchy: grids, thread blocks, and individual threads; mapping to GPU hardware. – Kernel definition and launch syntax; configuration parameters (grid and block dimensions). – Thread indexing: threadIdx, blockIdx, blockDim, gridDim; computing global indices in 1D, 2D, and 3D. – Warp divergence: performance implications of branching within a warp. – Intra-block synchronization primitive: __syncthreads(). • CUDA Memory Hierarchy – Global memory: large, high-latency, accessible by all threads; coalesced access patterns for performance. – Shared memory: low-latency on-chip memory shared within a thread block; bank conflicts. – Registers and local memory: per-thread private storage. – Constant and texture memory: read-only caches optimized for specific access patterns. – Unified Memory: simplified programming model with automatic host-device data migration.	• Write a CUDA kernel that performs a data-parallel operation (e.g., vector addition, matrix multiplication) and correctly configure grid and block dimensions [Write] • Analyze the impact of global memory access patterns (coalesced vs. strided) and shared memory usage on GPU kernel performance [Analyze] • Design a parallel algorithm using the CUDA programming model, including thread hierarchy, memory allocation strategy, and host-device data transfers [Design]
Readings : [**Kirk2016**], [**CUDADocs**]	

Unit 7: Communication (8 hours)

Competences Expected: AG-C09,AG-C11

Topics	Learning Outcomes
• Media – Varieties: channels (message passing or I/O), shared memory, heterogeneous, data stores – Reliance on the availability and nature of underlying hardware, connectivity, and protocols; language support, emulation. See also: Interfacing and Communication • Channels – Explicit (usually named) party-to-party communication media – APIs: Sockets, architectural, language-based, and toolkit constructs, such as Message Passing Interface (MPI), and layered constructs such as Remote Procedure Call (RPC). See also: Fundamentals of Networks and Communications – I/O channel APIs • Memory – Shared memory architectures in which parties directly communicate only with memory at given addresses, with extensions to heterogeneous memory supporting multiple memory stores with explicit data transfer across them; for example, GPU local and shared memory, Direct Memory Access (DMA) – Memory hierarchies: Multiple layers of sharing domains, scopes, and caches; locality: latency, false-sharing – Consistency properties: Bitwise atomicity limits, coherence, local ordering • Data Stores – Cooperatively maintained data structures implementing maps and related ADTs – Varieties: Owned, shared, sharded, replicated, immutable, versioned	• Explain the similarities and differences among: (1) Party A sends a message on channel X with contents 1 received by party B (2) A sets shared variable X to 1, read by B (3) A sets "X=1" in a distributed shared map accessed by B [Explain] • Write a program that distributes different segments of a data set to multiple workers, and collects results (for the simplest example, summing segments of an array) [Write] • Write a parallel program that requests data from multiple sites and summarizes them using some form of reduction [Write] • Compare the performance of buffered versus unbuffered versions of a producer-consumer program [Compare]

Readings : [**VanSteen2023**], [**Kleppmann2017**]

Unit 8: Communication: Properties and Extensions (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• One or more of the following properties and extensions – Topologies: Unicast, Multicast, Mailboxes, Switches; Routing via hardware and software interconnection networks – Media concurrency properties: Ordering, consistency, idempotency, overlapping communication with computation – Media performance: Latency, bandwidth (throughput) contention (congestion), responsiveness (liveness), reliability (error and drop rates), protocol-based progress (acks, timeouts, mediation) – Media security properties: integrity, privacy, authentication, authorization. See also: Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security – Data formats: Marshaling, validation, encryption, compression – Channel policies: Endpoints, sessions, buffering, saturation response (waiting vs dropping), rate control – Multiplexing and demultiplexing many relatively slow I/O devices or parties; completion-based and scheduler-based techniques; async-await, select and polling APIs – Formalization and analysis of channel communication; for example, CSP – Applications of queuing theory to model and predict performance.	• Determine whether a given communication scheme provides sufficient security properties for a given usage [Determine] • Give an example of a scenario in which blocking message sends can deadlock [Create] • Describe at least one design technique for avoiding liveness failures in programs using multiple locks [Describe]
Readings : [**VanSteen2023**], [**Kleppmann2017**]	

Unit 9: Memory and Consistency (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Memory models, data consistency, and fault tolerance – Memory models: sequential and release/acquire consistency – Memory management; including reclamation of shared data; reference counts and alternatives – Bulk data placement and transfer; reducing message traffic and improving locality; overlapping data transfer and computation; impact of data layout such as array-of-structs vs struct-of-arrays – Emulating shared memory: distributed shared memory, Remote Direct Memory Access (RDMA) – Data store consistency: Atomicity, linearizability, transactionality, coherence, causal ordering, conflict resolution, eventual consistency, blockchains – Faults, partitioning, and partial failures; voting; protocols such as Paxos and Raft. – Design tradeoffs among consistency, availability, partition (fault) tolerance; impossibility of meeting all at once – Security and trust: Byzantine failures, proof of work and alternatives	• Give an example of an ordering of accesses among concurrent activities (e.g., program with a data race) that is not sequentially consistent [Create] • Write a program that illustrates memory-access or message reordering [Write] • Describe the relative merits of optimistic versus conservative concurrency control under different rates of contention among updates [Describe] • Give an example of a scenario in which an attempted optimistic update may never complete [Create] • Modify a concurrent system to use a more scalable, reliable, or available data store [Create] • Using an existing platform supporting replicated data stores, write a program that maintains a key-value mapping even when one or more hosts fail [Write]
Readings : [**VanSteen2023**], [**Kleppmann2017**]	

Unit 10: Coordination (7 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Dependencies – Initiation or progress of one activity may be dependent on other activities, so as to avoid race conditions, ensure termination, or meet other requirements – Ensuring progress by avoiding dependency cycles, using monotonic conditions, removing inessential dependencies • Control constructs and design patterns – Completion-based: Barriers, joins, including termination control – Data-enabled: Queues, producer-consumer designs – Condition-based: Polling, retrying, backoffs, helping, suspension, signaling, timeouts – Reactive: Enabling and triggering continuations	• Show how to ensure that a program correctly terminates when all of a set of concurrent tasks have completed [Design] • Write a function that efficiently counts events such as sensor inputs or networking packet receptions [Write] • Write a filter/map/reduce program in multiple styles [Write] • Write a program in which the termination of one set of parallel actions is followed by another [Write] • Write a service that creates a thread (or other procedural form of activation) to return a requested web page to each new client [Write]
Readings : [**Herlihy2020**], [**VanSteen2023**]	

Unit 11: Coordination: Synchronization and Atomicity (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Atomicity – Atomic instructions, enforced local access orderings – Locks and mutual exclusion; lock granularity – Using locks in a specific language; maintaining liveness without introducing races – Deadlock avoidance: Ordering, coarsening, randomized retries; backoffs, encapsulation via lock managers – Common errors: Failing to lock or unlock when necessary, holding locks while invoking unknown operations – Avoiding locks: replication, read-only, ownership, and non-blocking constructions	• Show how to avoid or repair a race error in a given program [Analyze]
Readings : [**Herlihy2020**]	

Unit 12: Coordination: Advanced Properties (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• One or more of the following properties and extensions – Progress properties including lock-free, wait-free, fairness, priority scheduling, interactions with consistency, reliability – Performance with respect to contention, granularity, convoying, scaling – Non-blocking data structures and algorithms – Ownership and resource control – Lock variants and alternatives: sequence locks, read-write locks; Read-Copy-Update (RCU), reentrancy; tickets; controlling spinning versus blocking – Transaction-based control: Optimistic and conservative – Distributed locking: reliability – Alternatives to barriers: Clocks; counters, virtual clocks; dataflow and continuations; futures and RPC; consensus-based, gathering results with reducers and collectors – Speculation, selection, cancellation; observability and security consequences – Resource control using semaphores and condition variables – Control flow: Scheduling computations, series-parallel loops with (possibly elected) leaders, pipelines and streams, nested parallelism – Exceptions and failures. Handlers, detection, timeouts, fault tolerance, voting	• Write a program that speculatively searches for a solution by multiple activities, terminating others when one is found [Write] • Write a program in which a numerical exception (such as divide by zero) in one activity causes termination of others [Write] • Write a program for multiple parties to agree upon the current time of day; discuss its limitations compared to protocols such as network transfer protocol (NTP) [Write]
Readings : [**Herlihy2020**]	

Unit 13: Evaluation (7 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Safety and liveness requirements in terms of temporal logic constructs to express "always" and "eventually" See also: Parallel and Distributed Computing • Identifying, testing for, and repairing violations, including common forms of errors such as failure to ensure necessary ordering (race errors), atomicity (including check-then-act errors), and termination (livelock) • Performance requirements metrics for throughput, responsiveness, latency, availability, energy consumption, scalability, resource usage, communication costs, waiting and rate control, fairness; service level agreements. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Performance impact of design and implementation choices, including granularity, overhead, consensus costs, and energy consumption. See also: Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context • Estimating scalability limitations, for example using Amdahl's Law or Universal Scalability Law. See also: Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models	• Revise a specification to enable parallelism and distribution without violating other essential properties or features [Redesign] • Explain how concurrent notions of safety and liveness extend their sequential counterparts [Explain] • Specify a set of invariants that must hold at each bulk-parallel step of a computation [Analyze] • Write a test program that can reveal a data race error; for example, missing an update when two activities both try to increment a variable [Write] • In a given context, explain the extent to which introducing parallelism in an otherwise sequential program would be expected to improve throughput and/or reduce latency, and how it may impact energy efficiency [Explain] • Show how scaling and efficiency change for sample problems without and with the assumption of problem size changing with the number of processors; further explain whether and how scalability would change under relaxations of sequential dependencies [Design]
Readings : [**Sterling2024**], [**Pacheco2021**]	

Unit 14: Evaluation: Formal Methods (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Formal verification and analysis methods – Extensions to formal sequential requirements such as linearizability – Protocol, session, and transactional specifications – Use of tools such as Unified Modelling Language (UML), Temporal Logic of Actions (TLA), program logics – Security analysis: safety and liveness in the presence of hostile or buggy behaviors by other parties; required properties of communication mechanisms (for example lack of cross-layer leakage), input screening, rate limiting. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles – Static analysis applied to correctness, throughput, latency, resources, energy. See also: Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context – Directed Acyclic Graph (DAG) model analysis of algorithmic efficiency (work, span, critical paths)	• Specify and measure behavior when a service is requested by unexpectedly many clients [Analyze] • Identify and repair a performance problem due to sequential bottlenecks [Analyze] • Empirically compare throughput of two implementations of a common design (perhaps using an existing test harness framework) [Compare]
Readings : [**Sterling2024**], [**Pacheco2021**]	

Unit 15: Evaluation: Testing and Measurement (2 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Testing tools and measurement techniques – Testing and debugging; tools such as race detectors, fuzzers, lock dependency checkers, unit/stress/torture tests, visualizations, continuous integration, continuous deployment, and test generators – Measuring and comparing throughput, overhead, waiting, contention, communication, data movement, locality, resource usage, behavior in the presence of excessive numbers of events, clients, or threads. See also: Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models – Application domain specific analyses and evaluation techniques	• Identify and repair a performance problem due to communication or data latency [Analyze] • Identify and repair a performance problem due to resource management overhead [Analyze] • Identify and repair a reliability or availability problem [Analyze]
Readings : [**Sterling2024**], [**Pacheco2021**]	

Unit 16: Algorithms (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Expressing and implementing algorithms in given languages and frameworks, to initiate activities (for example threads), use shared memory constructs, and channel, socket, and/or remote procedure call APIs. See also: Parallel and Distributed Computing. – Data parallel examples including map/reduce. – Using channel, socket, and/or RPC APIs in a given language, with program control for sending (usually procedural) vs receiving. (usually reactive or RPC-based). – Using locks, barriers, and/or synchronizers to maintain liveness without introducing races.	• Implement a parallel/distributed component based on a known algorithm [Implement] • Write a data-parallel program that for example computes the average of an array of numbers [Write] • Write a producer-consumer program in which one component generates numbers, and another computes their average. Measure speedups when the numbers are small scalars versus large multi-precision values [Write]
Readings : [**Pacheco2021**], [**Herlihy2020**]	

Unit 17: Algorithms: Application Domains Survey (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Survey of common application domains across multicore, reactive, data parallel, cluster, cloud, open distributed systems, and frameworks (with reference to the following table). – Multicore: Typical Execution agents: Threads. Typical Communication mechanisms: Shared memory, Atomics, locks. Typical Algorithmic domains: Resource management, data processing. Typical Engineering goals: Throughput, latency, energy. – Reactive: Typical Execution agents: Handlers, threads. Typical Communication mechanisms: I/O Channels. Typical Algorithmic domains: Services, real-time. Typical Engineering goals: Latency. – Data parallel: Typical Execution agents: GPU, SIMD, accelerators, hybrid. Typical Communication mechanisms: Heterogeneous memory. Typical Algorithmic domains: Linear algebra, graphics, data analysis. Typical Engineering goals: Throughput, energy. – Cluster: Typical Execution agents: Managed hosts. Typical Communication mechanisms: Sockets, channels. Typical Algorithmic domains: Simulation, data analysis. Typical Engineering goals: Throughput. – Cloud: Typical Execution agents: Provisioned hosts. Typical Communication mechanisms: Service APIs. Typical Algorithmic domains: Web applications. Typical Engineering goals: Scalability. – Open distributed: Typical Execution agents: Autonomous hosts. Typical Communication mechanisms: Sockets, Data stores. Typical Algorithmic domains: Fault tolerant data stores and services. Typical Engineering goals: Reliability.	• Extend an event-driven sequential program by establishing a new activity in an event handler (for example a new thread in a GUI action handler) [Design] • Improve the performance of a sequential component by introducing parallelism and/or distribution [Create] • Choose among different parallel/distributed designs for components of a given system [Assess]
Readings : [**Pacheco2021**], [**Herlihy2020**]	

Unit 18: Algorithms: Algorithmic Domains (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• One of more of the following algorithmic domains. See also: Algorithmic Strategies: – Linear algebra: Vector and matrix operations, numerical precision/stability, applications in data analytics and machine learning. – Data processing: sorting, searching and retrieval, concurrent data structures. – Graphs, search, and combinatorics: Marking, edge-parallelization, bounding, speculation, network-based analytics. – Modeling and simulation: differential equations; randomization, N-body problems, genetic algorithms. – Computational logic: satisfiability (SAT), concurrent logic programming. – Graphics and computational geometry: Transforms, rendering, ray-tracing. – Resource management: Allocating, placing, recycling and scheduling processors, memory, channels, and hosts; exclusive vs shared resources; static, dynamic and elastic algorithms; Real-time constraints; Batching, prioritization, partitioning; decentralization via work-stealing and related techniques. – Services: Implementing web APIs, electronic currency, transaction systems, multiplayer games.	• Design, implement, analyze, and evaluate a component or application for X operating in a given context, where X is in one of the listed domains, for example, a genetic algorithm for factory floor design [Design] • Critique the design and implementation of an existing component or application, or one developed by classmates [Critique] • Compare the performance and energy efficiency of multiple implementations of a similar design, for example, multicore versus clustered versus GPU [Compare]
Readings : [**Pacheco2021**], [**Herlihy2020**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the

different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS402. Capstone Project I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS402. Capstone Project I
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS401. Research Methodology. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course aims to enable students to conduct a deep state-of-the-art study on a specific topic chosen for their thesis project. Students will produce a technical report with preliminary comparative experiments and formulate a research proposal, culminating this part alongside with a faculty advisor who specializes in the chosen research area.

5. GOALS

- Conduct a deep state-of-the-art study on a specific topic in computer science.
- Demonstrate mastery of the relevant literature through critical and comparative analysis of existing approaches.
- Design and execute preliminary experiments that reproduce and compare state-of-the-art techniques.
- Write a high-quality technical report with theoretical framework, methodology, and experimental results.
- Formulate a research proposal with research question, hypothesis, objectives, and schedule.
- Select a thesis advisor who specializes in the chosen research area.
- The deliverables for this course are:

  Midterm Progress: Solid bibliography and draft of the theoretical framework in LaTeXformat.

