NATIONAL UNIVERSITY OF SAN MARCOS

FACULTY OF MATHEMATICAL SCIENCES

SYLLABUS

1. GENERAL INFORMATION

1.1 Course name	:	Introduction to Computer Science
1.2 Code	:	CS100
1.3 Academic Program	:	Scientific Computing
1.4 Credits	:	3
1.5 Prerrequisites	:	None
1.6 Type of Course	:	Mandatory
1.7 Area of Studies	:	Estudios Generales
1.8 Number of Weeks	:	16
1.9 Hours per week	:	2 HT; 2 HP;
1.10 Cycle	:	I
1.11 Learning modality	:	Face to face
1.12 Academic Period	:	2026-I
1.13 Semester Start Date	:	
1.14 Semester End Date	:	

2. PROFESSORS

Meetings after coordination with the professor

3. 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.

4. 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.

5. COMPETENCES

ABET-1) An ability to identify, formulate, and solve broadly defined technical or scientific problems by applying knowledge of mathematics and science and/or technical topics to areas relevant to the discipline. (Familiarity)

ABET-5) An ability to understand ethical and professional responsibilities and the impact of technical and/or scientific solutions in global, economic, environmental, and societal contexts. (Familiarity)

ABET-2) An ability to formulate or design a system, process, procedure, or program for the intended purpose. (Familiarity)

6. TOPICS

Unit 1: Computational Thinking (Part I) (4 hours)	
Competences Expected: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5

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: ABET-1,ABET-2,ABET-5

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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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: ABET-1,ABET-2,ABET-5	
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**]	

7. WORKPLAN

7.1 Methodology

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

7.2 Theory Sessions

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

7.3 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.

8. EVALUATION SYSTEM

********* EVALUATION MISSING ********

9. BASIC BIBLIOGRAPHY