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

1. COURSE

CS3E1. Blockchain and Decentralized Systems (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS3E1. Blockchain and Decentralized Systems
2.2 Cycle	:	CS300 Cycle.
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	:	• CS3P1. Parallel and Distributed Computing . (8^{th} Sem) • CS292. Software Engineering II. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Blockchain and decentralized systems have moved from a niche cryptocurrency topic to a general-purpose computing paradigm used in finance, supply chains, digital identity, and public infrastructure. This elective gives students a rigorous, systems-level understanding of how a blockchain actually works: the data structures and cryptographic primitives (hashing, digital signatures, Merkle trees) that make a distributed ledger tamper-evident, the peer-to-peer networking and consensus protocols (PoW, PoS, BFT-family) that let mutually distrusting nodes agree on a single history without a central authority, and the execution environment (the EVM) that turns a blockchain into a general-purpose computing platform for smart contracts.

Building on these foundations, students develop, test, and audit smart contracts using Solidity and industry-standard tooling, and architect decentralized applications that combine an on-chain backend with a Web3 frontend and wallet-based identity. Because smart contracts manage real economic value and cannot be easily patched after deployment, the course places particular emphasis on the vulnerability classes unique to this domain – reentrancy, integer overflow, access-control flaws, and oracle manipulation – and on the auditing discipline needed to catch them before deployment.

The course closes by situating these technical skills in their broader context: the scalability and interoperability techniques (Layer-2, bridges) and economic primitives (tokens, automated market makers, lending protocols) that underpin decentralized finance, and the governance, regulatory, and ethical questions – on-chain voting, DAO governance failures, and the evolving legal treatment of cryptoassets – that any engineer working in this space must be prepared to reason about.

5. GOALS

- Explain the data-structure, cryptographic, and peer-to-peer networking foundations that make a blockchain a tamper-evident, decentralized ledger.
- Analyze and compare consensus mechanisms, including Proof-of-Work, Proof-of-Stake, and BFT-family protocols, in terms of security, performance, and trust assumptions.
- Design, program, and test smart contracts on an EVM-compatible platform using Solidity and modern development and testing tooling.
- Architect decentralized applications that integrate a Web3 frontend, a wallet, and an on-chain smart-contract backend.
- Identify common smart-contract security vulnerabilities and apply auditing techniques and tools to mitigate them.

- Evaluate blockchain scalability and interoperability approaches, the economic primitives behind decentralized finance (DeFi), and the governance and regulatory implications of decentralized systems.

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. (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: Distributed Ledgers and Peer-to-Peer Architectures (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Distributed ledger structure: a chain of blocks linked by cryptographic hashes, providing append-only, tamper-evident storage of ordered transactions Cryptographic Algorithms. • Peer-to-peer network topology and gossip/broadcast protocols for propagating blocks and transactions without a central server Networked Applications. • Node roles: full nodes, light nodes, and miners/validators, and the responsibilities each assumes in maintaining the ledger. • Decentralization versus centralization trade-offs: trust assumptions, single points of failure, censorship resistance, and performance costs Evaluation.	• Explain how a distributed ledger's block structure and hash linkage provide tamper-evidence [Explain] • Compare peer-to-peer and client-server architectures in terms of fault tolerance, scalability, and trust [Compare] • Analyze the decentralization trade-offs of a given blockchain design against a centralized alternative [Analyze]
Readings : [Nar+16], [Ant23]	

Unit 2: Cryptographic Foundations for Blockchain (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Cryptographic hash functions: properties (preimage resistance, collision resistance, avalanche effect) and their use in chaining blocks Cryptographic Algorithms. • Digital signatures for transaction authentication: key pairs, signing and verification, and non-repudiation Cryptographic Algorithms. • Merkle trees: construction, Merkle roots, and Merkle proofs for efficient verification of transaction inclusion in a block. • Public-key-derived addresses and key management considerations Cryptography Fundamentals.	• Explain which cryptographic hash-function properties make a blockchain's chain of blocks tamper-evident [Explain] • Implement a Merkle tree and verify an inclusion proof for a given transaction [Implement] • Apply digital signatures to authenticate and verify a blockchain transaction [Apply]
Readings : [Nar+16], [Ant23]	

Unit 3: Memory and Consistency (8 hours)	
Competences Expected: AG-C09,AG-C12	
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 : [Nar+16], [Bas23]	

Unit 4: Blockchain Execution Platforms: The EVM Model (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• EVM architecture: stack-based virtual machine, opcodes, and the bytecode execution model Common Aspects: Platforms, APIs and Constraints. • Gas model: execution cost accounting, gas limits, and its role in bounding computation and preventing denial-of-service. • Account and state model: externally-owned accounts versus contract accounts, and persistent contract storage. • Smart contract lifecycle: compilation to bytecode, deployment, invocation, and transaction receipts.	• Explain how the gas model bounds smart contract execution and mitigates denial-of-service risk [Explain] • Describe how EVM bytecode is executed against the stack and persistent contract storage [Describe] • Compare externally-owned accounts and contract accounts in terms of capabilities and state [Compare]
Readings : [AW18]	