  Final: Technical Report with preliminary comparative experiments and research proposal approved by the advisor.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Deep State-of-the-Art Study on a Specific Topic (10 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Selection and refinement of the research topic in coordination with the advisor. • Exhaustive multi-source search: IEEE Xplore, ACM Digital Library, Google Scholar, arXiv, and specialized databases. • Identification of seminal works and most-cited papers in the area. • Construction of the citation graph and temporal evolution analysis of the topic. • Identification of sub-areas, variants, and main approaches in the field.	• Build a comprehensive literature map for a specific topic identifying the foundational works. [Evaluar] • Analyze the temporal evolution of a research area identifying trends and gaps. [Evaluar] • Organize the literature into coherent subcategories that reflect the structure of the field. [Evaluar]
Readings : [**ieee**], [**acm**], [**citeseer**]	

Unit 2: Critical and Comparative Analysis of Approaches (10 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Critical analysis of proposals: methodology, datasets, metrics, and limitations of each approach. • Construction of comparative tables of existing proposals using objective criteria. • Identification of unresolved research gaps in the area. • Formulation of the student's own research question based on the state-of-the-art analysis. • Theoretical framework: structured and coherent synthesis of relevant knowledge in the area.	• Systematically compare multiple approaches using objective and quantifiable criteria. [Evaluar] • Identify unresolved research gaps in a specific area of computer science. [Evaluar] • Formulate an original research question grounded in the state-of-the-art analysis. [Evaluar]
Readings : [**ieee**], [**acm**], [**citeseer**]	

Unit 3: Design and Execution of Preliminary Experiments (12 hours)	
Competences Expected: AG-C05,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Experimental design: variables, hypotheses, metrics, and benchmarks of the area. • Reproduction of results from relevant state-of-the-art works. • Implementation of baselines for objective comparison of approaches. • Basic statistical analysis: significance tests and confidence intervals. • Experimental data management: datasets, configurations, versions, and results logging.	• Reproduce the results of at least two relevant works from the state of the art of the area. [Usar] • Design and execute a preliminary comparative experiment with defined baselines. [Usar] • Interpret and report experimental results with appropriate statistical rigor. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 4: Technical Report Writing (12 hours)	
Competences Expected: AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Structure of the technical report: introduction, theoretical framework, methodology, experiments, results, and conclusions. • Writing the theoretical framework: thematic organization and critical synthesis of the literature. • Documentation of methodology and experimental configuration in LaTeX. • Presentation of results: high-quality tables, charts, and comparative analyses. • Revision cycles: writing, advisor review, and incorporation of feedback.	• Write a complete technical report including theoretical framework and preliminary experiments. [Usar] • Structure and present experimental results with clear and precise visualizations. [Usar] • Apply revision cycles to improve the quality, coherence, and rigor of the document. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 5: Research Proposal and Advisor Selection (8 hours)	
Competences Expected: AG-C02,AG-C05,AG-C09	
Topics	**Learning Outcomes**
• Structure of a research proposal: problem, motivation, hypothesis, and objectives. • Definition of thesis project scope: methodology, expected contribution, and schedule. • Feasibility analysis: resources, timeline, risks, and contingencies. • Criteria for selecting a thesis advisor: expertise, availability, and thematic alignment. • Formalization of the researcher-advisor relationship within the academic program.	• Produce a research proposal defining the problem, hypothesis, objectives, and methodology. [Evaluar] • Select an advisor with demonstrated expertise in the chosen research area. [Usar] • Define a viable thesis plan with objectives, scope, schedule, and risk analysis. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 6: Presentation and Defense of the State of the Art (8 hours)	
Competences Expected: AG-C02,AG-C04,AG-C05	
Topics	**Learning Outcomes**
• Preparation of the oral presentation of the technical report before the evaluation committee. • Defense techniques for handling questions and challenges from an academic committee. • Design of a technical poster of the state of the art for symposium presentation. • Incorporation of evaluating committee feedback into the final revision of the report. • Planning the transition to CS403: objectives, expected contribution, and next steps.	• Present and orally defend the technical report before an academic committee with rigor. [Evaluar] • Constructively incorporate committee feedback into a revision of the report. [Usar] • Plan the transition to the thesis project (CS403) with clear objectives and next steps. [Usar]
Readings : [**ieee**], [**acm**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

EX301. Extracurricular Activities (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	EX301. Extracurricular Activities
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	1 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course records the student's participation in activities that complement their academic training, such as workshops, conferences, social outreach, volunteering, or technical skills. Its objective is to promote comprehensive development, soft skills, and social commitment.

5. GOALS

- Participate in activities that complement their professional training.
- Develop communication and teamwork skills.
- Contribute to initiatives with a social or technical impact.

6. COMPETENCES

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Familiarity)

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies. (Familiarity)

7. TOPICS

Unit 1: Extracurricular Activities (48 hours)	
Competences Expected: AG-C01,AG-C06	
Topics	**Learning Outcomes**
• Technical or interdisciplinary workshops. • Attendance at conferences, seminars, or talks. • Social outreach (volunteering, mentoring). • Academic skills (hackathons, science fairs).	• Demonstrate participation in at least three activities [Familiarity]. • Prepare reflective reports on experiences [Usage]. • Collaborate in teams for social or technical projects [Evaluar].
Readings : [**ACM-Extra2019**], [**ABET-Criteria2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

ET201. Entrerpreneurship I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	ET201. Entrerpreneurship I
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	FG350. Leadership and Performance. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the first course in the area of technology-based business formation. Its objective is to equip future professionals with the knowledge, attitudes, and skills necessary to develop a business plan for a technology-based company. The course is divided into the following units: Introduction, Creativity, From Idea to Opportunity, the Canvas Model, Customer Development and Lean Startup, Legal and Marketing Aspects, Business Finance, and Presentation.

The course seeks to leverage students' creative and innovative potential and their efforts in the creation of new businesses.

5. GOALS

- Students will learn how to develop a business plan to launch a technology-based company.
- Students will be able to design and present a business proposal using business models.

6. COMPETENCES

No items for this table

7. TOPICS

Unit 1: (5 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Entrepreneur, entrepreneurship, and technological innovation • Business models • Team formation	• Identify the characteristics of entrepreneurs [Familiarizarse] • Introduce business models [Familiarizarse]
Readings : [**byers10**], [**osterwalder10**], [**garzozi14**]	

Unit 2: (5 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Vision • Mission • Value proposition • Creativity and invention • Types and sources of innovation • Strategy and technology • Scale and scope	• Correctly formulate a company's vision and mission [Usar] • Characterize an innovative value proposition [Evaluar] • Identify different types and sources of innovation [Familiarizarse]
Readings : [**byers10**], [**blank12**], [**garzozi14**]	

Unit 3: (5 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Business strategy • Barriers • Sustainable competitive advantage • Alliances • Organizational learning • Product development and design	• Understand business strategies [Familiarizarse] • Characterize barriers and competitive advantages [Familiarizarse]
Readings : [**byers10**], [**osterwalder10**], [**ries11**], [**garzozi14**]	

Unit 4: (20 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Creating a new business • The business plan • Canvas • Elements of the Canvas	• Understand the elements of the Canvas model [Usar] • Develop a business plan based on the Canvas model [Usar]
Readings : [**osterwalder10**], [**blank12**], [**garzozi14**]	

Unit 5: (20 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Acceleration vs. incubation • Customer Development • Lean Startup	• Understand and apply the Customer Development model [Usar] • Understand and apply the Lean Startup model [Usar]
Readings : [**blank12**], [**ries11**], [**garzozi14**]	

Unit 6: (5 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Legal and tax aspects for company formation • Intellectual property • Patents • Copyrights and trademarks • Marketing objectives and market segments • Market research and customer acquisition	• Understand the legal aspects necessary for forming a technology company [Familiarizarse] • Identify market segments and marketing objectives [Familiarizarse]
Readings : [**byers10**], [**ries11**], [**congreso96**], [**congreso97**], [**garzozi14**]	

Unit 7: (5 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Cost model • Profit model • Pricing • Financial plan • Financing methods • Sources of capital • Venture capital	• Define a cost and profit model [Evaluar] • Understand various financing sources [Familiarizarse]
Readings : [**byers10**], [**blank12**], [**garzozi14**]	

Unit 8: (5 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• The Elevator Pitch • Presentation • Negotiation	• Understand various methods for presenting business proposals [Familiarizarse] • Deliver a business proposal presentation [Usar]
Readings : [**byers10**], [**blank12**], [**garzozi14**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI268. Computational Vision (Elective)

2. GENERAL INFORMATION

2.1 Course	:	AI268. Computational Vision
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	AI264. Deep Learning. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Computer Vision is a key area of Artificial Intelligence that enables machines to interpret and understand the visual world. This course covers fundamental techniques for image processing, feature extraction, and object recognition. With the advent of deep learning, computer vision has become essential for autonomous vehicles, medical imaging, and augmented reality. Students will learn the mathematical foundations and algorithmic approaches to solve real-world visual perception problems.

5. GOALS

- Understand the fundamentals of image formation and digital representation.
- Master image processing techniques such as filtering and edge detection.
- Apply feature extraction methods to identify key patterns in images.
- Implement object detection and recognition using deep learning models.
- Analyze 3D vision and motion estimation techniques.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Introduction to AI: Concepts and Intelligent Behavior (5 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Overview of AI problems, Examples of successful recent AI applications • Definitions of agents with examples (e.g., reactive, deliberative) • What is intelligent behavior?: – The Turing test and its flaws – Multimodal input and output – Simulation of intelligent behavior – Rational versus non-rational reasoning • Overview of AI Applications, growth, and impact (economic, societal, ethics) • Philosophical issues • History of AI	• Describe the Turing test and the "Chinese Room" thought experiment [Explain] • Differentiate between optimal reasoning/behavior and human-like reasoning/behavior [Evaluate] • Differentiate the terms: AI, machine learning, and deep learning [Explain]
Readings : [**Szeliski2022**], [**Gonzales2018**]	

Unit 2: Problem Characteristics and Agent Nature (5 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Problem characteristics: – Fully versus partially observable – Single versus multi-agent – Deterministic versus stochastic – Static versus dynamic – Discrete versus continuous • Nature of agents: – Autonomous, semi-autonomous, mixed-initiative autonomy – Reflexive, goal-based, and utility-based – Decision making under uncertainty and with incomplete information – The importance of perception and environmental interactions – Learning-based agents – Embodied agents: ∗ sensors, dynamics, effectors • Practice identifying problem characteristics in example environments • Additional depth on nature of agents with examples • Additional depth on AI Applications, Growth, and Impact (economic, societal, ethics, security)	• Enumerate the characteristics of a specific problem [Apply]
Readings : [**Szeliski2022**], [**Gonzales2018**]	

Unit 3: Perception and Computer Vision (12 hours)

Competences Expected: AG-C08

Topics	**Learning Outcomes**
• Computer vision: – Image acquisition, representation, processing, and properties – Shape representation, object recognition, and segmentation – Motion analysis – Generative models • Audio and speech recognition • Touch and proprioception • Other modalities (e.g., olfaction) • Modularity in recognition • Approaches to pattern recognition: – Classification algorithms and measures of classification quality – Statistical techniques – Deep learning techniques	• Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize] • List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List] • Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply] • Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish] • Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate] • Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply] • Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply] • Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply] • Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate] • Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe] • Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]
Readings : [**Szeliski2022**], [**Prince2012**]	

Unit 4: Machine Learning Fundamentals (5 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Definition and examples of a broad variety of machine learning tasks: – Supervised learning: ∗ Classification ∗ Regression – Reinforcement learning – Unsupervised learning: ∗ Clustering • Fundamental ideas: – No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. – Sources of error and undecidability in machine learning • A simple statistical-based supervised learning such as linear regression or decision trees: – Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly • The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): – The bias (underfitting) - variance (overfitting) tradeoff	• Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] • Differentiate the terms of AI, machine learning, and deep learning [Evaluate] • Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] • Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] • Explain how machine learning works as an optimization/search process [Explain]
Readings : [**Goodfellow2016**], [**Russell2020**]	

Unit 5: Machine Learning Data, Representations and Evaluation (5 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Working with Data: – Data preprocessing: ∗ Importance and pitfalls of preprocessing choices – Handling missing values (imputing, flag-as-missing): ∗ Implications of imputing vs flag-as-missing – Encoding categorical variables, encoding real-valued data – Normalization/standardization – Emphasis on real data, not textbook examples • Representations: – Hypothesis spaces and complexity – Simple basis feature expansion, such as squaring univariate features – Learned feature representations • Machine learning evaluation: – Separation of train, validation, and test sets – Performance metrics for classifiers – Estimation of test performance on held-out data – Tuning the parameters of a machine learning model with a validation set – Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions	• Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] • Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] • Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] • Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]
Readings : [**Goodfellow2016**], [**Russell2020**]	

Unit 6: Neural Networks and Ethics (4 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Basic neural networks: – Fundamentals of understanding how neural networks work and their training process, without details of the calculations – Basic introduction to generative neural networks (e.g., large language models) • Ethics for Machine Learning: – Focus on real data, real scenarios, and case studies – Dataset/algorithmic/evaluation bias and unintended consequences	• Describe the neural network training process and resulting learned representations [Explain]
Readings : [**Goodfellow2016**], [**Russell2020**]	

Unit 7: Machine Learning Formulation and Evaluation (3 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: – Objective function – Gradient descent – Regularization to avoid overfitting (mathematical formulation) • Ensembles of models: – Simple weighted majority combination • Performance evaluation: – Other metrics for classification (e.g., error, precision, recall) – Performance metrics for regressors – Confusion matrix – Cross-validation: ∗ Parameter tuning (grid/random search, via cross-validation)	• Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare] • Evaluate the performance of a simple learning system on a real-world dataset [Evaluate] • Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain] • Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain] • Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]
Readings : [**Goodfellow2016**]	

Unit 8: Deep Learning, RL and Ethics (3 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Deep learning: – Deep feed-forward networks (intuition only, no mathematics) – Convolutional neural networks (intuition only, no mathematics) – Visualization of learned feature representations from deep nets – Other architectures (generative NN, recurrent NN, transformers, etc.) • Overview of reinforcement learning methods • Two or more applications of machine learning algorithms: – E.g., medicine and health, economics, vision, natural language, robotics, game play • Ethics for Machine Learning: – Continued focus on real data, real scenarios, and case studies – Privacy – Fairness – Intellectual property – Explainability	• Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] • Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]
Readings : [**Goodfellow2016**]	

Unit 9: Perception and Computer Vision (12 hours)	
Competences Expected: AG-C08	
Topics	**Learning Outcomes**
• Computer vision: – Image acquisition, representation, processing, and properties – Shape representation, object recognition, and segmentation – Motion analysis – Generative models • Audio and speech recognition • Touch and proprioception • Other modalities (e.g., olfaction) • Modularity in recognition • Approaches to pattern recognition: – Classification algorithms and measures of classification quality – Statistical techniques – Deep learning techniques	• Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize] • List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List] • Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply] • Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish] • Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate] • Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply] • Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply] • Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply] • Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate] • Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe] • Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]
Readings : [**Szeliski2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS391. Software Engineering III (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS391. Software Engineering III
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS292. Software Engineering II. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Software Engineering III focuses on advanced topics of the software lifecycle, emphasizing software quality, evolution, and deployment in modern architectures. This course prepares students to handle the complexities of maintaining systems, ensuring software quality through rigorous validation, and the application of professional tools and design practices for resilient and maintainable software products.

5. GOALS

- Implement advanced quality assurance and reliability metrics.
- Understand software evolution, refactoring, and technical debt management.
- Integrate development and operations tools in collaborative environments.
- Design cloud-native applications using advanced software architectures.
- Analyze and apply systematic testing and validation strategies.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Verification and validation Concepts (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation concepts – Verification: Are we building the thing right? – Validation: Did we build the right thing? • Why testing matters: Does the component remain functional as the code evolves? • Testing objectives – Usability – Reliability – Conformance to specification – Performance – Security	• Explain why testing is important [Explain] • Distinguish between program validation and verification [Distinguish] • Describe different objectives of testing [Describe]
Readings : [**Sommerville2015**], [**Pressman2019**]	

Unit 2: Testing Tools and Analysis (3 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation tooling and automation – Static analysis – Code coverage – Fuzzing – Dynamic analysis and fault containment (sanitizers, etc.) – Fault logging and fault tracking • Test planning and generation – Fault estimation and testing termination including defect seeding – Use of random and pseudo random numbers in testing • Testing asynchronous, parallel, and concurrent systems • Verification and validation of non-code artifacts (documentation, training materials) • Behavior-driven development (BDD) – Expressing acceptance criteria as Gherkin Given/When/Then scenarios – Executing BDD scenarios as automated end-to-end tests • Mutation testing and static code-quality analysis – Mutation testing to assess test-suite effectiveness – Static detection of code smells and excessive cyclomatic complexity	• Describe and compare different tools for verification and validation [Describe] • Automate the use of different tools in a small software project [Design] • Explain how and when random numbers should be used in testing [Explain] • Describe approaches for fault estimation [Describe] • Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] • Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] • Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] • Describe techniques for the verification and validation of non-code artifacts [Describe] • Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] • Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]
Readings : [**Pressman2019**]	

Unit 3: Performance Testing and Benchmarking (3 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Performance testing and benchmarking – Throughput and latency – Degradation under load (stress testing, FIFO vs LIFO handling of requests) – Speedup and scaling ∗ Amdahl's law ∗ Gustafson's law ∗ Soft and weak scaling – Identifying and measuring figures of merits – Common performance bottlenecks ∗ Compute-bound ∗ Memory-bandwidth bound ∗ Latency-bound – Statistical methods and best practices for benchmarking ∗ Estimation of uncertainty ∗ Confidence intervals – Analysis and presentation (graphs, etc.) – Timing techniques	• Describe throughput and latency and provide examples of each [Describe] • Explain speedup and the different forms of scaling and how they are computed [Explain] • Describe common performance bottlenecks [Describe] • Describe statistical methods and best practices for benchmarking software [Describe] • Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] • Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]
Readings : [**Bondi2015**], [**Gregg2020**]	

Unit 4: API Compatibility and Versioning (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Hyrum's Law/The Law of Implicit Interfaces • Backward compatibility – Compatibility is not a property of a single entity, it's a property of a relationship. – Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises. • Versioning – Semantic Versioning (SemVer) – Trunk-based development	• Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze] • Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze] • Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]
Readings : [**Sommerville2015**]	

Unit 5: Version Control and CI/CD (12 hours)

Competences Expected: AG-C03,AG-C09

Topics	Learning Outcomes
• Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant. • Release management. • Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security • Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Collaborative review workflow as a quality gate – Mandatory pull-request review before merging – Code ownership via CODEOWNERS-style rules – Conventional commit message conventions – Branch-protection rules enforcing checks before merge • Dependency and supply-chain vulnerability scanning integrated into CI pipelines	• Describe the difference between centralized and distributed software configuration management [Describe] • Describe how version control can be used to help manage software release management [Describe] • Identify configuration items and use a source code control tool in a small team-based project [Analyze] • Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] • Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] • Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] • Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Readings : [**Kim2016**]

Unit 6: Principles and Software Architecture (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• System design principles. System Reliability – Levels of abstraction (e.g., architectural design and detailed design) – Separation of concerns – Information hiding – Coupling and cohesion • Software architecture. System Reliability – Design paradigms ∗ Top-down functional decomposition/layered design ∗ Data-oriented architecture ∗ Object-oriented analysis and design ∗ Event-driven design – Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) – Identifying component boundaries and dependencies • Programming in the large vs programming in the small. System Reliability • Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering	• Identify the standard software architecture of a given high-level design [Analyze] • Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] • Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] • Identify the dependencies among a set of software components in an architectural design [Analyze]
Readings : [**Fowler2017**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS393. Information systems (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS393. Information systems
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS291. Software Engineering I. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course provides a comprehensive overview of how information systems support business processes and decision-making within organizations. It covers the strategic role of information technology, the lifecycle of information systems, data management strategies, and the integration of enterprise applications to achieve competitive advantages in a globalized digital economy.

5. GOALS

- Understand the fundamental components and types of Information Systems.
- Analyze the role of IS in organizational strategy and decision-making.
- Evaluate emerging technologies and their impact on business models.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Foundations of Information Systems (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Components of Information Systems: Hardware, Software, Data, Networks. • Types of IS: TPS, MIS, DSS, and ESS. • Competitive advantage through Information Technology. • Business processes and Information Systems integration.	• Identify the core components of an information system [Familiarizarse]. • Explain how IS supports various levels of organizational hierarchy [Usar]. • Analyze a business case to identify strategic uses of IT [Evaluar].
Readings : [**Laudon2021**], [**Stair2017**]	

Unit 2: Data Management and Business Intelligence (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Database management systems and data warehousing. • Big Data analytics in a business context. • Business Intelligence (BI) and Data Visualization tools. • Knowledge Management Systems.	• Design a basic data architecture for a business requirement [Usar]. • Evaluate the importance of data quality and governance [Evaluar].
Readings : [**Laudon2021**], [**Valacich2021**]	

Unit 3: Enterprise Systems and E-Commerce (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Enterprise Resource Planning (ERP) systems. • Supply Chain Management (SCM) and Customer Relationship Management (CRM). • E-commerce models: B2B, B2C, and C2C. • Mobile commerce and digital payment systems.	• Describe the functional areas of an ERP system [Familiarizarse]. • Assess the impact of e-commerce on traditional business models [Evaluar].
Readings : [**Stair2017**], [**Valacich2021**]	

Unit 4: IS Development and Security (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Systems Development Life Cycle (SDLC) and Agile methodologies. • Outsourcing and cloud-based IS services. • Information security threats and risk management. • Ethical and social issues in Information Systems.	• Compare different system development methodologies [Usar]. • Formulate ethical guidelines for data privacy and security [Evaluar].
Readings : [**Laudon2021**], [**Stair2017**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS394. Software Architecture and DevOps for Cloud Software (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS394. Software Architecture and DevOps for Cloud Software
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS292. Software Engineering II. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Modern software design requires combining solid architectural principles with agile continuous delivery practices and cloud operations. This course integrates three areas that are inseparable in today's industry: software architecture, DevOps, and cloud computing.

Students will learn to design systems at scale using well-established architectural patterns, to automate build, test, and deployment cycles through CI/CD pipelines, and to provision and operate infrastructure on public cloud platforms using containers, orchestration, and infrastructure-as-code tools.

The course places special emphasis on the reliability and observability of distributed systems, covering metrics, distributed traces, error budgets, and site reliability engineering (SRE) practices as applied in high-performing technology organizations.

5. GOALS

- Design scalable and maintainable software systems by applying modern architectural patterns.
- Implement continuous integration and continuous delivery (CI/CD) pipelines to accelerate and secure the software lifecycle.
- Deploy and manage containerized applications on public cloud platforms using industry-standard tools.
- Provision cloud infrastructure in a reproducible and versioned manner through infrastructure as code (IaC).
- Apply observability and SRE practices to ensure the reliability of systems in production.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Principles and Software Architecture (8 hours)

Competences Expected: AG-C09,AG-C12

Topics	Learning Outcomes
• System design principles. System Reliability – Levels of abstraction (e.g., architectural design and detailed design) – Separation of concerns – Information hiding – Coupling and cohesion • Software architecture. System Reliability – Design paradigms ∗ Top-down functional decomposition/layered design ∗ Data-oriented architecture ∗ Object-oriented analysis and design ∗ Event-driven design – Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) – Identifying component boundaries and dependencies • Programming in the large vs programming in the small. System Reliability • Structural and behavioral models of software designs • Design modeling, for instance with class diagrams, entity relationship diagrams, or sequence diagrams • API design principles – Consistency ∗ Consistent APIs are easier to learn and less error-prone ∗ Consistency is both internal (between different portions of the API) and external (following common API patterns) – Composability – Documenting contracts ∗ API operations should describe their effect on the system, but not generally their implementation ∗ Preconditions, postconditions, and invariants – Expandability – Error reporting ∗ Errors should be clear, predictable, and actionable ∗ Input that does not match the contract should produce an error ∗ Errors that can be reliably managed without reporting should be managed	• Identify the standard software architecture of a given high-level design [Analyze] • Identify the dependencies among a set of software components in an architectural design [Analyze] • Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] • Translate a natural language software design into class diagrams [Translate]

Readings : [Bass2021] [Richards2020]

Unit 2: Software Design Quality and Evaluation (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Evaluating design tradeoffs (e.g., efficiency vs reliability, security vs usability) • Architectural styles: layered, event-driven, microkernel, service-oriented, pipeline. • Quality attributes: availability, scalability, performance, security, maintainability. • Requirement traceability – Understanding which requirements are satisfied by a design • Measurement and analysis of design quality	• Contrast two software designs across different qualities, such as efficiency or usability [Contrast] • Identify which requirements are satisfied by a provided software design [Analyze] • Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] • Select the most appropriate architectural style given the quality attributes required by a system. [Evaluar]
Readings : [**Richards2020**]	

Unit 3: API Design and Modeling (6 hours)

Competences Expected: AG-C09,AG-C11

Topics	**Learning Outcomes**
• Microservices principles: cohesion, loose coupling, and service boundaries. • Monolith decomposition: domain-driven design (DDD). • API design principles – Consistency ∗ Consistent APIs are easier to learn and less error-prone ∗ Consistency is both internal (between different portions of the API) and external (following common API patterns) – Composability – Documenting contracts ∗ API operations should describe their effect on the system, but not generally their implementation ∗ Preconditions, postconditions, and invariants – Expandability – Error reporting ∗ Errors should be clear, predictable, and actionable ∗ Input that does not match the contract should produce an error ∗ Errors that can be reliably managed without reporting should be managed • API Gateway and service mesh. • Resilience patterns: Circuit Breaker, Retry, Bulkhead, Timeout. • Data Stores – Cooperatively maintained data structures implementing maps and related ADTs – Varieties: Owned, shared, sharded, replicated, immutable, versioned	• Design an API for a single component of a large software system, including identifying and documenting each operation's invariants, contract, and error conditions [Design] • Evaluate an API description in terms of consistency, composability, and expandability [Evaluate] • Design a microservices-based system by decomposing a monolithic application into independent services. [Usar] • Apply resilience patterns to tolerate failures in distributed systems. [Usar]
Readings : [**Newman2021**]	

Unit 4: Virtualization (6 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• Containers and their comparison to virtual machines. • Types of virtualizations (including hardware/software, OS, server, service, network). See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Portable virtualization; emulation vs isolation. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation • Virtual machines and container escapes, dangers from a security perspective. See also: Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering • Dockerfile: best practices for building efficient and secure images. • Multi-container application composition with docker-compose.	• Explain the difference between emulation and isolation [Explain] • Evaluate virtualization tradeoffs [Evaluate] • Build and deploy containerized applications using Docker. [Usar] • Apply security best practices when building Docker images. [Usar]
Readings : [**Burns2022**]	

Unit 5: Coordination (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Kubernetes architecture: control plane and worker nodes. • Core objects: Pods, Deployments, Services, ConfigMaps, Secrets. • Horizontal Pod Autoscaling (HPA) and self-healing. • Dependencies – Initiation or progress of one activity may be dependent on other activities, so as to avoid race conditions, ensure termination, or meet other requirements – Ensuring progress by avoiding dependency cycles, using monotonic conditions, removing inessential dependencies • Package management with Helm.	• Deploy and manage containerized applications on a Kubernetes cluster. [Usar] • Configure auto-scaling and self-healing for cloud-native workloads. [Usar] • Modify a concurrent system to use a more scalable, reliable, or available data store [Create]
Readings : [**Burns2022**]	

Unit 6: Version Control and CI/CD (6 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant. • Release management. • Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security • Tool integration concepts and mechanisms. Software Development Practices • DevOps culture: flow, feedback, and continuous learning.	• Describe the difference between centralized and distributed software configuration management [Describe] • Describe how version control can be used to help manage software release management [Describe] • Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] • Demonstrate the capability to use software tools in support of the development of a software product of medium size [Demonstrate]
Readings : [**Kim2021**]	

Unit 7: Version Control and CI/CD (6 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• IaC principles: reproducibility, idempotency, and infrastructure versioning. • Declarative vs. imperative approaches. • Terraform: providers, resources, modules, and state management. • Cloud-native IaC tools: AWS CloudFormation, Azure ARM Templates, GCP Deployment Manager. • Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant.	• Write and apply Terraform configurations to provision cloud infrastructure. [Usar] • Identify configuration items and use a source code control tool in a small team-based project [Analyze] • Compare IaC tools based on their state management and modularity capabilities. [Evaluar]
Readings : [**Morris2021**], [**Kim2021**]	

Unit 8: Networked Applications (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Cloud service models: IaaS, PaaS, SaaS, FaaS. • Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination • Major providers: AWS, Google Cloud, Azure — core services. • Compute services: virtual machines, serverless functions, managed containers. • Storage services: object, block, and managed databases. • Cloud networking: VPCs, load balancers, CDN.	• Analyze the needs of specific networked application demands [Analyze] • Compare IaaS, PaaS, SaaS, and FaaS models to select the most appropriate one for a given architecture. [Evaluar] • Design a cloud-native application using managed services from a major cloud provider. [Usar]
Readings : [**Kim2021**], [**Burns2022**]	

Unit 9: Reliability Engineering (6 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• Measurement-based analysis of software reliability - telemetry, monitoring and alerting, dashboards, release qualification metrics, etc. • The three pillars of observability: metrics, logs, and distributed traces. • Distributed tracing and correlation identifiers. • Software reliability engineering practices - including reviews, testing, practical model checking • Alerting strategies and on-call practices.	• Describe how to determine the level of reliability required by a software system [Describe] • Identify methods that will lead to the realization of a software architecture that achieves a specified level of reliability [Analyze] • Implement observability in a cloud application using metrics, logs, and distributed traces. [Usar]
Readings : [**Kim2021**]	

Unit 10: System Reliability (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures • Compare the ”error budget” approach to reliability with the ”error-free” approach and identify domains where each is relevant. • Software fault tolerance techniques and models – Contextual differences in fault tolerance (e.g., crashing a flight critical system is strongly avoided, crashing a data processing system before corrupt data is written to storage is highly valuable) • Reliability through redundancy: – check and retry Reliability Support, Fault tolerance – redundant encoding (error correction codes, CRC, FEC, RAID) Memory Hierarchy, Distributed Databases/Cloud Computing, Reliability Support – duplication/mirroring/replicas Distributed Databases/Cloud Computing • Software reliability, system reliability and failure behavior • Service Level Indicators (SLI), Objectives (SLO), and Agreements (SLA).	• Identify ways to apply redundancy to achieve fault tolerance [Analyze] • Identify single-point-of-failure (SPF) dependencies in a system design [Analyze] • Explain the problems that exist in achieving very high levels of reliability [Explain] • Define SLIs, SLOs, and error budgets for a service in production. [Usar]
Readings : [**Kim2021**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS395. Software Quality (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS395. Software Quality
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS292. Software Engineering II. (7^{th} Sem)

3. PROFESSORS

Lecturer

- Ernesto Cuadros-Vargas <ecuadrosv@uni.edu.pe>
  - PhD in Computer Science, ICMC-USP, Brazil, 2004.
  - MSc in Computer Science, ICMC-USP, Brazil, 1998.

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Delivering software that is merely "functional" is no longer sufficient for industry: modern organizations demand systems that are reliable, secure, maintainable, and built through disciplined, auditable engineering processes. This course teaches software quality as a first-class, hands-on engineering discipline practiced through the same tools and workflows used by professional teams on GitHub: branch protection, mandatory code review, automated testing at every level, continuous integration/continuous delivery (CI/CD), static analysis, and production observability.

Students progress from foundational quality models through advanced collaborative-development workflows, automated testing strategies (unit, integration, end-to-end, and behavior-driven), CI/CD pipelines, static analysis and refactoring, and production-quality concerns such as supply-chain security, performance, and reliability engineering. The course closes with an emphasis on DevOps culture – the shared team practices and values (Definition of Done, blameless postmortems, continuous learning) that sustain quality at scale.