Unit 5: Smart Contract Development and Testing (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• The Solidity programming language: contract structure, state variables, functions, modifiers, and events Common Aspects: Platform Languages and Web Programming Patterns. • Smart contract development frameworks and tooling (e.g., Hardhat, Foundry, Truffle) for compiling, deploying, and scripting contracts Development Environment and IDE. • Local and public test networks for simulating contract execution prior to mainnet deployment. • Unit and integration testing of smart contracts, including test coverage and mocking of external dependencies Advanced Testing Practices.	• Program a smart contract in Solidity that implements a specified business rule [Program] • Write unit and integration tests for a smart contract using a development framework [Write] • Determine whether a smart contract's test suite provides adequate coverage of its state transitions [Determine]
Readings : [AW18], [SKH19]	

Unit 6: Decentralized Application (DApp) Architecture and Web3 Integration (6 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	**Learning Outcomes**
• DApp architecture: on-chain smart-contract backend versus off-chain frontend and storage Frontend, Frameworks and Web Standards. • Wallets as identity and signing agents: key custody, transaction signing, and connection protocols. • Web3 client libraries (e.g., ethers.js, web3.js) for reading blockchain state and submitting signed transactions Frontend, Frameworks and Web Standards. • Oracles and off-chain data feeds for supplying external data to on-chain contracts.	• Design the on-chain/off-chain architecture of a decentralized application for a given use case [Design] • Integrate a web frontend with a user's wallet using a Web3 client library to sign and submit transactions [Integrate] • Explain the role of oracles in supplying trustworthy off-chain data to smart contracts [Explain]
Readings : [AW18], [SKH19]	

Unit 7: Smart Contract Security and Auditing (8 hours)	
Competences Expected: AG-C11,AG-C12	
Topics	**Learning Outcomes**
• Reentrancy attacks and checks-effects-interactions mitigation patterns Memory Safety and Types. • Integer overflow and underflow in contract arithmetic, and safe-math mitigations Memory Safety and Types. • Access-control flaws in contract functions, such as missing modifiers or unprotected initialization Access Control and Applications. • Oracle and price-manipulation attacks, and front-running/transaction-ordering risks. • Auditing methodology: manual code review, static analysis, and formal verification tools Threat Analysis and Security Engineering.	• Analyze a smart contract for reentrancy and integer-overflow vulnerabilities [Analyze] • Apply checks-effects-interactions and safe-math patterns to remediate an identified vulnerability [Apply] • Evaluate the findings of a static-analysis tool run against a smart contract and prioritize remediation [Evaluate]
Readings : [ABC17], [AW18]	

Unit 8: Blockchain Scalability, Interoperability, and DeFi Primitives (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Layer-2 scaling techniques: rollups, state channels, and sidechains Evaluation. • Cross-chain interoperability: bridges and their trust assumptions. • Tokenomics and decentralized finance (DeFi) primitives: fungible/non-fungible tokens, automated market makers, and lending protocols Economic Models and Strategies.	• Compare Layer-2 scaling approaches in terms of throughput, security, and trust assumptions [Compare] • Analyze the trust assumptions and failure modes of a given cross-chain bridge design [Analyze] • Describe how a given DeFi primitive, such as an automated market maker, uses token incentives to provide a decentralized financial service [Describe]
Readings : [Bas23], [Wer+22]	

Unit 9: Blockchain Governance, Regulation, and Ethical Case Studies (4 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• On-chain governance and decentralized autonomous organizations (DAOs): voting mechanisms and protocol upgrade processes. • Regulatory treatment of cryptoassets across jurisdictions and its implications for compliance Security Laws, Policies and Responsibility. • Ethical case studies: notable governance failures and contested community interventions Ethical Foundations and Frameworks.	• Explain how a DAO's on-chain voting mechanism governs protocol upgrades [Explain] • Debate the trade-offs between permissionless innovation and regulatory compliance in a given cryptoasset case [Debate] • Evaluate the ethical and governance implications of a historical blockchain governance failure [Evaluate]
Readings : [FW18], [Bas23]	

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

[Nar+16] Arvind Narayanan et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. 1st. Princeton University Press, 2016.

[ABC17] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. "A Survey of Attacks on Ethereum Smart Contracts (SoK)". In: *Principles of Security and Trust (POST)*. Springer, 2017. DOI: 10.1007/978-3-662-54455-6_8.

[AW18] Andreas M. Antonopoulos and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. 1st. O'Reilly Media, 2018.

[FW18] Primavera De Filippi and Aaron Wright. *Blockchain and the Law: The Rule of Code*. 1st. Harvard University Press, 2018.

[SKH19] Kevin Solorio, Randall Kanna, and David H. Hoover. *Hands-On Smart Contract Development with Solidity and Ethereum*. 1st. O'Reilly Media, 2019.

[Wer+22] Sam M. Werner et al. "SoK: Decentralized Finance (DeFi)". In: *Proceedings of the 4th ACM Conference on Advances in Financial Technologies (AFT)*. ACM, 2022. DOI: 10.1145/3558535.3559780.

[Ant23] Andreas M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. 3rd. O'Reilly Media, 2023.

[Bas23] Imran Bashir. *Mastering Blockchain: A Deep Dive into Distributed Ledgers, Consensus Protocols, Smart Contracts, DeFi, and More*. 4th. Packt Publishing, 2023.