5. GOALS

- Evaluate software quality using an internationally recognized quality-characteristics model and reason about the cost of poor quality and technical debt.
- Apply a professional, review-gated Git/GitHub workflow, including branch protection, CODEOWNERS, and conventional commits.
- Design and implement unit, integration, end-to-end, and behavior-driven test suites, including test-driven development.
- Build and operate CI/CD pipelines that enforce automated quality gates, dependency scanning, and semantic versioning.
- Apply static analysis, mutation testing, and refactoring techniques to identify and remove code smells and technical debt.
- Analyze production-quality concerns, including performance testing and reliability/observability practices.
- Adopt DevOps culture practices – Definition of Done, blameless postmortems, and continuous team learning – to sustain quality.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

<table>
<tr><td colspan="2">Unit 1: Software Design Quality and Evaluation (4 hours)</td></tr>
<tr><td colspan="2">Competences Expected: AG-C09,AG-C12</td></tr>
<tr><td>Topics</td><td>Learning Outcomes</td></tr>
<tr><td>

Data design Data Modeling

Data structures
Storage systems

Requirement traceability

Understanding which requirements are satisfied by a design

Design modeling, for instance with class diagrams, entity relationship diagrams, or sequence diagrams
Measurement and analysis of design quality
Principles of secure design and coding Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering

Principle of least privilege
Principle of fail-safe defaults
Principle of psychological acceptability

Evaluating design tradeoffs (e.g., efficiency vs reliability, security vs usability)
ISO/IEC 25010 quality characteristics model and cost of poor quality

Quality characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability
Cost of poor quality and technical debt as a business/economic concern, not just a technical one

</td><td>

Design a set of data structures to implement a provided API surface [Design]
Identify which requirements are satisfied by a provided software design [Analyze]
Translate a natural language software design into class diagrams [Translate]
Adapt a flawed system design to better follow the principles of least privilege and fail-safe defaults [Create]
Contrast two software designs across different qualities, such as efficiency or usability [Contrast]
Evaluate a software system against the ISO/IEC 25010 quality characteristics and estimate the business cost of poor quality/technical debt [Evaluate]

</td></tr>
<tr><td colspan="2">Readings : [ISO25010], [Sommerville2015]</td></tr>
</table>

Unit 2: Version Control and CI/CD (12 hours)

Competences Expected: AG-C11,AG-C12

Topics	**Learning Outcomes**
• Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant. • Release management. • Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security • Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Collaborative review workflow as a quality gate – Mandatory pull-request review before merging – Code ownership via CODEOWNERS-style rules – Conventional commit message conventions – Branch-protection rules enforcing checks before merge • Dependency and supply-chain vulnerability scanning integrated into CI pipelines	• Describe the difference between centralized and distributed software configuration management [Describe] • Describe how version control can be used to help manage software release management [Describe] • Identify configuration items and use a source code control tool in a small team-based project [Analyze] • Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] • Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] • Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] • Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Readings : [**Kim2021**]

Unit 3: Test Planning and Types (8 hours)

Competences Expected: AG-C09

Topics	**Learning Outcomes**
• Test kinds – Unit – Integration – Validation – System • Stylistic differences between tests and production code: DAMP vs DRY - more duplication is warranted in test code. • Test planning and generation – Test case generation, from formal models, specifications, etc. – Test coverage * Test matrices * Code coverage - how much of the code is tested? * Environment coverage - how many hardware architectures, operating systems, browsers, etc. are tested? – Test data and inputs	• Compare and contrast the different types and levels of testing (regression, unit, integration, systems, and acceptance) [Compare] • Describe techniques for creating a test plan and generating test cases [Describe] • Create a test plan for a medium-size code segment which includes a test matrix and generation of test data and inputs [Create] • Implement a test plan for a medium-size code segment [Implement]

Readings : [**Sommerville2015**]

Unit 4: Advanced Testing Practices (4 hours)

Competences Expected: AG-C11

Topics	**Learning Outcomes**
• Test development – Test-driven development – Object oriented testing, mocking, and dependency injection – Opaque-box (previously, black-box) and transparent-box (previously, white-box) testing techniques – Test tooling, including code coverage, static analysis, and fuzzing • Verification and validation in the development cycle – Code reviews – Test automation, including automation of tooling – Pre-commit and post-commit testing – Tradeoffs between test coverage and throughput/latency of testing – Defect tracking and prioritization: reproducibility of reported defects • Domain specific verification and validation challenges – Performance testing and benchmarking – Asynchrony, parallelism, and concurrency – Safety-critical – Numeric	• Identify the fundamental principles of test-driven development methods and explain the role of automated testing in these methods [Analyze] • Discuss issues involving the testing of object-oriented software [Debate] • Describe mocking and dependency injection and their application [Describe] • Undertake, as part of a team activity, a code review of a medium-size code segment [Evaluate] • Describe the role that tools can play in the validation of software [Describe] • Automate the testing in a small software project [Design] • Explain the roles, pros, and cons of pre-commit and post-commit testing [Explain] • Discuss the tradeoffs between test coverage and test throughput/latency and how this can impact verification [Debate] • Use a defect tracking tool to manage software defects in a small software project [Use] • Discuss the limitations of testing in certain domains [Debate]

Readings : [**Beck2002**]

Unit 5: Testing Tools and Analysis (8 hours)	
Competences Expected: AG-C11	
Topics	**Learning Outcomes**
• Verification and validation tooling and automation – Static analysis – Code coverage – Fuzzing – Dynamic analysis and fault containment (sanitizers, etc.) – Fault logging and fault tracking • Test planning and generation – Fault estimation and testing termination including defect seeding – Use of random and pseudo random numbers in testing • Testing asynchronous, parallel, and concurrent systems • Verification and validation of non-code artifacts (documentation, training materials) • Behavior-driven development (BDD) – Expressing acceptance criteria as Gherkin Given/When/Then scenarios – Executing BDD scenarios as automated end-to-end tests • Mutation testing and static code-quality analysis – Mutation testing to assess test-suite effectiveness – Static detection of code smells and excessive cyclomatic complexity	• Describe and compare different tools for verification and validation [Describe] • Automate the use of different tools in a small software project [Design] • Explain how and when random numbers should be used in testing [Explain] • Describe approaches for fault estimation [Describe] • Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] • Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] • Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] • Describe techniques for the verification and validation of non-code artifacts [Describe] • Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] • Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]
Readings : [**Wynne2012**]	

Unit 6: API Compatibility and Versioning (4 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• Hyrum's Law/The Law of Implicit Interfaces • Backward compatibility – Compatibility is not a property of a single entity, it's a property of a relationship. – Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises. • Versioning – Semantic Versioning (SemVer) – Trunk-based development	• Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze] • Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze] • Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]
Readings : [**Sommerville2015**]	

Unit 7: Refactoring Techniques (4 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Refactoring – Standard refactoring patterns (rename, inline, outline, etc.) – Use of refactoring tools in IDE – Application of static-analysis tools (to identify code in need of refactoring, generate changes, etc.) – Value of refactoring as a remedy for technical debt • "Large Scale" Refactoring - techniques when a refactoring change is too large to commit safely (large projects), or when it is impossible to synchronize change between provider + all consumers (multiple repositories, consumers with private code). – Express both old and new APIs so that they can co-exist. – Minimize the size of behavior changes. – Why these techniques are required, (e.g., "API consumers I can see" vs "consumers I can't see").	• Consider inputs from static analysis tools and/or Software Design principles to identify code in need of refactoring [Analyze] • Refactor the implementation of an interface to improve design, clarity, etc. with minimal/zero impact on existing users [Create] • Plan a complex multi-step refactoring to change default behavior of an API safely [Plan]
Readings : [**Fowler2017**]	

Unit 8: Performance Testing and Benchmarking (4 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Performance testing and benchmarking – Throughput and latency – Degradation under load (stress testing, FIFO vs LIFO handling of requests) – Speedup and scaling ∗ Amdahl's law ∗ Gustafson's law ∗ Soft and weak scaling – Identifying and measuring figures of merits – Common performance bottlenecks ∗ Compute-bound ∗ Memory-bandwidth bound ∗ Latency-bound – Statistical methods and best practices for benchmarking ∗ Estimation of uncertainty ∗ Confidence intervals – Analysis and presentation (graphs, etc.) – Timing techniques	• Describe throughput and latency and provide examples of each [Describe] • Explain speedup and the different forms of scaling and how they are computed [Explain] • Describe common performance bottlenecks [Describe] • Describe statistical methods and best practices for benchmarking software [Describe] • Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] • Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]
Readings : [**Bondi2015**], [**Gregg2020**]	

Unit 9: Reliability Engineering (4 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• Software reliability models • Software fault tolerance techniques and models – Contextual differences in fault tolerance (e.g., crashing a flight critical system is strongly avoided, crashing a data processing system before corrupt data is written to storage is highly valuable) • Software reliability engineering practices - including reviews, testing, practical model checking • Identification of dependent and independent failure domains, and their impact on system reliability • Measurement-based analysis of software reliability - telemetry, monitoring and alerting, dashboards, release qualification metrics, etc.	• Demonstrate the ability to apply multiple methods to develop reliability estimates for a software system [Demonstrate] • Identify methods that will lead to the realization of a software architecture that achieves a specified level of reliability [Analyze] • Identify ways to apply redundancy to achieve fault tolerance [Analyze] • Identify single-point-of-failure (SPF) dependencies in a system design [Analyze]
Readings : [**Kim2021**], [**Forsgren2018**]	

Unit 10: Large-Scale Construction and Process (4 hours)

Competences Expected: AG-C12

Topics	**Learning Outcomes**
• Larger-scale testing – Test doubles (stubs, mocks, fakes) – Dependency injection • Work sequencing, including dependency identification, milestones, and risk retirement – Dependency identification: Identifying the dependencies between different tasks – Milestones: A collection of tasks that serve as a marker of progress when completed. Ideally, the milestone encompasses a useful unit of functionality. – Risk retirement: Identifying what elements of a project are risky and prioritizing completing tasks that address those risks. • Potential security problems in programs Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security – Buffer and other types of overflows – Race conditions – Improper initialization, including choice of privileges – Input validation • Documentation (autogenerated) • Development context: "green field" vs existing code base – Change impact analysis – Change actualization • Release management • DevOps practices – Definition of Done as a shared quality bar for completed work – Blameless postmortems for learning from incidents/failures – Team quality culture: shared ownership, feedback, and continuous learning	• Rewrite a simple program to remove common vulnerabilities, such as buffer overflows, integer overflows and race conditions [Apply] • Write a software component that performs some non-trivial task and is resilient to input and run-time errors [Write]

Readings : [**Forsgren2018**]

Unit 11: Capstone: End-to-End Quality Pipeline Demonstration (8 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Integration of all course practices into a single live repository: branch protection, mandatory PR review, automated test suites (unit, integration, E2E/BDD), CI/CD pipeline with quality gates, dependency scanning, static analysis, and observability. • Live repository review: demonstration and defense of the team's quality pipeline in front of peers/instructor. • Retrospective and blameless postmortem of the project's quality journey.	• Integrate version control, automated testing, CI/CD, static analysis, and observability into a single, end-to-end quality pipeline for a real repository. [Usar] • Defend the design and effectiveness of a team's quality pipeline through a live repository review. [Usar] • Conduct a blameless postmortem to identify process improvements for future projects. [Evaluar]
Readings : [**Forsgren2018**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS370. Big Data (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS370. Big Data
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS3P1. Parallel and Distributed Computing . (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

In the current era, the volume, velocity, and variety of data exceed the capabilities of traditional storage and processing systems. This course introduces the fundamental concepts and technologies of Big Data. It covers distributed file systems, parallel processing frameworks, NoSQL databases, and real-time data streaming. Students will learn to design scalable architectures to extract value from massive datasets.

5. GOALS

- Understand the 5 V's of Big Data and their impact on system design.
- Design and implement scalable solutions using the MapReduce paradigm.
- Master the use of distributed file systems like HDFS.
- Analyze different architectural patterns for Big Data (Lambda and Kappa).
- Apply stream processing techniques for real-time data analysis.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Core Database System Concepts (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Purpose and advantages of database systems • Components of database systems • Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods) • Database architecture, data independence, and data abstraction • Transaction management • Normalization • Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce) • How to support CRUD-only applications • Distributed databases/cloud-based systems • Structured, semi-structured, and unstructured data • Use of a declarative query language • Systems supporting structured and/or stream content	• Identify at least four advantages that using a database system provides [Analyze] • Enumerate the components of a (relational) database system [Enumerate] • Follow a query as it is processed by the components of a (relational) database system [Analyze] • Defend the value of data independence [Defend] • Compose a simple select-project-join query in SQL [Compose] • Enumerate the four properties of a correct transaction manager [Enumerate] • Describe the advantages for eliminating duplicate repeated data [Describe] • Outline how MapReduce uses parallelism to process data efficiently [Outline] • Evaluate the differences between structured and semi/unstructured databases [Evaluate]
Readings : [**White2015**], [**Karau2017**]	

Unit 2: Distributed Databases/Cloud Computing (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Distributed DBMS: – Distributed data storage – Distributed query processing – Distributed transaction model – Homogeneous and heterogeneous solutions – Client-server distributed databases • Parallel DBMS: – Parallel DBMS architectures: shared memory, shared disk, shared nothing – Speedup and scale-up, e.g., use of the MapReduce processing model – Data replication and weak consistency models	• Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe] • Analyze the tradeoffs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze] • Describe data replication strategies and weak consistency models in distributed database systems [Describe]
Readings : [**White2015**], [**Lin2010**]	

Unit 3: NoSQL Systems (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Why NoSQL? (e.g., Impedance mismatch between Application [CRUD] and RDBMS) • Key-Value and Document data model • Storage systems (e.g., Key-Value systems, Data Lakes) • Distribution Models (Sharding and Replication) • Graph Databases • Consistency Models (Update and Read, Quorum consistency, CAP theorem) • Processing model (e.g., Map-Reduce, multi-stage map-reduce, incremental map-reduce) • Case Studies: Cloud storage system (e.g., S3); Graph databases; "When not to use NoSQL"	• Develop a use case for the use of NoSQL over RDBMS [Create] • Describe the defining characteristics behind Key-Value and Document-based data models [Describe]
Readings : [**Sadalage2012**], [**Kleppmann2017**]	

Unit 4: Data Analytics (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Exploratory data techniques (motivation, representation, descriptive statistics, visualizations) • Data science lifecycle: business understanding, data understanding, data preparation, modeling, evaluation, deployment, and user acceptance • Data mining and machine learning algorithms: e.g., classification, clustering, association, regression • Data acquisition and governance • Data security and privacy considerations • Data fairness and bias • Data visualization techniques and their use in data analytics • Entity Resolution	• Describe several data exploration approaches, including visualization, to understanding unfamiliar datasets [Describe] • Apply several data exploration approaches to understanding unfamiliar datasets [Apply] • Describe basic machine learning/data mining algorithms and when they are appropriate for use [Describe] • Apply several machine learning/data mining algorithms [Apply] • Describe legal and ethical considerations in acquiring, using, and modifying datasets [Describe] • Describe issues of fairness and bias in data collection and usage [Describe]
Readings : [**zaharia2012resilient**], [**chambers2018spark**], [**Kleppmann2017**], [**Psaltis2017**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS400. Pre-professional internships (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS400. Pre-professional internships
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	2
2.4 Hours	:	1 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	110Cr

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course allows students to apply the knowledge acquired during their academic training in a real work environment, under the supervision of a company and the university. Pre-professional internships are essential to develop professional, ethical, and technical competencies, ensuring that the student gains experience in projects related to the degree program.

5. GOALS

- Evaluate student performance in a real work environment, applying Computer Science knowledge.
- Develop professional, ethical, and teamwork skills in a business context.
- Ensure that the activities performed are aligned with the degree program objectives.

6. COMPETENCES

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Usage)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Usage)

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Performance Evaluation in Work Environment (48 hours)	
Competences Expected: AG-C02,AG-C04,AG-C05,AG-C09	
Topics	**Learning Outcomes**
• Student integration into a professional work team. • Application of technical knowledge in real projects. • Evaluation of ethical and professional competencies in the work environment. • Presentation of reports and feedback from the company and university.	• Demonstrate technical skills in a real work environment [Evaluar]. • Apply ethical and professional principles in their activities [Evaluar]. • Work effectively in teams and communicate professionally [Evaluar].
Readings : [**ACMInternshipGuide**], [**IEEEProfessionalPractices**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS403. Capstone Project II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS403. Capstone Project II
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS402. Capstone Project I. (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course aims to enable students to develop their research proposal and execute formal experiments, producing the core chapters of their thesis under the supervision of their faculty advisor.

5. GOALS

- Formalize the thesis plan in the institutional format with objectives, schedule, and feasibility analysis.
- Write the theoretical framework and related works chapter with critical synthesis of the literature.
- Design the original methodology and implement a functional prototype of the research proposal.
- Execute preliminary comparative experiments with statistical rigor and well-grounded interpretation of results.
- Present and defend the thesis progress before the evaluation committee incorporating their feedback.
- The deliverables for this course are:

  Midterm Progress: Thesis plan in institutional format, theoretical framework chapter and related works in LaTeX.

  Final: Methodology, proposal, and preliminary experiments chapters approved by the advisor.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Thesis Plan Formalization (10 hours)	
Competences Expected: AG-C05,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Structure of the thesis plan following the institutional format. • Precise definition of the research problem and central thesis question. • Formulation of hypothesis, general and specific verifiable objectives. • Activity schedule with milestones through the final thesis defense. • Feasibility analysis: resources, timeline, risks, and contingencies of the project.	• Produce the thesis plan in the format required by the academic institution. [Evaluar] • Define an original research problem with verifiable hypothesis and objectives. [Evaluar] • Develop a detailed schedule with milestones and risk analysis through the thesis defense. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 2: Theoretical Framework and Advanced State of the Art (10 hours)	
Competences Expected: AG-C04,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Thematic organization and critical synthesis of the literature in the thesis area. • Writing the theoretical framework chapter in LaTeXwith coherent structure. • Identification and justification of research gaps specific to the problem. • Comparative analysis of approaches: classification tables and diagrams of the literature. • Reference integration with BibTeX and version control with Git.	• Write the theoretical framework chapter with structured and critical synthesis of the literature. [Evaluar] • Justify the originality of the problem in relation to the state of the art. [Evaluar] • Build comparative tables that evidence the research gaps of the area. [Evaluar]
Readings : [**ieee**], [**acm**], [**citeseer**]	

Unit 3: Original Methodology Design (10 hours)	
Competences Expected: AG-C05,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Definition of the methodological approach: experimental, analytical, or constructive. • Specification of the original contribution: algorithm, system, model, or methodology. • Experimental protocol design: variables, metrics, datasets, and benchmarks. • Validation criteria and generalization of the proposal's results. • Alignment of methodology with the objectives and hypothesis of the thesis plan.	• Design the research methodology aligned with the hypothesis and objectives. [Evaluar] • Specify the original contribution of the thesis project with verifiable criteria. [Evaluar] • Define the complete experimental protocol with variables, metrics, and benchmarks. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 4: Implementation and Preliminary Experiments (12 hours)	
Competences Expected: AG-C05,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Implementation of the original proposal: prototype, algorithm, or thesis model. • Reproduction of relevant baselines from the state of the art for comparison. • Execution of preliminary experiments with basic statistical analysis. • Recording and documentation of configurations, versions, and results. • Identification of limitations and iterative refinement of the proposal.	• Implement a functional prototype of the original thesis proposal. [Usar] • Execute preliminary comparative experiments with defined baselines. [Usar] • Analyze and interpret results with appropriate statistical rigor. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 5: Writing the Core Thesis Chapters (10 hours)	
Competences Expected: AG-C02,AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Writing the methodology and experimental design chapter in LaTeX. • Writing the related works chapter with critical comparative analysis. • Presentation of preliminary results: tables, figures, and comparative analysis. • Revision cycles with the advisor: structured incorporation of feedback. • Narrative coherence and consistency across thesis chapters.	• Write the methodology and related works chapters with academic rigor. [Evaluar] • Present preliminary results with clear visualizations and comparative analysis. [Usar] • Apply revision cycles with the advisor to improve document quality and coherence. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 6: Progress Presentation and Final Planning (8 hours)	
Competences Expected: AG-C02,AG-C04,AG-C05	
Topics	**Learning Outcomes**
• Oral presentation of thesis progress before the evaluation committee. • Defense of methodology and preliminary results before committee questions. • Incorporation of committee feedback into the document revision. • Planning of remaining activities toward the final thesis defense. • Research ethics management: authorship, academic integrity, and data confidentiality.	• Present and orally defend the thesis progress before an academic committee. [Evaluar] • Incorporate committee feedback into a well-grounded revision of the document. [Evaluar] • Plan the remaining activities with clear objectives toward the final defense. [Usar]
Readings : [**ieee**], [**acm**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI365. Advanced Generative AI Models (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	AI365. Advanced Generative AI Models
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	AI264. Deep Learning. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course explores modern generative models (GANs, Diffusion Models, LLMs) for multimodal content creation, covering both mathematical foundations and practical applications with ethical considerations for synthetic media.

5. GOALS

- Implement image/text/video generation pipelines
- Design AI-assisted editing systems
- Evaluate synthetic content risks

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

7. TOPICS

Unit 1: Generative Fundamentals (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• GANs theory (minimax, Wasserstein) • Autoregressive models (PixelCNN, Transformers) • Evaluation metrics (FID, Inception Score)	• Train basic GANs [Usar] • Analyze generation failure modes [Evaluar]
Readings : [**goodfellow2014gan**]	

Unit 2: Diffusion Models (24 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Diffusion processes (forward/reverse) • Stable Diffusion and latent architectures • Control techniques (ControlNet, LoRA)	• Implement diffusion samplers [Usar] • Fine-tune for domain-specific generation [Evaluar]
Readings : [**sohl-dickstein2023diffusion**], [**rombach2022highres**]	

Unit 3: Synthetic Media Ethics (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Deepfakes and misinformation • Copyright in generated content • Authenticity marking (C2PA, watermarking)	• Detect synthetic content [Familiarizarse] • Implement traceability systems [Usar]
Readings : [**sohl-dickstein2023diffusion**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CB309. Bioinformatics (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CB309. Bioinformatics
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The use of computational methods in the biological sciences has become one of the key tools for the field of molecular biology, being a fundamental part of research in this area.
In Molecular Biology, there are several applications that involve both DNA, protein analysis or sequencing of the human genome, which depend on computational methods. Many of these problems are really complex and deal with large data sets.
This course can be used to see concrete use cases of several areas of knowledge of Computer Science such as Programming Languages (PL), Algorithms and Complexity (AL), Probabilities and Statistics, Information Management (IM), Intelligent Systems (IS).

5. GOALS

- That the student has a solid knowledge of molecular biological problems that challenge computing.
- That the student is able to abstract the essence of the various biological problems to pose solutions using their knowledge of Computer Science

6. COMPETENCES

AG-C10) Inquiry: Studies complex computing problems using information science methods. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Introduction to Molecular Biology (4 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Review of organic chemistry: molecules and macromolecules, sugars, nucleic acids, nucleotides, RNA, DNA, proteins, amino acids and levels of structure in proteins. • The Dogma of Life: From DNA to Proteins, Transcription, Translation, Protein Synthesis. • Genome study: Maps and sequences, specific techniques	• Achive a general knowledge of the most important topics in Molecular Biology. [Familiarizarse] • Understand that biological problems are a challenge to the computational world. [Evaluar]
Readings : [**Clote2000**], [**Setubal1997**]	

Unit 2: Sequence Comparison (4 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Sequences of nucleotides and amino acid sequences. • Sequence alignment, paired alignment problem, exhaustive search, Dynamic programming, global alignment, local alignment, gaps penalty • Comparison of multiple sequences: sum of pairs, complexity analysis by dynamic programming, alignment heuristics, star algorithm, progressive alignment algorithms.	• Understand and solve the problem of aligning a pair of sequences. [Usar] • Understand and solve the problem of multiple sequence alignment. [Usar] • Know the various algorithms for aligning existing sequences in the literature . [Familiarizarse]
Readings : [**Clote2000**], [**Setubal1997**], [**Pevzner2000**]	

Unit 3: Phylogenetic Trees (4 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Phylogeny: Introduction and phylogenetic relations • Phylogenetic trees: definition, type of trees, problem of search and reconstruction of trees • Reconstruction methods: parsimony methods, distance methods, maximum likelihood methods, confidence of reconstructed trees	• Understand the concept of phylogeny, phylogenetic trees and the methodological difference between biology and molecular biology. [Familiarizarse] • Understand the problem of the reconstruction of phylogenetic trees, to know and apply the main algorithms for the reconstruction of phylogenetic trees. [Evaluar]
Readings : [**Clote2000**], [**Setubal1997**], [**Pevzner2000**]	

Unit 4: DNA Sequence Assembling (4 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Biological basis: ideal case, difficulties, alternative methods for DNA sequencing • Formal Assembly Models: Shortest Common Superstring, Reconstruction, Multicontig • Algorithms for sequence assembly: representation of overlaps, paths to create superstrings, voracious algorithm, acyclic graphs. • Assembly heuristics: search for overlays, ordering fragments, alignments and consensus.	• Understand the computational challenge of the Sequence Assembly problem. [Familiarizarse] • Understand the principle of formal model for assembly. [Evaluar] • Know the main heuristics for the problem of assembjale of DNA sequences[Usar]
Readings : [**Setubal1997**], [**Aluru2006**]	

Unit 5: Secondary and tertiary structures (4 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Molecular structures: primary, secondary, tertiary, quaternary. • Prediction of secondary structures of RNA: formal model, pair energy, structures with independent bases, solution with Dynamic Programming, structures with loops. • *Protein folding*: Estructuras en proteinas, problema de protein folding. • *Protein Threading*: Definitions, Branch Bound Algorithm, Branch Bound for protein threading. • *Structural Alignment*: Definitions, DALI algorithm	• Know the protein structures and the necessity of computational methods for the prediction of the geometry. [Familiarizarse] • Know the algorithms for solving prediction problems of secondary structures RNA, and structures in proteins. [Evaluar]
Readings : [**Setubal1997**], [**Clote2000**], [**Aluru2006**]	

Unit 6: Probabilistic Models in Molecular Biology (4 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Probability: Random Variables, Markov Chains, Metropoli-Hasting Algorithm, Markov Random Fields, and Gibbs Sampler, Maximum Likelihood. • Hidden Markov Models (HMM), parameter estimation, Viterbi algorithm and Baul-Welch method, Application in paired and multiple alignments, Motifs detection in proteins, in eukaryotic DNA, in sequences families. • Probabilistic phylogeny: probabilistic models of evolution, likelihood of alignments, likelihood for inference, comparison of probailistic and non-probabilistic methods	• Review concepts of Probabilistic Models and understand their importance in Computational Molecular Biology. [Evaluar] • Know and apply Hidden Markov Models for various analyzes in Molecular Biology.. [Usar] • Know the application of probabilistic models in Phylogeny and to compare them with non-probabilistic models[Evaluar]
Readings : [**Durbin1998**], [**Clote2000**], [**Aluru2006**], [**Krogh1994**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

ET301. Entrerpreneurship II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	ET301. Entrerpreneurship II
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	ET201. Entrerpreneurship I. (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Este curso tiene como objetivo dotar al futuro profesional de conocimientos, actitudes y aptitudes que le permitan formar su propia empresa de desarrollo de software y/o consultoría en informática. El curso está dividido en tres unidades: Valorización de Proyectos, Marketing de Servicios y Negociaciones. En la primera unidad se busca que el alumno pueda analizar y tomar decisiones en relación a la viabilidad de un proyecto y/o negocio.

En la segunda unidad se busca preparar al alumno para que este pueda llevar a cabo un plan de marketing satisfactorio del bien o servicio que su empresa pueda ofrecer al mercado. La tercera unidad busca desarrollar la capacidad negociadora de los participantes a través del entrenamiento vivencial y práctico y de los conocimientos teóricos que le permitan cerrar contrataciones donde tanto el cliente como el proveedor resulten ganadores. Consideramos estos temas sumamente críticos en las etapas de lanzamiento, consolidación y eventual relanzamiento de una empresa de base tecnológica.

5. GOALS

- Que el alumno comprenda y aplique la terminología y conceptos fundamentales de ingeniería económica que le permitan valorizar un proyecto para tomar la mejor decisión económica.
- Que el alumno adquiera las bases para formar su propia empresa de base tecnológica.

6. COMPETENCES

No items for this table

7. TOPICS

Unit 1: (20 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Introducción • Proceso de toma de decisiones • El valor del dinero en el tiempo • Tasa de interés y tasa de rendimiento • Interés simple e interés compuesto • Identificación de costos • Flujo de Caja Neto • Tasa de Retorno de Inversión (TIR) • Valor Presente Neto (VPN) • Valorización de Proyectos	• Permitir al alumno tomar decisiones sobre como invertir mejor los fondos disponibles, fundamentadas en el análisis de los factores tanto económicos como no económicos que determinen la viabilidad de un emprendimiento. [Evaluar]
Readings : [**blank06**]	

Unit 2: (30 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Introducción • Importancia del marketing en las empresas de servicios • El Proceso estratégico. • El Plan de Marketing • Marketing estratégico y marketing operativo • Segmentación, targeting y posicionamiento de servicios en mercados competitivos • Ciclo de vida del producto • Aspectos a considerar en la fijación de precios en servicios • El rol de la publicidad, las ventas y otras formas de comunicación • El comportamiento del consumidor en servicios • Fundamentos de marketing de servicios • Creación del modelo de servicio • Gestión de la calidad de servicio	• Brindar las herramientas al alumno para que pueda identificar, analizar y aprovechar las oportunidades de marketing que generan valor en un emprendimiento. [Usar] • Lograr que el alumno conozca, entienda e identifique criterios, habilidades, métodos y procedimientos que permitan una adecuada formulación de estrategias de marketing en sectores y medios específicos como lo es una empresa de base tecnológica. [Usar]
Readings : [**kotler06**], [**love09**]	

Unit 3: (10 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Introducción. ¿Qué es una negociación? • Teoría de las necesidades de la negociación • La proceso de la negociación • Estilos de negociación • Teoría de juegos • El método Harvard de negociación	• Conocer los puntos clave en el proceso de negociación. [Usar] • Establecer una metodología de negociación eficaz. [Usar] • Desarrollar destrezas y habilidades que permitan llevar a cabo una negociación exitosa. [Usar]
Readings : [**fish96**], [**dasi06**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI369. Robotics (Elective)

2. GENERAL INFORMATION

2.1 Course	:	AI369. Robotics
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	AI268. Computational Vision. (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Robotics is the physical manifestation of Artificial Intelligence. This course introduces the fundamental concepts of robot design, including kinematics, perception, and control. Students will explore how robots interact with the physical world through sensors and actuators, and how they make autonomous decisions using probabilistic mapping and localization techniques. The course bridges the gap between abstract algorithms and physical execution.

5. GOALS

- Understand the mechanical and mathematical foundations of robot kinematics.
- Master sensor fusion techniques for robot perception.
- Implement navigation and path planning algorithms.
- Analyze the integration of machine learning in robotic control.
- Design and simulate autonomous agents for specific environments.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

7. TOPICS

Unit 1: Robotics (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Overview: problems and progress: – State-of-the-art robot systems, including their sensors and an overview of their sensor processing – Robot control architectures, e.g., deliberative vs reactive control and Braitenberg vehicles – World modeling and world models – Inherent uncertainty in sensing and in control • Sensors and effectors: – Sensors: e.g., LIDAR, sonar, vision, depth, stereoscopic, event cameras, microphones, haptics – Effectors: e.g., wheels, arms, grippers • Coordinate frames, translation, and rotation (2D and 3D) • Configuration space and environmental maps • Interpreting uncertain sensor data • Localization and mapping • Navigation and control • Forward and inverse kinematics • Motion path planning and trajectory optimization • Manipulation and grasping • Joint control and dynamics • Vision-based control • Multiple-robot coordination and collaboration • Human-robot interaction: – Shared workspaces – Human-robot teaming and physical HRI – Social assistive robots – Motion/task/goal prediction – Collaboration and communication (explicit vs implicit, verbal or symbolic vs non-verbal or visual) – Trust • Applications and Societal, Economic, and Ethical Issues: – Societal, economic, right-to-work implications – Ethical and privacy implications of robotic applications – Liability in autonomous robotics – Autonomous weapons and ethics – Human oversight and control	• List capabilities and limitations of today's state-of-the-art robot systems, including their sensors and the crucial sensor processing that informs those systems [List] • Integrate sensors, actuators, and software into a robot designed to undertake a specific task [Apply] • Program a robot to accomplish simple tasks using deliberative, reactive, and/or hybrid control architectures [Apply] • Implement fundamental motion planning algorithms within a robot configuration space [Apply] • Characterize the uncertainties associated with common robot sensors and actuators; articulate strategies for mitigating these uncertainties [Characterize] • List the differences among robots' representations of their external environment, including their strengths and shortcomings [List] • Compare and contrast at least three strategies for robot navigation within known and/or unknown environments, including their strengths and shortcomings [Compare] • Describe at least one approach for coordinating the actions and sensing of several robots to accomplish a single task [Describe] • Compare and contrast a multi-robot coordination and a human-robot collaboration approach and attribute their differences to differences between the problem settings [Compare] • Analyze the societal, economic, and ethical issues of a real-world robotics application [Analyze]

Unit 2: Perception and Computer Vision (12 hours)

Competences Expected: AG-C09,AG-C11

Topics	**Learning Outcomes**
• Computer vision: – Image acquisition, representation, processing, and properties – Shape representation, object recognition, and segmentation – Motion analysis – Generative models • Audio and speech recognition • Touch and proprioception • Other modalities (e.g., olfaction) • Modularity in recognition • Approaches to pattern recognition: – Classification algorithms and measures of classification quality – Statistical techniques – Deep learning techniques	• Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize] • List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List] • Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply] • Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish] • Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate] • Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply] • Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply] • Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply] • Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate] • Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe] • Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]

Readings : [**Thrun2005**], [**Siegwart2011**]

Unit 3: Robotics (24 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Overview: problems and progress: – State-of-the-art robot systems, including their sensors and an overview of their sensor processing – Robot control architectures, e.g., deliberative vs reactive control and Braitenberg vehicles – World modeling and world models – Inherent uncertainty in sensing and in control • Sensors and effectors: – Sensors: e.g., LIDAR, sonar, vision, depth, stereoscopic, event cameras, microphones, haptics – Effectors: e.g., wheels, arms, grippers • Coordinate frames, translation, and rotation (2D and 3D) • Configuration space and environmental maps • Interpreting uncertain sensor data • Localization and mapping • Navigation and control • Forward and inverse kinematics • Motion path planning and trajectory optimization • Manipulation and grasping • Joint control and dynamics • Vision-based control • Multiple-robot coordination and collaboration • Human-robot interaction: – Shared workspaces – Human-robot teaming and physical HRI – Social assistive robots – Motion/task/goal prediction – Collaboration and communication (explicit vs implicit, verbal or symbolic vs non-verbal or visual) – Trust • Applications and Societal, Economic, and Ethical Issues: – Societal, economic, right-to-work implications – Ethical and privacy implications of robotic applications – Liability in autonomous robotics – Autonomous weapons and ethics – Human oversight and control	• List capabilities and limitations of today's state-of-the-art robot systems, including their sensors and the crucial sensor processing that informs those systems [List] • Integrate sensors, actuators, and software into a robot designed to undertake a specific task [Apply] • Program a robot to accomplish simple tasks using deliberative, reactive, and/or hybrid control architectures [Apply] • Implement fundamental motion planning algorithms within a robot configuration space [Apply] • Characterize the uncertainties associated with common robot sensors and actuators; articulate strategies for mitigating these uncertainties [Characterize] • List the differences among robots' representations of their external environment, including their strengths and shortcomings [List] • Compare and contrast at least three strategies for robot navigation within known and/or unknown environments, including their strengths and shortcomings [Compare] • Describe at least one approach for coordinating the actions and sensing of several robots to accomplish a single task [Describe] • Compare and contrast a multi-robot coordination and a human-robot collaboration approach and attribute their differences to differences between the problem settings [Compare] • Analyze the societal, economic, and ethical issues of a real-world robotics application [Analyze]

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS351. Topics in Computer Graphics (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS351. Topics in Computer Graphics
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS251. Computer Graphics . (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Modern computer graphics go beyond simple image generation, encompassing real-time interaction, physical simulation, and immersive environments. This course explores advanced topics in the graphics pipeline, focusing on programmable shading, complex geometric modeling, and computer animation. Additionally, it addresses the intersection of graphics with physical computing and the ethical implications of synthetic content generation.

5. GOALS

- Master advanced shader programming and non-photorealistic rendering.
- Implement physically-based simulations and complex animations.
- Design immersive and interactive experiences using tangible computing.
- Evaluate the ethical and societal impact of computer-generated imagery.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Shading and Advanced Rendering (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Advanced shading: physically based shading, subsurface scattering • Global illumination: radiosity, photon mapping, path tracing • Real-time rendering techniques: level of detail, culling, occlusion • GPU programming and shaders	• Implement advanced shading models for realistic materials [Implement] • Compare global illumination algorithms and their trade-offs [Compare] • Optimize real-time rendering using techniques like level of detail and culling [Create] • Develop GPU shaders for custom rendering effects [Create]
Readings : [**Marschner2021**], [**Shirley2021**]	

Unit 2: Geometric Modeling (10 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Geometric representations: polygons, curves, surfaces, implicit forms • Modeling techniques: splines, subdivision surfaces, constructive solid geometry • Procedural modeling and generation • Geometric algorithms: intersection, distance, decomposition	• Describe different geometric representations and their trade-offs [Describe] • Implement modeling techniques such as spline curves or subdivision surfaces [Implement] • Create procedural models using algorithms or rules [Create] • Apply geometric algorithms to solve problems like intersection testing or mesh processing [Apply]
Readings : [**Marschner2021**]	

Unit 3: Computer Animation (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Animation principles: timing, spacing, anticipation, follow-through • Keyframe animation and interpolation • Procedural animation: particle systems, flocking, physics-based • Character animation: skeletal animation, skinning, inverse kinematics • Motion capture and retargeting	• Apply animation principles to create believable motion [Apply] • Implement keyframe animation with interpolation techniques [Implement] • Create procedural animations such as particle systems or physics-based simulations [Create] • Implement character animation using skeletal systems and skinning [Implement] • Process motion capture data for animation retargeting [Implement]
Readings : [**Parent2012**]	

Unit 4: Simulation (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Physics-based simulation: rigid bodies, fluids, cloth • Numerical methods for simulation: integration, collision detection • Real-time simulation constraints and optimizations • Applications in games, visual effects, and scientific visualization	• Implement physics-based simulations for rigid bodies or fluids [Implement] • Apply numerical integration methods for dynamic simulations [Apply] • Optimize simulations for real-time performance [Create] • Develop simulations for specific applications such as games or scientific visualization [Create]
Readings : [**Parent2012**]	

Unit 5: Immersion (5 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Virtual reality (VR) and augmented reality (AR) concepts • Immersive display technologies: head-mounted displays, CAVEs • Tracking and input for immersive environments • Presence and comfort in immersive systems	• Explain the core concepts and differences between VR and AR [Explain] • Evaluate immersive display technologies and their applications [Evaluate] • Implement tracking and input solutions for immersive environments [Implement] • Design immersive experiences that maximize presence and minimize discomfort [Design]
Readings : [**LaValle2023**]	

Unit 6: Interaction (5 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Interaction paradigms: direct manipulation, gesture-based, tangible • User interface design for graphics applications • Multimodal interaction: combining visual, auditory, haptic feedback • Evaluation of interaction techniques: usability, user experience	• Compare different interaction paradigms and their suitability for various tasks [Compare] • Design user interfaces for graphics-intensive applications [Design] • Implement multimodal interaction systems incorporating multiple feedback channels [Implement] • Conduct usability evaluations of interaction techniques [Evaluate]
Readings : [**LaValle2023**]	

Unit 7: Tangible/Physical Computing (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Tangible interfaces and physical computing • Sensors and actuators for interactive systems • Embedded systems for interactive art and design • Interactive installations and environments	• Design tangible interfaces that bridge physical and digital interactions [Design] • Integrate sensors and actuators to create responsive systems [Integrate] • Develop embedded systems for interactive art or design projects [Create] • Create interactive installations that engage users in physical spaces [Create]
Readings : [**Noble2012**]	

Unit 8: Society, Ethics, and the Profession (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Ethical considerations in graphics and interaction: representation, accessibility, privacy • Social impact of immersive technologies and visualization • Professional practice in graphics and interactive industries	• Analyze ethical issues related to representation, accessibility, and privacy in graphics and interaction [Analyze] • Discuss the social implications of immersive technologies and data visualization [Debate] • Describe professional practices and standards in graphics and interactive industries [Describe]
Readings : [**Marschner2021**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS353. Quantum Computing (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS353. Quantum Computing
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS3I1. Computer Security. (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course introduces the fundamental principles of quantum computing, including qubits, superposition, entanglement, quantum algorithms, and hardware. Students will contrast classical and quantum models, exploring applications in cryptography, optimization, physical simulation, and quantum machine learning, using frameworks like Qiskit or Cirq. Includes a practical module on hardware architectures (superconductors, ion traps) and their impact on algorithm design.

5. GOALS

- Understand the postulates of quantum mechanics applied to computing.
- Implement basic quantum circuits and optimization algorithms.
- Analyze the ethical, technical, and commercial impact of quantum computing.
- Evaluate hardware limitations in real-world applications.

6. COMPETENCES

AG-C10) Inquiry: Studies complex computing problems using information science methods. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Foundational concepts (10 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Qubits and Bloch sphere representation. • Postulates of quantum mechanics. • Quantum gates (Hadamard, CNOT, Toffoli).	• Mathematically model a qubit [Usar]. • Simulate basic operations in Qiskit [Evaluar].
Readings : [**NielsenChuang**], [**QiskitTextbook**]	

Unit 2: Quantum Architectures (10 hours)

Competences Expected: AG-C10,AG-C11

Topics	**Learning Outcomes**
• Principles: – The wave-particle duality principle – The uncertainty principle in the double-slit experiment – What is a Qubit? Superposition, interference, and measurement. Photons as qubits – Systems of two qubits, Entanglement, Bell states, The No-Signaling theorem • Axioms of QM: superposition principle, measurement axiom, unitary evolution • Single qubit gates for the circuit model of quantum computation: X, Z, H • Two qubit gates and tensor products, working with matrices • The No-Cloning Theorem. The Quantum Teleportation protocol • Algorithms: – Simple quantum algorithms: Bernstein-Vazirani, Simon's algorithm – Implementing Deutsch-Josza with Mach-Zehnder Interferometers – Quantum factoring (Shor's Algorithm) – Quantum search (Grover's Algorithm) • Implementation aspects: – The physical implementation of qubits – Classical control of a Quantum Processing Unit (QPU) – Error mitigation and control, NISQ and beyond – Measurement approaches • Emerging Applications: – Post-quantum encryption – The Quantum Internet – Adiabatic quantum computation (AQC) and quantum annealing	• Discuss how a quantum object produced as a particle propagates like a wave and is detected as a particle with a probability distribution corresponding to the wave [Debate] • Discuss the quantum-level nature that is inherently probabilistic [Debate] • Express your view on entanglement that can be used to create non-classical correlations, but there is no way to use quantum entanglement to send messages faster than the speed of light [Evaluate] • Describe quantum parallelism and the role of constructive vs destructive interference in quantum algorithms given the probabilistic nature of measurement(s) [Describe] • Analyze a code snippet providing the role of quantum Fourier transform (QFT) in Shor's algorithm [Analyze] • Write a program to implement Shor's algorithm in a simulator, highlighting the classical components and aspects of Shor's algorithm [Write] • Enumerate the specifics of each qubit modality (e.g., trapped ion, superconducting, silicon spin, photonic, quantum dot, neutral atom, topological, color center, electron-on-helium) [Enumerate] • Contrast AQC with the gate model of quantum computation and the problems each is better suited to solve [Contrast]

Readings : [**QuantumHardwareBook**], [**IBMQuantumRoadmap2023**]

Unit 3: Algorithmic Strategies (12 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Quantum computing • Deutsch-Jozsa algorithm. • Grover's search (applied to combinatorial optimization). • Shor's factorization and its impact on RSA.	• Implement Grover for SAT problems [Usar].
Readings : [**PreskillNotes**], [**QiskitTextbook**]	

Unit 4: Formal Languages and Automata (10 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Quantum computation: – Postulates of quantum mechanics: ∗ State space ∗ State evolution ∗ State composition ∗ State measurement – Column vector representations of qubits – Matrix representations of quantum operations – Simple quantum gates (e.g., XNOT, CNOT) • Quantum Approximate Optimization Algorithm (QAOA). • Logistics/finance applications (Portfolio Optimization).	• For a quantum system give examples that explain the following postulates: – State Space - system state represented as a unit vector in Hilbert space – State Evolution - the use of unitary operators to evolve system state – State Composition - the use of tensor product to compose systems states – State Measurement - the probabilistic output of measuring a system state. [Familiarizarse]. • Design a QAOA for routing problems [Evaluar].
Readings : [**QMLBook**], [**Bravyi2022**]	

Unit 5: Ethics and Post-Quantum Security (6 hours)	
Competences Expected: AG-C10,AG-C11	
Topics	**Learning Outcomes**
• Post-quantum cryptography (NIST PQC Standardization). • Quantum supremacy: technical/ethical debate. • Patents and quantum geopolitics.	• Propose post-quantum migration strategies [Familiarizarse].
Readings : [**QuantumSupremacyPaper**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS392. Tópicos en Ingeniería de Software (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS392. Tópicos en Ingeniería de Software
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS391. Software Engineering III. (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Modern software development requires rigorous engineering practices to ensure quality, maintainability, and reliability. This course explores advanced topics in the software lifecycle, focusing on team collaboration, formal verification, and advanced refactoring. Students will master the use of professional tools to design, construct, and validate complex systems while ensuring high standards of reliability and formal correctness.

5. GOALS

- Apply advanced software construction and refactoring techniques to improve code quality.
- Use formal methods and validation strategies to ensure system reliability.
- Manage complex requirements and design architectures that meet stakeholder needs.
- Lead software teams using professional tools and collaborative workflows.

6. COMPETENCES

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Team Collaboration and Dynamics (4 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture • Common causes of team conflict, and approaches for conflict resolution. • Cooperative programming: – Pair programming or Swarming – Code review – Collaboration through version control • Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics – Advantages of teamwork – Risks and complexity of such collaboration • Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule	• Follow effective team communication practices [Analyze] • Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate] • Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze] • Collaborate effectively in cooperative development/programming [Analyze] • Propose and delegate necessary roles and responsibilities in a software development team [Propose] • Compose and follow an agenda for a team meeting [Compose] • Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]
Readings : [**Sommerville2015**]	

Unit 2: Version Control and CI/CD (4 hours)

Competences Expected: AG-C03,AG-C09

Topics	Learning Outcomes
• Software configuration management and version control: Software Development Practices – Configuration in version control, reproducible builds/configuration. – Version control branching strategies. Development branches vs release branches. Trunk-based development. – Merging/rebasing strategies, when relevant. • Release management. • Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security • Software process automation: – Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. – Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. – Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. – Dependency management - updating external/upstream dependencies, package management, SemVer. • Collaborative review workflow as a quality gate – Mandatory pull-request review before merging – Code ownership via CODEOWNERS-style rules – Conventional commit message conventions – Branch-protection rules enforcing checks before merge • Dependency and supply-chain vulnerability scanning integrated into CI pipelines	• Describe the difference between centralized and distributed software configuration management [Describe] • Describe how version control can be used to help manage software release management [Describe] • Identify configuration items and use a source code control tool in a small team-based project [Analyze] • Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] • Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] • Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] • Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Readings : [**Pressman2014**]

Unit 3: Requirements Engineering (6 hours)

Competences Expected: AG-C03,AG-C09

Topics	**Learning Outcomes**
• Describe functional requirements using, for example, use cases or user stories. – Using at least one method of documenting and structuring functional requirements. – Understanding how the method supports design and implementation. – Strengths and weaknesses of using a specific approach. • Properties of requirements including consistency, validity, completeness, and feasibility. • Requirements elicitation. – Sources of requirements, for example, users, administrators, or support personnel. – Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis. • Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context • Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics • Communicating and/or formalizing requirement specifications.	• Compare different methods of eliciting requirements along multiple axes [Compare] • Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze] • Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze] • Collect a set of requirements for a simple software system [Analyze] • Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze] • Identify the functional and non-functional requirements in a set of requirements [Analyze]
Readings : [**Sommerville2015**]	

Unit 4: Principles and Software Architecture (6 hours)

Competences Expected: AG-C03,AG-C09

Topics	**Learning Outcomes**
• System design principles. System Reliability – Levels of abstraction (e.g., architectural design and detailed design) – Separation of concerns – Information hiding – Coupling and cohesion • Software architecture. System Reliability – Design paradigms * Top-down functional decomposition/layered design * Data-oriented architecture * Object-oriented analysis and design * Event-driven design – Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) – Identifying component boundaries and dependencies • Programming in the large vs programming in the small. System Reliability • Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering	• Identify the standard software architecture of a given high-level design [Analyze] • Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] • Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] • Identify the dependencies among a set of software components in an architectural design [Analyze]

Readings : [**Sommerville2015**]

Unit 5: Coding Practices (6 hours)

Competences Expected: AG-C03,AG-C09

Topics	Learning Outcomes
• Practical small-scale testing Software Development Practices – Unit testing – Test-driven development - This is particularly valuable for students psychologically, as it is far easier to engage constructively with the challenge of identifying challenging inputs for a given API (edge cases, corner cases) a priori. If they implement first, the instinct is often to avoid trying to crash their new creation, while a test-first approach gives them the intellectual satisfaction of spotting the problem cases and then watching as more tests pass during the development process. • Documentation Software Development Practices – Interface documentation - describe interface requirements, potentially including (formal or informal) contracts, pre and post conditions, invariants. – Implementation documentation should focus on tricky and non-obvious pieces of code, whether because the code is using advanced language features, or the behavior of the code is complex. (Do not add comments that re-state common/obvious operations and simple language features.) ∗ Clarify dataflow, computation, etc., focusing on what the code is. ∗ Identify subtle/tricky pieces of code and refactor to be self-explanatory if possible or provide appropriate comments to clarify. • Coding style Software Development Practices – Style guides – Commenting – Naming • ”Best Practices” for coding: techniques, idioms/patterns, mechanisms for building quality programs Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security – Defensive coding practices – Secure coding practices and principles – Using exception handling mechanisms to make programs more robust, fault-tolerant • Debugging Software Development Practices • Logging • Use of libraries and frameworks developed by others	• Write appropriate unit tests for a small component (several functions, a single type, etc.) [Write] • Write appropriate interface and (if needed) implementation comments for a small component [Write] • Describe techniques, coding idioms and mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness [Describe] • Write robust code using exception handling mechanisms [Write] • Describe secure coding and defensive coding practices [Describe] • Select and use a defined coding standard in a small software project [Use] • Compare and contrast integration strategies including top-down, bottom-up, and sandwich integration [Compare] • Describe the process of analyzing and implementing changes to code base developed for a specific project [Describe] • Describe the process of analyzing and implementing changes to a large existing code base [Describe]

Unit 6: API Compatibility and Versioning (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Hyrum's Law/The Law of Implicit Interfaces • Backward compatibility – Compatibility is not a property of a single entity, it's a property of a relationship. – Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises. • Versioning – Semantic Versioning (SemVer) – Trunk-based development	• Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze] • Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze] • Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]
Readings : [**Fowler2018**]	

Unit 7: Verification and validation Concepts (6 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation concepts – Verification: Are we building the thing right? – Validation: Did we build the right thing? • Why testing matters: Does the component remain functional as the code evolves? • Testing objectives – Usability – Reliability – Conformance to specification – Performance – Security	• Explain why testing is important [Explain] • Distinguish between program validation and verification [Distinguish] • Describe different objectives of testing [Describe]
Readings : [**Pressman2014**]	

Unit 8: Testing Tools and Analysis (3 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Verification and validation tooling and automation – Static analysis – Code coverage – Fuzzing – Dynamic analysis and fault containment (sanitizers, etc.) – Fault logging and fault tracking • Test planning and generation – Fault estimation and testing termination including defect seeding – Use of random and pseudo random numbers in testing • Testing asynchronous, parallel, and concurrent systems • Verification and validation of non-code artifacts (documentation, training materials) • Behavior-driven development (BDD) – Expressing acceptance criteria as Gherkin Given/When/Then scenarios – Executing BDD scenarios as automated end-to-end tests • Mutation testing and static code-quality analysis – Mutation testing to assess test-suite effectiveness – Static detection of code smells and excessive cyclomatic complexity	• Describe and compare different tools for verification and validation [Describe] • Automate the use of different tools in a small software project [Design] • Explain how and when random numbers should be used in testing [Explain] • Describe approaches for fault estimation [Describe] • Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] • Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] • Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] • Describe techniques for the verification and validation of non-code artifacts [Describe] • Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] • Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]
Readings : [**Pressman2014**]	

Unit 9: Performance Testing and Benchmarking (3 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Performance testing and benchmarking – Throughput and latency – Degradation under load (stress testing, FIFO vs LIFO handling of requests) – Speedup and scaling ∗ Amdahl's law ∗ Gustafson's law ∗ Soft and weak scaling – Identifying and measuring figures of merits – Common performance bottlenecks ∗ Compute-bound ∗ Memory-bandwidth bound ∗ Latency-bound – Statistical methods and best practices for benchmarking ∗ Estimation of uncertainty ∗ Confidence intervals – Analysis and presentation (graphs, etc.) – Timing techniques	• Describe throughput and latency and provide examples of each [Describe] • Explain speedup and the different forms of scaling and how they are computed [Explain] • Describe common performance bottlenecks [Describe] • Describe statistical methods and best practices for benchmarking software [Describe] • Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] • Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]
Readings : [**Bondi2015**], [**Gregg2020**]	

Unit 10: Reliability Concepts (5 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures • Identifying reliability requirements for different kinds of software • Software failures caused by defects/bugs, and so for high reliability the goal is to have minimum defects - by injecting fewer defects (better training, education, planning), and by removing most of the injected defects (testing, code review, etc.) • Software reliability, system reliability and failure behavior • Defect injection and removal cycle, and different approaches for defect removal • Compare the "error budget" approach to reliability with the "error-free" approach and identify domains where each is relevant.	• Describe how to determine the level of reliability required by a software system [Describe] • Explain the problems that exist in achieving very high levels of reliability [Explain] • Understand approaches to minimizing faults that can be applied at each stage of the software lifecycle [Explain]
Readings : [**Pressman2014**]	

Unit 11: Formal Methods (5 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Formal specification of interfaces – Specification of pre- and post- conditions – Formal languages for writing and analyzing pre- and post-conditions. • Problem areas well served by formal methods – Lock-free programming, data races – Asynchronous and distributed systems, deadlock, livelock, etc. • Comparison to other tools and techniques for defect detection – Testing – Fuzzing • Formal approaches to software modeling and analysis – Model checkers – Model finders	• Describe the role formal specification and analysis techniques can play in the development of complex software and compare their use as validation and verification techniques with testing [Describe] • Apply formal specification and analysis techniques to software designs and programs with low complexity [Apply] • Explain the potential benefits and drawbacks of using formal specification languages [Explain]
Readings : [**Sommerville2015**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.

- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS3P3. Internet of Things (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS3P3. Internet of Things
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS3P1. Parallel and Distributed Computing . (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The Internet of Things (IoT) has transformed how we interact with the physical world by connecting billions of devices. This course explores the architectural foundations of IoT, emphasizing device mobility, wireless communication, and the societal implications of pervasive sensing. Students will analyze the technical challenges of constrained systems and the ethical responsibilities associated with large-scale data collection in interconnected environments.

5. GOALS

- Understand the architecture of IoT systems and the role of mobility in network design.
- Evaluate the societal and ethical impacts of IoT technologies.
- Implement communication protocols for constrained and mobile devices.
- Analyze security risks in the lifecycle of an IoT ecosystem.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Mobility Support (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Principles of cellular communication (e.g., 4G, 5G) • Principles of Wireless LANs (mainly 802.11) • Device to device communication (e.g., IoT communication) • Multi-hop wireless networks (e.g., ad hoc networks, opportunistic, delay tolerant)	• Describe some aspects of cellular communication such as registration [Describe] • Describe how 802.11 supports mobile users [Describe] • Describe practical uses of device-to-device communication, as well as multihop [Describe] • Describe one type of mobile network such as ad hoc [Describe]
Readings : [**Hanes2017**], [**Buyya2016**]	

Unit 2: Security Ethics, Law and Society (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Societal impacts of Internet of Things (IoT) devices and other emerging technologies on security and privacy Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Privacy by design and privacy engineering Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.	• Analyze the societal impacts of pervasive IoT surveillance [Evaluar]. • Apply privacy-by-design principles to IoT data architectures [Usar].
Readings : [**Hanes2017**]	

Unit 3: Networked Applications (24 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination • Protocols for constrained networks (CoAP, MQTT, LoRaWAN).	• Define the principles of naming, addressing, resource location [Define] • Analyze the needs of specific networked application demands [Analyze] • Describe the details of one application layer protocol [Describe] • Implement a simple client-server socket-based application [Implement]
Readings : [**Hanes2017**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS3P2. Cloud Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3P2. Cloud Computing
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	3
2.4 Hours	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS370. Big Data. (9^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Modern computing relies heavily on distributed architectures and cloud-native platforms. This course provides an in-depth exploration of cloud computing services, distributed database concepts, and web platforms. Students will learn to design scalable architectures, understand the implications of distributed data consistency, and evaluate the privacy and security challenges inherent in cloud-based environments.

5. GOALS

- Understand the architecture and deployment models of cloud computing.
- Design distributed database solutions considering consistency and availability.
- Develop applications using modern cloud-based web platforms and frameworks.
- Analyze security and privacy implications in distributed and cloud services.

6. COMPETENCES

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

7. TOPICS

Unit 1: Cloud Architecture and Data Management (12 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Web platforms, frameworks, or meta-frameworks – Cloud services – API, Web Components • Computing services NoSQL Systems – Cloud Hosting – Scalability (e.g., Autoscaling, Clusters) – Cost estimation for services • Architecture – Monoliths vs Microservices – Micro-frontends – Event-Driven vs RESTful architectures: advantages and disadvantages – Serverless, cloud computing on demand	• Design and implement a web-based application using a microservice architecture design [Design] • Describe the constraints, such as hosting, services, and scalability, related to web platforms [Describe]
Readings : [**Marinescu2022**], [**Buyya2013**]	

Unit 2: Distributed Databases/Cloud Computing (10 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Distributed databases/cloud-based systems	• Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe] • Analyze the tradeoffs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze] • Describe data replication strategies and weak consistency models in distributed database systems [Describe]
Readings : [**Buyya2013**]	

Unit 3: Parallel and Distributed Computing (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Safety and liveness: – Race conditions – Dependencies/preconditions – Fault models – Termination • Programming models: – Actor models – Procedural and reactive models – Synchronous/asynchronous programming models – Data parallelism • Properties: – Order-based properties: ∗ Commutativity ∗ Independence – Consistency-based properties: ∗ Atomicity ∗ Consensus • Execution control: – Async await – Promises – Threads • Communication and coordination: – Mutexes – Message-passing – Shared memory – Cobegin-coend – Monitors – Channels – Threads – Guards • Futures • Language support for data parallelism such as forall, loop unrolling, map/reduce • Effect of memory-consistency models on language semantics and correct code generation • Representational State Transfer Application Programming Interfaces (REST APIs) • Technologies and approaches: cloud computing, high performance computing, quantum computing, ubiquitous computing • Overheads of message-passing	• Explain why programming languages do not guarantee sequential consistency in the presence of data races and what programmers must do as a result [Explain] • Implement correct concurrent programs using multiple programming models, such as shared memory, actors, futures, synchronization constructs, and data-parallelism primitives [Implement] • Use a message-passing model to analyze a communication protocol [Use] • Use synchronization constructions such as monitor/synchronized methods in a simple program [Use] • Modeling data dependency using simple programming constructs involving variables, read and write [Implement] • Modeling control dependency using simple constructs such as selection and iteration [Implement] • Explain how REST API's integrate applications and automate processes [Explain] • Explain benefits, constraints and challenges related to distributed and parallel computing [Explain]

Unit 4: Networked Applications (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination	• Define the principles of naming, addressing, resource location [Define] • Analyze the needs of specific networked application demands [Analyze] • Describe the details of one application layer protocol [Describe] • Implement a simple client-server socket-based application [Implement]
Readings : [**Marinescu2022**]	

Unit 5: Algorithms: Application Domains Survey (4 hours)

Competences Expected: AG-C09,AG-C11

Topics	**Learning Outcomes**
• Survey of common application domains across multicore, reactive, data parallel, cluster, cloud, open distributed systems, and frameworks (with reference to the following table). – Multicore: Typical Execution agents: Threads. Typical Communication mechanisms: Shared memory, Atomics, locks. Typical Algorithmic domains: Resource management, data processing. Typical Engineering goals: Throughput, latency, energy. – Reactive: Typical Execution agents: Handlers, threads. Typical Communication mechanisms: I/O Channels. Typical Algorithmic domains: Services, real-time. Typical Engineering goals: Latency. – Data parallel: Typical Execution agents: GPU, SIMD, accelerators, hybrid. Typical Communication mechanisms: Heterogeneous memory. Typical Algorithmic domains: Linear algebra, graphics, data analysis. Typical Engineering goals: Throughput, energy. – Cluster: Typical Execution agents: Managed hosts. Typical Communication mechanisms: Sockets, channels. Typical Algorithmic domains: Simulation, data analysis. Typical Engineering goals: Throughput. – Cloud: Typical Execution agents: Provisioned hosts. Typical Communication mechanisms: Service APIs. Typical Algorithmic domains: Web applications. Typical Engineering goals: Scalability. – Open distributed: Typical Execution agents: Autonomous hosts. Typical Communication mechanisms: Sockets, Data stores. Typical Algorithmic domains: Fault tolerant data stores and services. Typical Engineering goals: Reliability.	• Extend an event-driven sequential program by establishing a new activity in an event handler (for example a new thread in a GUI action handler) [Design] • Improve the performance of a sequential component by introducing parallelism and/or distribution [Create] • Choose among different parallel/distributed designs for components of a given system [Assess]

Readings : [**Marinescu2022**]

Unit 6: Security Mindset and Principles (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• Applications of a security mindset: web, cloud, and mobile devices System Design, Common Aspects: Platforms, APIs and Constraints, Common Aspects: Platform Languages and Web Programming Patterns. • Privacy implications of widespread data collection including but not limited to transactional databases, data warehouses, surveillance systems, cloud computing, and artificial intelligence.	• Analyze the security risks associated with distributed cloud applications [Usar]. • Evaluate the privacy implications of data storage in multi-tenant cloud environments [Evaluar].
Readings : [**Marinescu2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS404. Research Workshop (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS404. Research Workshop
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS403. Capstone Project II. (9^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course aims to enable students to complete their thesis with sufficient quality for public defense, finishing experiments, final chapters, advisor revisions, and the formal institutional submission.

5. GOALS

- Complete the experimental part of the thesis with statistical rigor and full analysis of results.
- Write the final thesis chapters: conclusions, contributions, and future work.
- Ensure the document complies with the institutional format and has complete narrative coherence.
- Incorporate advisor revisions until obtaining an approved document ready for defense.
- Prepare and deliver the formal thesis presentation before the faculty committee.
- The deliverables for this course are:

  Midterm: Advanced draft with completed experiments, introduction, theoretical framework, proposal, and results chapters in LaTeX.

  Final: Complete thesis document approved by the advisor and ready for public defense.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Completion of Experiments and Results Analysis (12 hours)	
Competences Expected: AG-C05,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Full execution of the experimental protocol defined in the thesis plan. • Final statistical analysis: significance tests, confidence intervals, and effect size. • Objective comparison with baselines and state-of-the-art using established metrics. • Interpretation of results: findings, limitations, and threats to validity. • Reproducible documentation: code, datasets, configurations, and results.	• Complete the execution of the experimental protocol covering all defined cases. [Evaluar] • Analyze results with statistical rigor and interpret findings with respect to the hypothesis. [Evaluar] • Document experiments, configurations, and results in a reproducible manner. [Usar]
Readings : [**ieee**], [**acm**]	

Unit 2: Writing the Final Thesis Chapters (12 hours)	
Competences Expected: AG-C02,AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Writing the results chapter: high-quality tables, figures, and comparative analysis. • Writing the conclusions chapter: findings, original contributions, and limitations. • Writing the future work chapter: extensions and open research problems. • Writing the final introduction: motivation, problem statement, and contributions summary. • Writing the executive abstract in both Spanish and English.	• Write the results chapter with clear visualizations and comparative analysis. [Evaluar] • Formulate the conclusions and original contributions of the thesis with precision. [Evaluar] • Write the abstract and final introduction articulating the problem and the contributions. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 3: Document Formatting, Integration, and Quality (10 hours)	
Competences Expected: AG-C02,AG-C04,AG-C09	
Topics	**Learning Outcomes**
• Verification of compliance with the institutional thesis format in LaTeX. • Quality review of figures, tables, and equations in the final document. • Narrative and terminological consistency across all thesis chapters. • Review and update of bibliography: coverage, BibTeX format, and correct citations. • Generation of the final document: PDF, cover page, indices, and appendices.	• Verify that the document fully complies with the institutional thesis format. [Evaluar] • Correct the narrative and terminological consistency across all chapters. [Evaluar] • Produce the final PDF document with cover page, indices, and complete bibliography. [Usar]
Readings : [**ieee**], [**acm**]	

Unit 4: Advisor Review and Correction Cycles (10 hours)	
Competences Expected: AG-C02,AG-C08,AG-C09	
Topics	**Learning Outcomes**
• Structured revision dynamic: submission, review, and response to the advisor. • Incorporation of advisor feedback: prioritization and argued response. • Review of proposal soundness and experimental validity according to the advisor. • Handling academic disagreements with the advisor constructively and ethically. • Advisor approval criteria for authorizing the thesis defense.	• Systematically incorporate advisor observations with argued responses. [Evaluar] • Improve the scientific soundness of the document through revision cycles. [Evaluar] • Obtain formal advisor approval for the thesis defense authorization. [Evaluar]
Readings : [**ieee**], [**acm**]	

Unit 5: Defense Preparation and Rehearsal (8 hours)	
Competences Expected: AG-C02,AG-C04,AG-C05	
Topics	**Learning Outcomes**
• Design of the oral presentation: structure, narrative, and time management. • Preparation of technical slides: figures, results, and key comparisons. • Defense techniques for jury questions: argumentation and objection handling. • Defense rehearsal with the advisor and/or academic peers. • Stress management and effective communication in academic evaluation contexts.	• Design and deliver a clear and structured oral presentation of the thesis work. [Evaluar] • Defend methodology and results before critical questions from the jury. [Evaluar] • Demonstrate full command of the topic through prior defense rehearsals. [Usar]
Readings : [**ieee**], [**acm**]	

Unit 6: Formal Submission and Pre-Defense (8 hours)	
Competences Expected: AG-C02,AG-C04,AG-C05	
Topics	**Learning Outcomes**
• Institutional thesis submission process: documentation and regulatory deadlines. • Pre-defense before faculty authorities: evaluation of the final draft. • Incorporation of pre-defense committee corrections into the document. • Administrative procedures: jury designation and defense scheduling. • Ethics in dissemination: open access, institutional repository, and authorship rights.	• Submit the complete thesis document meeting all institutional requirements. [Evaluar] • Present the final draft before the pre-defense committee and incorporate their corrections. [Evaluar] • Complete the administrative procedures for scheduling the public defense. [Usar]
Readings : [**ieee**], [**acm**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI367. Topics in Artificial Intelligence (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	AI367. Topics in Artificial Intelligence
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	AI264. Deep Learning. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Artificial Intelligence is a rapidly evolving field. This course explores advanced and contemporary topics that are currently shaping the industry and research. Focus is placed on Generative AI, Large Language Models (LLMs), and Advanced Reinforcement Learning. Students will analyze the architectures behind modern AI systems, such as Transformers, and discuss the technical and societal challenges of deploying autonomous intelligent agents at scale.

5. GOALS

- Understand the architecture and training process of Large Language Models.
- Master the principles of Generative AI for text and image synthesis.
- Implement advanced Reinforcement Learning algorithms for decision making.
- Apply prompt engineering and fine-tuning techniques to existing models.
- Critically evaluate the latest research papers and trends in AI.

6. COMPETENCES

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies. (Usage)

7. TOPICS

Unit 1: Machine Learning Fundamentals (6 hours)

Competences Expected: AG-C06

Topics	**Learning Outcomes**
• Definition and examples of a broad variety of machine learning tasks: – Supervised learning: ∗ Classification ∗ Regression – Reinforcement learning – Unsupervised learning: ∗ Clustering • Fundamental ideas: – No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. – Sources of error and undecidability in machine learning • A simple statistical-based supervised learning such as linear regression or decision trees: – Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly • The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): – The bias (underfitting) - variance (overfitting) tradeoff	• Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] • Differentiate the terms of AI, machine learning, and deep learning [Evaluate] • Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] • Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] • Explain how machine learning works as an optimization/search process [Explain]

Readings : [**Goodfellow2016**], [**Foster2022**]

Unit 2: Machine Learning Data, Representations and Evaluation (5 hours)

Competences Expected: AG-C06

Topics	**Learning Outcomes**
• Working with Data: – Data preprocessing: ∗ Importance and pitfalls of preprocessing choices – Handling missing values (imputing, flag-as-missing): ∗ Implications of imputing vs flag-as-missing – Encoding categorical variables, encoding real-valued data – Normalization/standardization – Emphasis on real data, not textbook examples • Representations: – Hypothesis spaces and complexity – Simple basis feature expansion, such as squaring univariate features – Learned feature representations • Machine learning evaluation: – Separation of train, validation, and test sets – Performance metrics for classifiers – Estimation of test performance on held-out data – Tuning the parameters of a machine learning model with a validation set – Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions	• Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] • Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] • Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] • Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]

Readings : [**Goodfellow2016**], [**Foster2022**]

Unit 3: Neural Networks and Ethics (5 hours)	
Competences Expected: AG-C06	
Topics	**Learning Outcomes**
• Basic neural networks: – Fundamentals of understanding how neural networks work and their training process, without details of the calculations – Basic introduction to generative neural networks (e.g., large language models) • Ethics for Machine Learning: – Focus on real data, real scenarios, and case studies – Dataset/algorithmic/evaluation bias and unintended consequences	• Describe the neural network training process and resulting learned representations [Explain]
Readings : [**Goodfellow2016**], [**Foster2022**]	

Unit 4: Machine Learning Formulation and Evaluation (3 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: – Objective function – Gradient descent – Regularization to avoid overfitting (mathematical formulation) • Ensembles of models: – Simple weighted majority combination • Performance evaluation: – Other metrics for classification (e.g., error, precision, recall) – Performance metrics for regressors – Confusion matrix – Cross-validation: ∗ Parameter tuning (grid/random search, via cross-validation)	• Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare] • Evaluate the performance of a simple learning system on a real-world dataset [Evaluate] • Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain] • Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain] • Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]
Readings : [**Goodfellow2016**]	

Unit 5: Deep Learning, RL and Ethics (3 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Deep learning: – Deep feed-forward networks (intuition only, no mathematics) – Convolutional neural networks (intuition only, no mathematics) – Visualization of learned feature representations from deep nets – Other architectures (generative NN, recurrent NN, transformers, etc.) • Overview of reinforcement learning methods • Two or more applications of machine learning algorithms: – E.g., medicine and health, economics, vision, natural language, robotics, game play • Ethics for Machine Learning: – Continued focus on real data, real scenarios, and case studies – Privacy – Fairness – Intellectual property – Explainability	• Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] • Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]
Readings : [**Goodfellow2016**]	

Unit 6: Natural Language Processing (20 hours)

Competences Expected: AG-C06

Topics	**Learning Outcomes**
• Deterministic and stochastic grammars • Parsing algorithms: – CFGs and chart parsers (e.g., CYK) – Probabilistic CFGs and weighted CYK • Representing meaning/Semantics: – Logic-based knowledge representations – Semantic roles – Temporal representations – Beliefs, desires, and intentions • Corpus-based methods • N-grams and HMMs • Smoothing and backoff • Examples of use: POS tagging and morphology • Information retrieval: – Vector space model: ∗ TF & IDF – Precision and recall • Information extraction • Language translation • Text classification, categorization: – Bag of words model • Deep learning for NLP: – RNNs – Transformers – Multi-modal embeddings (e.g., images + text) – Generative language models	• Define and contrast deterministic and stochastic grammars, providing examples to show the adequacy of each [Define] • Simulate, apply, or implement classic and stochastic algorithms for parsing natural language [Apply] • Identify the challenges of representing meaning [Analyze] • List the advantages of using standard corpora. Identify examples of current corpora for a variety of NLP tasks [List] • Identify techniques for information retrieval, language translation, and text classification [Analyze] • Implement a TF/IDF transform, use it to extract features from a corpus, and train an off-the-shelf machine learning algorithm using those features to do text classification [Apply]

Readings : [**Tunstall2022**], [**Russell2020**]

Unit 7: Machine Learning Fundamentals (6 hours)

Competences Expected: AG-C06

Topics	**Learning Outcomes**
• Definition and examples of a broad variety of machine learning tasks: – Supervised learning: ∗ Classification ∗ Regression – Reinforcement learning – Unsupervised learning: ∗ Clustering • Fundamental ideas: – No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. – Sources of error and undecidability in machine learning • A simple statistical-based supervised learning such as linear regression or decision trees: – Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly • The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): – The bias (underfitting) - variance (overfitting) tradeoff	• Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] • Differentiate the terms of AI, machine learning, and deep learning [Evaluate] • Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] • Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] • Explain how machine learning works as an optimization/search process [Explain]

Readings : [**Sutton2018**]

Unit 8: Machine Learning Data, Representations and Evaluation (5 hours)	
Competences Expected: AG-C06	
Topics	**Learning Outcomes**
• Working with Data: – Data preprocessing: ∗ Importance and pitfalls of preprocessing choices – Handling missing values (imputing, flag-as-missing): ∗ Implications of imputing vs flag-as-missing – Encoding categorical variables, encoding real-valued data – Normalization/standardization – Emphasis on real data, not textbook examples • Representations: – Hypothesis spaces and complexity – Simple basis feature expansion, such as squaring univariate features – Learned feature representations • Machine learning evaluation: – Separation of train, validation, and test sets – Performance metrics for classifiers – Estimation of test performance on held-out data – Tuning the parameters of a machine learning model with a validation set – Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions	• Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] • Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] • Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] • Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]
Readings : [**Sutton2018**]	

Unit 9: Neural Networks and Ethics (5 hours)	
Competences Expected: AG-C06	
Topics	**Learning Outcomes**
• Basic neural networks: – Fundamentals of understanding how neural networks work and their training process, without details of the calculations – Basic introduction to generative neural networks (e.g., large language models) • Ethics for Machine Learning: – Focus on real data, real scenarios, and case studies – Dataset/algorithmic/evaluation bias and unintended consequences	• Describe the neural network training process and resulting learned representations [Explain]
Readings : [**Sutton2018**]	

Unit 10: AI Applications and Societal Impact (6 hours)	
Competences Expected: AG-C06	
Topics	**Learning Outcomes**
• At least one application of AI to a specific problem and field, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose at least one for the CS Core): – Formulating and evaluating a specific application as an AI problem: ∗ How to deal with underspecified or ill-posed problems – Data availability/scarcity and cleanliness: ∗ Basic data cleaning and preprocessing ∗ Data set bias – Algorithmic bias – Evaluation bias – Assessment of societal implications of the application • Deployed deep generative models: – High-level overview of deep image generative models (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion, etc.), their uses, and their shortcomings/pitfalls. – High-level overview of large language models (e.g., as of 2023, ChatGPT, Bard, etc.), their uses, and their shortcomings/pitfalls. • Overview of societal impact of AI: – Ethics – Fairness – Trust/explainability – Privacy and usage of training data – Human autonomy and oversight/regulations/legal requirements – Sustainability	• Given a real-world application domain and problem, formulate an AI solution to it, identifying proper data/input, preprocessing, representations, AI techniques, and evaluation metrics/methodology [Apply]
Readings : [**Russell2020**], [**Bostrom2014**]	

Unit 11: Additional AI Applications and Societal Impact (6 hours)	
Competences Expected: AG-C06	
Topics	**Learning Outcomes**
• One or more additional applications of AI to a broad set of problems and diverse fields, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose a different area from that chosen for the CS Core): – Formulating and evaluating a specific application as an AI problem: ∗ How to deal with underspecified or ill-posed problems – Data availability/scarcity and cleanliness: ∗ Basic data cleaning and preprocessing ∗ Data set bias – Algorithmic bias – Evaluation bias – Assessment of societal implications of the application • Additional depth on deployed deep generative models: – Introduction to how deep image generative models work, (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion) including discussion of attention – Introduction to how large language models work, (e.g., as of 2023, ChatGPT, Bard) including discussion of attention – Idea of foundational models, how to use them, and the benefits/issues with training them from big data • Analysis and discussion of the societal impact of AI: – Ethics – Fairness – Trust/explainability – Privacy and usage of training data – Human autonomy and oversight/regulations/legal requirements – Sustainability	• Analyze the societal impact of one or more specific real-world AI applications, identifying issues regarding ethics, fairness, bias, trust, and explainability [Evaluate] • Describe some of the failure modes of current deep generative models for language or images, and how this could affect their use in an application [Describe]
Readings : [**Russell2020**], [**Bostrom2014**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

AI368. Evolutionary Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	AI368. Evolutionary Computing
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	AI264. Deep Learning. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Evolutionary Computing is a subfield of Artificial Intelligence inspired by natural selection and biological evolution. This course explores metaheuristic optimization algorithms such as Genetic Algorithms, Evolution Strategies, and Genetic Programming. These techniques are essential for solving complex optimization and search problems where traditional analytical methods are insufficient. Students will learn to design, implement, and analyze evolutionary systems applied to engineering, data science, and autonomous agents.

5. GOALS

- Understand the principles of natural selection applied to computation.
- Master the design of representations, operators, and fitness functions.
- Implement and tune Genetic Algorithms and Evolution Strategies.
- Analyze the convergence and performance of bio-inspired algorithms.
- Apply evolutionary techniques to multi-objective optimization problems.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Foundations of Evolutionary Computation (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Introduction to Evolutionary Algorithms (EAs) and biological metaphors. • The General Evolutionary Algorithm cycle: Initialization, Selection, Variation, Replacement. • Components: Populations, chromosomes, genes, alleles. • Fitness landscapes, genotype-phenotype mapping, and search spaces.	• Explain the relationship between biological evolution and artificial search processes [Familiarizarse] • Identify the components of a generic evolutionary algorithm and their roles [Usar] • Describe the concept of fitness landscapes and their impact on EA performance [Familiarizarse]
Readings : [**Eiben2015**], [**Back1996**]	

Unit 2: Genetic Algorithms and Variants (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Canonical Genetic Algorithms (CGA): binary representation, selection schemes (roulette, tournament). • Genetic Operators: Crossover (single-point, uniform, arithmetic), Mutation (bit-flip, Gaussian). • Schema Theorem and Building Block Hypothesis. • Advanced GA variants: Real-coded GAs, Genetic Programming (GP) with tree representations.	• Design appropriate representations and operators for specific optimization problems [Evaluar] • Implement a canonical Genetic Algorithm for function optimization [Usar] • Apply Genetic Programming to symbolic regression or automatic programming tasks [Evaluar]
Readings : [**Eiben2015**], [**Mitchell1998**]	

Unit 3: Evolution Strategies and Evolutionary Programming (18 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Evolution Strategies (ES): (μ, λ) and $(\mu+\lambda)$ selection mechanisms. • Self-adaptation of strategy parameters (mutation step sizes). • Covariance Matrix Adaptation (CMA-ES). • Evolutionary Programming (EP) and its focus on behavioral evolution.	• Implement Evolution Strategies with self-adaptive mutation for continuous optimization [Evaluar] • Compare the performance of GAs vs ES on benchmark problems [Usar] • Explain the role of strategy parameter adaptation in convergence speed [Familiarizarse]
Readings : [**Eiben2015**], [**Deb2001**]	

Unit 4: Multi-objective and Hybrid Evolutionary Algorithms (18 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Multi-objective Optimization: Pareto optimality, dominance relations. • Multi-objective EAs: NSGA-II, SPEA2, MOEA/D. • Diversity preservation mechanisms and archiving strategies. • Memetic Algorithms: Hybridization with local search. • Constraint handling techniques in evolutionary search.	• Solve multi-objective problems using Pareto-based selection methods [Evaluar] • Design a memetic algorithm combining global and local search [Evaluar] • Evaluate the trade-offs between convergence and diversity in MOEAs [Usar]
Readings : [**Eiben2015**], [**Deb2001**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY