

ANNA UNIVERSITY, CHENNAI
NON- AUTONOMOUS COLLEGES
AFFILIATED TO ANNA UNIVERSITY
M.E. SOFTWARE ENGINEERING
REGULATIONS 2025

PROGRAMME OUTCOMES (POs):

PO	Programme Outcomes
PO1	An ability to independently carry out research /investigation and development work to solve practical problems
PO2	An ability to write and present a substantial technical report/document.
PO3	Students should be able to demonstrate a degree of mastery over the area as per the specialization of the program. The mastery should be at a level higher than the requirements in the appropriate bachelor program

PROGRAMME SPECIFIC OUTCOMES:

PSO1: Apply advanced software engineering principles, tools, and methodologies to design, develop, and evaluate reliable, secure, and scalable software systems.

PSO2: Conduct research and apply innovative practices in software engineering to solve complex industrial and societal problems with sustainable solutions.



ANNA UNIVERSITY, CHENNAI

POSTGRADUATE CURRICULUM (NON-AUTONOMOUS AFFILIATED INSTITUTIONS)

Programme: M.E., Software Engineering

Regulations: 2025

Abbreviations:

BS – Basic Science (Mathematics)

L – Laboratory Course

ES – Engineering Science (Programme Core (**PC**),
Programme Elective (**PE**))

T – Theory

SD – Skill Development

LIT – Laboratory Integrated Theory

SL – Self Learning

PW – Project Work

TCP – Total Contact Period(s)

Semester I									
S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.	MA25C07	Advanced Mathematical Methods (CSIE)	T	3	1	0	4	4	BS
2.	CP25C01	Advanced Data Structures and Algorithms	LIT	3	0	4	7	5	ES (PC)
3.	CP25C02	Advanced Database Technologies	T	3	0	0	3	3	ES (PC)
4.	CP25C03	Advanced Operating Systems	T	3	0	0	3	3	ES (PC)
5.	CP25C04	Advanced Compiler Design	T	3	0	0	3	3	ES (PC)
6.	SE25101	Technical Seminar	-	0	0	2	2	1	SD
Total Credits							22	19	

Semester – II

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.	SE25201	Advanced Software Engineering and Tools	LIT	3	0	2	5	4	ES (PC)
2.	IF25C01	Advanced Software Testing techniques	LIT	3	0	2	5	4	ES (PC)
3.		Programme Elective I	T	3	0	0	3	3	ES (PE)
4.	CP25C07	Quantum Computing	T	2	0	0	2	2	ES (PC)
5.		Industry Oriented Course I	-	1	0	0	1	1	SD
6.	SE25202	Industrial Training	-	-	-	-	-	2	SD
7.		Self-Learning Course	-	-	-	-	-	1	-
Total Credits							16	17	

Semester – III

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.		Programme Elective II	T	3	0	0	3	3	ES (PE)
2.		Programme Elective III	T	3	0	0	3	3	ES (PE)
3.		Programme Elective IV	T	3	0	0	3	3	ES (PE)
4.		Programme Elective V	T	3	0	0	3	3	ES (PE)
5.		Industry Oriented Course II	-	1	0	0	1	1	SD
6.	SE25301	Project Work I	-	0	0	12	12	6	SD
Total Credits							25	19	

Semester – IV

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.	SE25401	Project Work II	-	0	0	24	24	12	SD
Total							24	12	

PROGRAMME ELECTIVE COURSES (PE)

S. No.	Course Code	Course Title	Periods per week			Total Contact Periods	Credits
			L	T	P		
1.	SE25001	AI-Driven Software Engineering	3	0	0	3	3
2.	SE25002	Blockchain for Software Engineering	3	0	0	3	3
3.	IF25C02	MLOps	3	0	0	3	3
4.	SE25003	5G and Next-Gen Network Software	3	0	0	3	3
5.	SE25004	Applied Machine Learning	3	0	0	3	3
6.	SE25005	Deep Learning for Software Engineering	3	0	0	3	3
7.	SE25006	AI for Software Testing and Quality Assurance	3	0	0	3	3
8.	SE25007	Data Visualization Techniques	3	0	0	3	3
9.	SE25008	Formal Models of Software Systems	3	0	0	3	3
10.	CP25C17	Edge and Fog Computing	3	0	0	3	3
11.	SE25009	Smart and Secure Software System	3	0	0	3	3
12.	SE25010	GPU Computing	3	0	0	3	3
13.	SE25011	Web Services and API Design	3	0	0	3	3
14.	SE25012	Decentralized Software Systems	3	0	0	3	3
15.	SE25013	Advanced Cloud Computing Technologies	3	0	0	3	3
16.	SE25014	Augmented Reality and Virtual Reality	3	0	0	3	3
17.	SE25015	Software Requirements Engineering	3	0	0	3	3
18.	CP25C19	Cognitive Computing	3	0	0	3	3
19.	SE25016	Large Language Models	3	0	0	3	3
20.	SE25017	Software Systems for IOT	3	0	0	3	3

MA25C07	Advanced Mathematical Methods (CSIE)	L	T	P	C
		3	1	0	4
<p>Course Objectives:</p> <ul style="list-style-type: none"> • Develop an in-depth understanding of advanced concepts in linear algebra, multivariate analysis, and number theory for computer science applications. • Apply mathematical tools such as eigenvalue decomposition, SVD, and multivariate statistical methods to real-world computing and data-driven problems. • Analyze and implement number-theoretic techniques for cryptography, security, and algorithmic problem-solving in computer science. 					
<p>Linear Algebra: Vector spaces, norms, Inner Products, Eigenvalues using QR transformations, QR factorization, generalized eigenvectors, Canonical forms, singular value decomposition and applications, pseudo inverse, least square approximations.</p>					
<p>Multivariate Analysis: Random vectors and matrices, Mean vectors and covariance matrices, Multivariate normal density and its properties, Principal components, Population principal components, Principal components from standardized variables.</p>					
<p>Elementary Number Theory: The division algorithm, Divisibility and the Euclidean algorithm, The fundamental theorem of arithmetic, Modular arithmetic and basic properties of congruences; Principles of mathematical induction and well ordering principle. Primality Testing algorithms, Chinese Remainder Theorem, Quadratic Congruence.</p>					
<p>Advanced Number Theory: Advanced Number Theory, Primality Testing algorithms, Chinese Remainder Theorem, Quadratic Congruence, Discrete Logarithm, Factorization Methods, Side Channel Attacks, Shannon Theory, Perfect Secrecy, Semantic Security.</p>					
<p>Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%.</p>					
<p>Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).</p>					
<p>References:</p> <ol style="list-style-type: none"> 1. Gilbert Strang, Linear Algebra and Its Applications, <i>Cengage Learning</i>. 2. Richard A. Johnson & Dean W. Wichern, Applied Multivariate Statistical Analysis, Pearson. 3. Neal Koblitz, A Course in Number Theory and Cryptography, Springer. 4. Victor Shoup, A Computational Introduction to Number Theory and Algebra, Cambridge University Press. 					

E-resources:

1. <https://ocw.mit.edu/courses/18-06-linear-algebra>
2. <https://nptel.ac.in/courses/111105041>
3. <https://crypto.stanford.edu/pbc/notes/numbertheory>

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, mathematical tools, and techniques used in advanced computational and information engineering problems.	--	--
CO2	Analyze complex mathematical models and methods to solve problems in computational systems and information engineering applications.	PO1 (3)	PSO1 (3)
CO3	Evaluate advanced mathematical approaches and solutions to determine their effectiveness and applicability to engineering and computational problems.	PO3 (2)	PSO2 (2)
CO4	Design innovative solutions for computational and information engineering problems by applying appropriate mathematical methods and modeling techniques.	PO2 (1)	PSO1 (3)

CP25C01	Advanced Data Structures and Algorithms	L	T	P	C
		3	0	4	5
<p>Course Objectives:</p> <ol style="list-style-type: none"> 1. To explore advanced linear, tree, and graph data structures and their applications. 2. To design efficient algorithms using appropriate algorithmic paradigms. 3. To evaluate computational complexity and identify tractable vs. intractable problems. 					
<p>Linear Data Structures and Memory Optimization: Advanced arrays: Sparse arrays, dynamic arrays, cache-aware structures, Linked lists: Skip lists, unrolled linked lists, XOR linked lists, Stacks and Queues: Priority queues, double-ended queues, circular buffers, Hashing: Perfect hashing, cuckoo hashing, extendible hashing.</p> <p>Practical:</p> <ul style="list-style-type: none"> • Implement skip lists and measure performance compared with balanced BST. • Experiment with cache-aware data structures and analyze memory utilization. 					
<p>Advanced Tree Data Structures: Balanced Trees: AVL, Red-Black Trees, Splay Trees, Treaps, Multi-way Trees: B-Trees, B+ Trees, R-Trees, Segment Trees, Fenwick Trees, Suffix Trees and Tries for string processing, Applications in indexing, text retrieval, computational geometry.</p> <p>Practical:</p> <ul style="list-style-type: none"> • Implement B+ tree for database indexing use-case. • Design a suffix tree-based algorithm for DNA sequence matching. 					
<p>Graph Data Structures and Algorithms: Representation: Adjacency list/matrix, incidence matrix, compressed storage, Traversals: DFS, BFS with applications, Shortest Path Algorithms: Dijkstra, Bellman-Ford, Floyd-Warshall, Johnson's algorithm, Minimum Spanning Trees: Prim's, Kruskal's, Borůvka's algorithm, Network Flow Algorithms: Ford-Fulkerson, Edmonds-Karp, Push-Relabel.</p> <p>Practical:</p> <ul style="list-style-type: none"> • Implement Johnson's algorithm for sparse graph shortest paths. • Demonstration of Maximum flow in traffic or network routing simulation. 					
<p>Algorithm Design and Paradigms: Divide and Conquer: Karatsuba's multiplication, Strassen's algorithm, Greedy Methods: Huffman coding, interval scheduling, set cover approximation, Dynamic Programming: Matrix chain multiplication, Floyd-Warshall, knapsack variants, Backtracking and Branch-and-Bound, Randomized Algorithms and Probabilistic Analysis.</p> <p>Practical:</p> <ul style="list-style-type: none"> • Implement Strassen's algorithm and compare with naive matrix multiplication. • Develop a randomized algorithm for primality testing (Miller–Rabin). 					

Computational Complexity and Approximation Algorithms: Complexity Classes: P, NP, NP-Complete, NP-Hard, Reductions: Polynomial-time reductions, Cook-Levin theorem (overview), Approximation Algorithms: Vertex cover, set cover, TSP, k-center problem, Heuristic Algorithms: Local search, simulated annealing, genetic algorithms.

Practical:

- Implement approximation algorithm for vertex cover.
- Complexity analysis of a chosen NP-hard problem and implement a heuristic.

Advanced Topics and Emerging Trends: Randomized Algorithms – Monte Carlo Algorithms, Parallel and Distributed Algorithms – PRAM Model, Divide and Conquer in Parallel, Load Balancing, Streaming Algorithms – Data Stream Models, Sketching and Sampling, Frequency Moments, Advanced String Matching – Suffix Trees, Suffix Arrays, Pattern Matching in Linear Time.

Practical:

- Implement randomized and streaming algorithms on real-world datasets.
- Design of parallel and distributed algorithms.

Weightage: Continuous Assessment: 50%, End Semester Examinations: 50%

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20)

References:

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT Press.
2. La Rocca, M. (2021). Advanced algorithms and data structures. Manning Publications.
3. Goodrich, M. T., Tamassia, R., & Mount, D. M. (2011). Data structures and algorithms in C++. John Wiley & Sons, Inc.
4. Weiss, M. A. (2014). Data structures and algorithm analysis in C++. Pearson Education.
5. Drozdek, A. (2013). Data structures and algorithms in C++. Cengage Publications.

E-resources:

1. <https://www.theiotacademy.co/blog/data-structures-and-algorithms-in-c/>
2. https://github.com/afrid18/Data_structures_and_algorithms_in_cpp
3. <https://www.udemy.com/course/introduction-to-algorithms-and-data-structures-in-c/?srsltid=AfmBOorEZlkgV7QzaEh6lqzAaKLjC-lpFU1NGgWfoHMLhOos-uDVKjCK>

	Description of CO	PO	PSO
CO1	Describe data structures and implement algorithmic solutions for complex computational problems.	--	--
CO2	Analyze the time complexity and efficiency of algorithms for various computing problems.	PO1(3)	PSO1(3)
CO3	Evaluate algorithmic techniques and datastructures to determine theirsuitability for different applications.	PO3(2)	PSO2(2)
CO4	Design optimized solutions for real-world problems using appropriate algorithms and data structures.	PO2(1)	PSO1(3)

CP25C02	Advanced Database Technologies	L	T	P	C
		3	0	0	3
<p>Course Objectives:</p> <ul style="list-style-type: none"> • To strengthen the understanding of enhanced ER models and their transformation into relational models with indexing and file structures. • To understand object-oriented and object-relational database concepts and querying using OQL. • To explore techniques in query processing, execution, and optimization strategies. 					
<p>Entity Relationship Model: Entity Relationship Model Revised-Subclasses, Superclasses and Inheritance -Specialization and Generalization-Union Types-Aggregation.</p> <p>Activity: Design ER Model for a specific use case.</p>					
<p>Enhanced Entity Relational Model: Relational Model Revised, Converting ER and EER Model to Relational Model-SQL and Advanced Features, File Structures, Hashing, and Indexing.</p> <p>Activity: Demonstration of SQL Implementation.</p>					
<p>Object Relational Databases: Object Database Concepts-Object Database Extensions to SQL, The ODMG Object Model and ODL, Object Database Conceptual Design-Object Query Language OQL-Language Binding in the ODMG Standard.</p> <p>Activity: Demonstration of Object Query Language.</p>					
<p>Query Processing and Optimization: Query Processing, Query Trees and Heuristics, Query Execution Plans, Cost Based Optimization.</p> <p>Activity: Design of Query Evaluation Plans.</p>					
<p>Distributed Databases: Real-Time Bidding, E-mail Marketing, Affiliate Marketing, Social Marketing Mobile Marketing, Distributed Database Concepts, Data Fragmentation, Replication and Allocation, Distributed Database Design Techniques, Distributed Database Design Techniques, Distributed Database Architectures.</p> <p>Activity: Demonstration of Concurrency and Transactions.</p>					
<p>NOSQL Systems and Bigdata: Introduction to NOSQL Systems-The CAP Theorem, Document, based NOSQL Systems, Key-value Stores, Column-Based or Wide Column NOSQL Systems, NOSQL Graph Databases and Neo4j.</p> <p>Activity: Design application with MongoDB.</p>					

Advanced Database Models, Systems and Applications: Active Database Concepts and Triggers, Temporal Database Concepts, Spatial Database Concepts, Multimedia Database Concepts, Deductive Database Concepts, Introduction to Information Retrieval and Web Search.

Activity: Demonstration of Hadoop infrastructure for Big Data Analytics.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).

References:

1. Elmasri, R., & Navathe, S. B. (2016). Fundamentals of database systems. Pearson Education.
2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database system concepts, McGraw Hill Education.
3. Ceri, S., & Pelagatti, G. Distributed databases: Principles and systems. McGraw Hill.
4. Ramakrishnan, R., & Gehrke, J. (2004). Database management systems. McGraw Hill.

E-resources:

1. <https://www.edx.org/learn/sql/stanford-university-databases-advanced-topics-in-sql>
2. <https://www.coursera.org/courses?query=sql&productDifficultyLevel=Advanced>

	Description of CO	PO	PSO
CO1	Elaborate different database models for effective database design.	--	--
CO2	Implement advanced database features for optimized data retrieval.	PO1(3)	PSO1(3)
CO3	Evaluate query processing and optimization strategies to improve system performance.	PO3(2)	PSO2(2)
CO4	Design solutions using advanced database models to address complex data-intensive applications.	PO2(1)	PSO1(3)

CP25C03	Advanced Operating Systems	L	T	P	C
		3	0	0	3
<p>Course Objectives:</p> <ul style="list-style-type: none"> • To analyze the architectures and design issues of advanced operating systems. • To develop the model for process synchronization and recovery in complex environments. • To evaluate algorithms for distributed coordination, resource management, fault tolerance, and security. 					
<p>Advanced Process and Thread Management: Multithreading models, thread pools, context switching, Synchronization issues and solutions: semaphores, monitors, lock-free data structures, CPU scheduling in multi-core systems</p> <p>Activity: CPU scheduler simulation for multicore systems.</p>					
<p>Memory and Resource Management in Modern OS: Virtual memory, demand paging, page replacement policies-Huge pages, NUMA-aware memory management-Resource allocation in cloud-native environments</p> <p>Activity: Simulate demand paging and page replacement algorithms.</p>					
<p>Virtualization and Containerization: Hypervisors (Type I & II), KVM, QEMU, Xen-Containers: Docker, LXC, systemd-nspawn-OS-level virtualization and namespaces</p> <p>Activity: Deploy and configure Docker containers with various images.</p>					
<p>Distributed Operating Systems and File Systems: Distributed scheduling, communication, and synchronization-Distributed file systems: NFS, GFS, HDFS-Transparency issues and fault tolerance</p> <p>Activity: Simulate distributed process synchronization.</p>					
<p>Security and Trust in Operating Systems: Access control models: DAC, MAC, RBAC-OS hardening techniques, sandboxing, SELinux, AppArmor-Secure boot, rootkit detection, trusted execution environments</p> <p>Activity: Implement Role-Based Access Control (RBAC) using Linux user and group permissions.</p>					
<p>Real-Time and Embedded Operating Systems: Real-time scheduling algorithms (EDF, RM)-POSIX RT extensions, RTOS architecture-TinyOS, FreeRTOS case studies</p> <p>Activity: Analyze FreeRTOS task scheduling behavior.</p>					
<p>Edge and Cloud OS: Future Paradigms: Serverless OS, unikernels, lightweight OS for edge computing-Mobile OS internals (Android, iOS)-OS for quantum and neuromorphic computing (intro)</p> <p>Activity: Analyze Android's system architecture using emulator tools.</p>					
<p>Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%</p>					

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).

References:

1. Tanenbaum, A. S., & Bos, H. (2023). Modern operating systems. Pearson.
2. Buyya, R., et al. (2022). Content delivery networks and emerging operating systems. Springer.
3. Silberschatz, A., Galvin, P. B., & Gagne, G. (2022). Operating system concepts. Wiley.
4. Anderson, T., & Dahlin, M. (2021). Operating systems: Principles and practice. Recursive Books.
5. Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2020). Operating systems: Three easy pieces.

E-Resources:

1. Prof. Smruti Ranjan Sarangi, "Advanced Distributed Systems", IIT Delhi, NPTEL, https://onlinecourses.nptel.ac.in/noc22_cs80/preview
2. Prof. Rajiv Misra, "Cloud Computing and Distributed Systems", IIT Patna, NPTEL, <https://nptel.ac.in/courses/106104182>

	Description of CO	PO	PSO
CO1	Describe operating system concepts for memory and resource management.	--	--
CO2	Analyse virtualization and distributed OS mechanisms for scalability and performance.	PO1(3)	PSO1(3)
CO3	Evaluate OS security and resource handling strategies in diverse environments.	PO3(2)	PSO2(2)
CO4	Design innovative OS solutions using modern tools and techniques.	PO2(1)	PSO1(3)

CP25C04	Advanced Compiler Design	L	T	P	C
		3	0	0	3
<p>Course Objectives:</p> <ul style="list-style-type: none"> • To analyze the theory and principles of modern compiler design and advanced optimization techniques. • To design and implement efficient front-end and back-end compiler components for programming languages. • To evaluate code optimization strategies and runtime environment management in contemporary architectures. 					
<p>Intermediate Representations and Control Flow Analysis: Static single assignment (SSA) form- Context-Free Grammar (CFG) construction-dominance relations-Intermediate Representation (IR) design for functional and imperative languages-Static single assignment and def-use chains</p> <p>Activities:</p> <ol style="list-style-type: none"> 1. Convert source code to SSA form using LLVM IR. 2. Visualize control flow graphs from SSA using LLVM tools. 					
<p>Program Analysis and Transformations: Data flow analysis- live variable analysis-reaching definitions-Alias analysis and dependence analysis-Loop optimizations and transformations</p> <p>Activities:</p> <ol style="list-style-type: none"> 1. Perform loop unrolling and strength reduction. 2. Conduct live variable analysis and visualize data flow graphs. 					
<p>Advanced Optimizations and Polyhedral Compilation: Polyhedral model for loop nests-Tiling, skewing, fusion, and vectorization-Profile-guided and feedback-directed optimizations</p> <p>Activities:</p> <ol style="list-style-type: none"> 1. Implement loop tiling and loop skewing on a matrix multiplication program. 2. Analyze the effect on loop-intensive code with LLVM optimization flags. 					
<p>Just-in-Time (JIT) and Runtime Compilation: JIT compilation models: tracing, method-based-GraalVM architecture, Java HotSpot internals-LLVM JIT and dynamic language support</p> <p>Activities:</p> <ol style="list-style-type: none"> 1. Develop a basic JIT-enabled interpreter with LLVM or GraalVM. 2. Implement dynamic dispatch using LLVM JIT API. 					

Machine Learning in Compiler Design: ML for phase ordering, auto-tuning, and IR prediction-Reinforcement learning for optimization passes-Dataset creation and benchmarking for compiler ML

Activities:

1. Train an ML model to predict optimization passes.
2. Use reinforcement learning for pass selection in toy compiler.

Domain-Specific Languages (DSLs) and Compiler Extensions: Designing DSLs for AI/ML, DSP, graphics-Code generation for custom accelerators-Integration with TensorFlow XLA and Halide

Activities:

1. Design and test a simple DSL grammar using ANTLR.
2. Integrate a DSL with TensorFlow XLA or Halide.

Security, Verification, and Future Trends: Secure compilation and type-safe intermediate representations-Compiler fuzzing and formal verification (e.g., CompCert)-Quantum compilers, multi-target compilers, and neuromorphic systems

Activities:

1. Use CompCert to verify compilation of simple programs.
2. Apply compiler fuzzing using tools like libFuzzer.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).

References:

1. Cooper, K. D., & Torczon, L. (2023). Engineering a compiler. Morgan Kaufmann.
2. Grune, D., Bal, H. E., Jacobs, C. J. H., & Langendoen, K. G. (2012). Modern compiler design (2nd ed.). Springer.
3. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). Compilers: Principles, techniques, and tools (2nd ed.). Pearson.
4. Völter, M. (2013). DSL engineering: Designing, implementing and using domain-specific languages. dslbook.org.
5. Sarda, S., & Pandey, M. (2015). LLVM essentials. Packt Publishing.

E-Resources:

1. Prof. AmeyKarkare, IIT Kanpur, "Advanced Compiler Optimizations"
Link:<https://www.cse.iitk.ac.in/users/karkare/Courses/cs738/>
2. Prof. Santanu Chattopadhyay, "Compiler Design", IIT Kharagpur
Link:"https://onlinecourses.nptel.ac.in/noc21_cs07/preview"

	Description of CO	PO	PSO
CO1	Explain intermediate control flow techniques in compiler design.	--	--
CO2	Apply program analysis techniques and advanced optimizations for design of compilers.	PO1(3)	PSO1(3)
CO3	Develop compiler features and machine learning techniques for optimization.	PO3(2)	PSO2(2)
CO4	Evaluate secure compilation strategies for quantum and multi-target compilation.	PO2(1)	PSO1(3)

SE25201	Advanced Software Engineering and Tools	L	T	P	C
		3	0	2	4
<p>Course Objective: To equip students with the principles and tools of advanced software engineering, including sustainable practices at scale, microservices architecture, and DevOps. Emphasis is placed on maintainability, observability, CI/CD, and automation for scalable systems.</p>					
<p>Culture and Sustainability in Software Engineering</p> <p>Working Effectively in Teams-Knowledge Sharing Practices-Engineering for Equity-Measuring Engineering Productivity-Style Guides and Coding Standards-</p> <p>Code Review Practices-Effective Documentation-Long-Term Programming Considerations.</p> <p>Practical:</p> <ol style="list-style-type: none"> 1. Define Project and Problem Statement- MS Word, Google Docs 2. Analyze Users and Choose Process Model- Draw.io, Lucidchart 3. Write Software Requirements (SRS)- MS Word, Markdown Editors <p>Building and Designing Microservices</p> <p>Introduction to Microservices-The Role of the Evolutionary Architect-Service Modeling Techniques-Integration Styles and Patterns-Workflow Management in Microservices-Decomposing the Monolith-REST-Based Communication-gRPC-Based Communication-Designing for System Failure.</p> <p>Practical:</p> <ol style="list-style-type: none"> 1. Create Project Plan and Estimate Effort- GanttProject, Trello, Excel 2. Make Work Breakdown and Risk Plan- WBS Creator, Risk Register Templates 3. Design System Architecture and Diagrams- Draw.io, Lucidchart, StarUML <p>Testing, Observability, and Resilience</p> <p>Testing Strategies and Approaches-Consumer-Driven Contract Testing-Monitoring and Observability Techniques-Logging and Distributed Tracing-Structured Logging Practices-Building Resilient Systems-Introduction to Chaos Engineering-Defining Metrics That Matter-Overview of Testing Practices</p>					

Practical

1. Draw DFD and Class Diagram- StarUML, Visual Paradigm
2. Create Interaction, Activity, and State Diagrams- UML Tools (e.g., StarUML, Creately)
3. Draw State, Sequence, and Deployment Diagrams- Visual Paradigm, PlantUML

Software at Scale and Tooling

Managing Code Complexity-Dependency Management Strategies-Handling Large-Scale Code Changes-Version Control and Branching Strategies-Build Systems and Build Philosophies-Continuous Integration Techniques-Tools for Code Review-Coordinating Software Launches-Automation and Reliability Engineering

Practical:

1. Design Frontend (UI/UX)- Figma, Adobe XD, Wireframe.cc
2. Write Sample Code for a Microservice- VS Code, Spring Boot, Flask
3. Create Test Plan and Test Cases- TestLink, Excel, Zephyr

Deployment and Microservice Operations

Microservices Deployment Strategies-Exploring Deployment Options-Introduction to Service Meshes-Configuration Management in Microservices-Securing Microservice Architectures-Testing in Production Environments-Data Management in Microservices-Observability and Diagnostics in Operations-Evaluating the Use of Microservices

Practical

1. Perform Manual Testing- Postman, JMeter, Browser DevTools
2. Write User Manual and Cost Analysis- MS Word, Excel
3. Final Demo and Report Submission- GitHub, PowerPoint, Google Slides

Course Outcomes:

1. Understand and apply sustainable software engineering practices used at large-scale organizations.
2. Design, build, and deploy microservice-based software systems.
3. Implement robust testing strategies and observability tools.
4. Utilize modern tooling for CI/CD, version control, and configuration management.
5. Analyze, monitor, and secure distributed applications in production environments.

Assessment Weightage:

Weightage:	Continuous Assessment: 60%	End Semester Theory Examinations: 40%
	(i) Activity: 15% (ii) Internal Theory Examination: 35% (iii) Internal Laboratory Examinations: 15%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References:

1. Titus Winters, Tom Manshreck, Hyrum Wright, Software Engineering at Google, O'Reilly Media, 2020.
2. Ian Sommerville, Software Engineering, 10th Edition, Pearson, 2015.
3. Steve McConnell, Code Complete, 2nd Edition, Microsoft Press, 2004.
4. Martin Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 2018.
5. Jez Humble and David Farley, Continuous Delivery, Addison-Wesley, 2010.

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the advanced principles, tools, and methodologies used in software engineering to develop high-quality software systems.	--	--
CO2	Analyze software engineering tools, design patterns, and methodologies to understand their application in complex system development and maintenance.	PO1 (3)	PSO1 (3)
CO3	Evaluate software engineering frameworks, tool chains, and quality assurance techniques to assess their effectiveness in large-scale software projects.	PO3 (2)	PSO2 (2)
CO4	Design software systems by selecting appropriate tools, techniques, and methodologies for specific development challenges in real-world applications.	PO2 (1)	PSO1 (3)

IF25C01	Advanced Software Testing Techniques	L	T	P	C
		3	0	2	4

Course Objective:

To provide students with a deep understanding of modern and advanced software testing techniques, including exploratory testing, test heuristics, risk analysis, and real-world defect detection strategies. This course emphasizes hands-on problem-solving and designing robust testing strategies.

Foundations and Testing Thinking

Role of the Software Tester-Common Testing Fallacies-Challenges in Software Testing-Characteristics of Good Tests-Developing a Tester’s Mindset-Heuristics in Software Testing-Risk Based Testing Approaches-Importance of Negative Testing-Software Modeling for Testing

Activities: Understand the Role of a Tester and Identify Risks, Design Good Tests using Test Heuristics

Advanced Test Design Techniques

Exploratory and Scripted Testing-Equivalence Partitioning and Boundary Value Analysis-State Transition Testing Techniques-Decision Tables and Pairwise Testing-Stress and Load Testing Methods-Approaches to Security Testing-Regression Test Planning-Designing Test Automation-Effective Bug Reporting Practices

Activities: Perform Boundary Value, Equivalence, and Pairwise Testing

Exploratory Testing in Practice

Overview of Exploratory Testing-Charter-Based Testing Sessions-Test Tour Techniques-Investigative Testing with Models-Observation and Note-Taking Techniques-Use of Oracles and Heuristics-Session-Based Test Management (SBTM)-Debriefing and Reporting in Exploratory Testing-Designing Focused Test Missions

Activities: Conduct Exploratory Testing with Session Notes, Practice Bug Reporting and Use Oracles, Perform Load and Stress Testing

Automation and Strategy Integration

Guidelines for Test Automation-Automation in Exploratory Testing-Test Framework Selection and Design-Continuous Testing in DevOps Environments-Testing in Agile and CI/CD-Workflows-Behavior-Driven Development (BDD) Techniques-Analyzing Test Coverage and Identifying Gaps-Test Metrics and Dashboarding Tools-Managing Technical Debt in Testing

Activities: Design and Automate Regression Tests, Use BDD for Agile Testing

Domain-Specific and Specialized Testing

Mobile Application Testing-Testing Web and Cloud-Based Applications-Usability and Accessibility Testing-Testing for Embedded and IoT Systems-Game Testing Methodologies-Testing AI/ML-Based Software-Ethical Considerations in Software Testing-Conducting Test Strategy Reviews and Audits-Emerging Trends in Software Testing

Activities: Perform Mobile and Web App Testing, Review Test Strategy and Metrics Dashboard

Course Outcomes:

1. Analyze and apply various advanced testing strategies to real-world problems.
2. Design and execute effective exploratory testing sessions.
3. Apply test heuristics and risk-based strategies in complex systems.
4. Integrate testing techniques into CI/CD and Agile environments.
5. Automate testing using appropriate tools and frameworks effectively.

Weightage: Continuous Assessment: 60%, End Semester Theory Examination: 40%

(i) Activity: 15% (ii) Internal Theory Examination: 35% (iii) Internal Laboratory Examinations: 15%

Assessment Methodology: Assignments (30), Quiz (10), Virtual demonstration (25), Flipped Classroom (10), Review of GATE & IES questions (25).

References:

1. Cem Kaner, James Bach, Bret Pettichord, Lessons Learned in Software Testing, Wiley, 2002.
2. Elisabeth Hendrickson, Explore It! Reduce Risk and Increase Confidence with Exploratory Testing, Pragmatic Bookshelf, 2013.
3. Rex Black, Advanced Software Testing – Vol 2, Rocky Nook, 2009.
4. Alan Richardson, Dear Evil Tester, Self-published, 2016.
5. Lisa Crispin & Janet Gregory, Agile Testing Condensed, Agile Testing Fellowship, 2019.

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, methodologies, and advanced practices of software testing used to ensure high-quality software systems.	--	--
CO2	Analyze advanced software testing techniques and test strategies to identify defects, assess risks, and improve software reliability.	PO1 (3)	PSO1 (3)
CO3	Evaluate software testing tools, automation frameworks, and testing models to determine their effectiveness in complex software environments.	PO3 (2)	PSO2 (2)
CO4	Design comprehensive and efficient software testing solutions by selecting appropriate techniques, tools, and automation strategies for real-world applications.	PO2 (1)	PSO1 (3)

CP25C07	Quantum Computing	L	T	P	C
		2	0	0	2
Course Objective:					
<ol style="list-style-type: none"> 1. To provide a mathematical foundation for Quantum Computing and provide the basics of working 2. To interpret the various aspects and applications of quantum computing. 3. To examine the factors that affect Quantum computation 					
Physical properties and mathematical foundations					
Double Slit Experiment; Light: Particle Vs Wave; Heisenberg Uncertainty Principle. Vector spaces – basis; Inner product; Outer product; Tensor product; Linear operators					
Activities:					
<ul style="list-style-type: none"> • Simulate the experiment using an interactive virtual lab • Construct simple operators and visualize action on vectors 					
Quantum computing postulates and gates					
Review of postulates, Bloch sphere, Single qubit states and gates, superposition; Two Qubit States and Gates - Bell States, Entanglement, CNOT gate, Phase oracles, Pauli Gates.					
Activities:					
<ul style="list-style-type: none"> • Group quiz to match postulates to physical implications • Visualization with Bloch sphere simulators 					
Quantum computing circuits					
Dirac's notation for quantum computing, Computational Basis, Orthonormality, Hadamard and Phase Gates- building quantum circuits					
Activities:					
<ul style="list-style-type: none"> • Use IBM Q Composer to build and simulate custom circuits • Hands-on: Apply X, H, Z gates and observe results on simulators 					
Fundamental Quantum Algorithms					
Deutsch–Josza Algorithm, Grover search algorithm: Problem definition, Amplitude amplification, Grover oracle, diffuser, multiple solutions in the search space					
Activities:					
<ul style="list-style-type: none"> • Construct DJ circuit for a 3-bit input function • Simulation of Grover's algorithm with multiple marked elements 					
Programming on a real quantum computer					
Coding a real time quantum computer via IBMQ to carry out basic quantum measurement and state analysis.					

Activities:

- Connect Qiskit with IBMQ using personal API token
- Hands-on: Create 1- and 2-qubit circuits using Hadamard, X, Z, and measurement gates
- Compare real and simulated results and Observe impact of quantum noise

Text Books:

1. Chuck Easttom, "Quantum Computing Fundamentals", 1st edition, Published by Addison-Wesley Professional (June 1st 2021)
2. Qiskit TextBook - <https://qiskit.org/textbook/preface.html> (2022)

References:

1. Kasirajan, Venkateswaran. Fundamentals of quantum computing. Springer International Publishing, 2021.
2. Chris Bernhardt, Quantum Computing for Everyone, The MIT Press, Cambridge, 2020
3. Nielsen, Michael A., and Isaac L. Chuang, "Quantum Computation and Quantum Information" Cambridge University Press (5 April 2013)

Weightage: Continuous Assessment:40%, End Semester Theory Examination: 60%

Assessment Methodology: Assignments (30), Quiz (10), Virtual demonstration (25), Flipped Classroom (10), Review of GATE & IES questions (25).

CO	Description of CO	PO	PSO
CO1	Describe the fundamental principles, postulates, and computational models of quantum computing and their significance in next-generation computing systems.	--	--
CO2	Analyze quantum algorithms and quantum circuit models to understand their computational advantages, limitations, and performance characteristics.	PO1 (3)	PSO1 (3)
CO3	Evaluate quantum computing paradigms, error correction techniques, and hardware technologies for their suitability in solving complex computational problems.	PO3 (2)	PSO2 (2)
CO4	Design quantum circuits and algorithmic solutions by selecting appropriate quantum gates, qubit architectures, and computational models for real-world problem scenarios.	PO2 (1)	PSO1 (3)

SE25001

AI-Driven Software Engineering

L	T	P	C
3	0	0	3

Course Objectives:

1. Understand the concepts of Software Engineering and integration of AI techniques in software engineering processes
2. Learn how AI can enhance requirement engineering, design, coding, testing, and maintenance.
3. Analyse and apply machine learning and NLP techniques in software development.
4. To explore various techniques for test case generation and explore real-world AI-SE tools

Foundations of Software Engineering

Introduction to Software Engineering -SDLC models (Waterfall, Agile, DevOps)-Key SE activities- Requirements Engineering-Design-Implementation-Testing-Maintenance-Principles of software quality-reliability and metrics- Overview of SDLC phases and challenges-AI vs Traditional Software Engineering

Foundations of Artificial Intelligence for Software Engineering

Basics of AI and ML-supervised, unsupervised, reinforcement learning- Introduction to NLP (text preprocessing, classification, embeddings)-AI tools for developers: Scikit-learn, SpaCy, Transformers, OpenAI APIs

AI for Requirements Engineering

Use case modeling, user stories, and scenario-based modeling -Requirement classification using supervised ML (SVM,Random Forest,BERT)- Requirement Extraction using Named Entity Recognition, Text Chunking- Clustering and sentiment analysis for prioritization - Traceability using Semantic Similarity.

AI for Software Design and Code Generation

Role of design in AI-driven development – design patterns and intelligent code structuring-AI-assisted software architecture modeling – decision support using ML-Code synthesis from natural language prompts – Prompt Engineering, CodeT5, Codex-Code refactoring and optimization using AI – static code analysis tools (e.g., SonarQube with AI plugins)-Software quality evaluation using AI-based code review tools-Knowledge-based and learning-based design evaluation-Tools and platforms: GitHub Copilot, ChatGPT Code Interpreter, OpenAI Codex

AI for Software Testing and Intelligent Maintenance

AI in Test Case Generation – Classification, Clustering, Genetic Algorithms-Test suite optimization using Reinforcement Learning-Fault prediction models using ML – classification algorithms for defect detection-AI in Regression Testing – Change Impact Analysis using NLP-Predictive maintenance of software systems – anomaly detection, performance monitoring-LLMs and Automated Bug Fixing – using models for automatic

patch generation-Case studies: AI-enabled testing frameworks (e.g., Testim, Mabl),
 Research trends-Industry tools: Diffblue, EvoSuite, DeepCode

Course Outcomes:

After successful completion of the course, students will be able to:

1. Explain key software engineering concepts and how AI can transform each phase
2. Apply NLP and ML techniques to extract, classify, and prioritize software requirements.
3. Generate and optimize code/design using AI techniques.
4. Use AI to improve test automation and debugging efficiency

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References

1. Ian Sommerville, Software Engineering, 10th Ed., Pearson
2. Hui Liu and Tianpei Xia, Artificial Intelligence Techniques for Software Engineering, Springer, 2021

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental principles and applications of artificial intelligence techniques in software engineering for automating and optimizing development processes.	--	--
CO2	Analyze AI algorithms and models used in software engineering to understand their application in requirements analysis, design, testing, and maintenance.	PO1 (3)	PSO1 (3)
CO3	Evaluate AI-driven tools and methodologies to assess their effectiveness in improving software development efficiency and software quality.	PO3 (2)	PSO2 (2)
CO4	Design AI-based software engineering solutions by selecting suitable machine learning models, automation frameworks, and optimization strategies for real-world software projects.	PO2 (1)	PSO1 (3)

SE25002	Blockchain for Software Engineering	L	T	P	C
		3	0	0	3
<p>Course Objective: To enable students to understand, design and implement blockchain technologies in the context of software engineering, covering architecture, platforms, security, lifecycle integration, and cutting-edge research applications.</p>					
<p>Introduction to Blockchain Technology: History and evolution of blockchain - Architecture: Blocks, hashes, Merkle trees - Types: Public, Private, Consortium - Cryptographic foundations: Hashing, Digital Signatures</p> <p>Activity: Flipped Classroom & Concept Mapping Students will create concept maps to visually compare blockchain types and components.</p> <p>Blockchain Protocols and Platforms: Consensus mechanisms: PoW, PoS, PBFT - Bitcoin, Ethereum, Hyperledger, and Corda - Ethereum Virtual Machine (EVM) - Cross-chain interoperability and APIs .</p> <p>Activity: Case Study & Platform Comparison Analyze and compare Ethereum and Hyperledger implementations.</p> <p>Smart Contracts and dApps: Solidity programming and contract lifecycle - Smart contract vulnerabilities and testing - Token standards: ERC-20, ERC-721 - Case study: dApp for software licensing.</p> <p>Activity: Project-Based Learning Students will develop and test a simple smart contract using Truffle Suite.</p> <p>Blockchain in Software Lifecycle: Blockchain-based requirement engineering - Decentralized CI/CD pipelines - Blockchain for quality assurance - Bug tracking and versioning.</p> <p>Activity: Group Simulation Teams will simulate integration of blockchain in DevOps lifecycle.</p> <p>Trust, Security and Privacy in Blockchain: Decentralized identity and auditability - GDPR and compliance in blockchain - zk-SNARKs and privacy-preserving tools - Secure code provenance.</p> <p>Activity: Research Paper Discussion Students will analyze a blockchain privacy protocol paper and simulate secure access control.</p> <p>Blockchain Integration and Interoperability: Legacy system integration with blockchain - Use of oracles and off-chain data - DevSecOps and blockchain synergy - Interoperable smart contracts.</p>					

Activity: Tool Demonstration

Students will demonstrate blockchain API use with existing DevOps tools.

Advanced Trends and Applications: DAOs and blockchain governance in SE - Blockchain for agile & software licensing - Blockchain with AI/ML in SE - Research trends and regulatory aspects.

Activity: Expert Talk + Whitepaper Review

Students review research whitepapers and attend a guest lecture on blockchain-AI integration.

E-Resources:

NPTTEL: 'Blockchain Architecture Design and Use Cases' by Dr. S. Rajagopalan, IIT Kharagpur

Coursera: 'Blockchain Basics' by University at Buffalo (SUNY)

Book: 'Mastering Blockchain' by Imran Bashir, Packt, 2023

Book: 'Blockchain Technology and Applications' by S. Shukla et al., Springer, 2021

Tutorial: IBM Blockchain Developer Center

YouTube: Ethereum & Solidity Programming Playlists

Textbooks

1. Mastering Blockchain – Imran Bashir, 4th Edition, Packt, 2023
2. Blockchain Technology and Applications – S. Shukla et al., Springer, 2021
3. The Science of the Blockchain – Roger Wattenhofer, 2021
4. Blockchain for Distributed Systems Security – Xueming Si, Patrick Lee, Springer, 2022
5. Blockchain and Artificial Intelligence – Kiran S. et al., CRC Press, 2021

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks:		
Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, components, and role of blockchain technology in modern software engineering practices.	--	--
CO2	Analyze blockchain-based software architectures, development frameworks, and integration strategies to understand security, reliability, and scalability aspects.	PO1 (3)	PSO1 (3)
CO3	Evaluate blockchain platforms, smart contract technologies, and software engineering methodologies to assess their effectiveness for building decentralized applications.	PO3 (2)	PSO2 (2)
CO4	Design blockchain-enabled software solutions by selecting appropriate architectures, development tools, and smart contract models for real-world software systems.	PO2 (1)	PSO1 (3)

IF25C02	MLOps	L	T	P	C
		3	0	0	3

Course Objectives:

- This course equips students with the necessary tools and expertise to become proficient MLOps engineers, addressing the industry's shortage of skilled professionals capable of deploying and managing machine learning models efficiently.
- Train & Deploy ML models, Monitor, Retrain & Improve, and Scale AI applications.

MLOps Fundamentals: Introduction to MLOps-purpose and benefits of MLOps, role in accelerating the ML development process, impact on model deployment and maintenance, MLOps Principles-continuous integration, continuous deployment, version control, and automation, ML Lifecycle- various stages of ML lifecycle, data collection, preprocessing, model training, validation, deployment, and monitoring, MLOps Architecture.

Activities:

- **Group Discussion/Presentation:** On benefits of MLOps vs traditional ML workflows.
- **Case Study Analysis:** Analyze a real-world MLOps implementation (e.g., Netflix, Google, or Uber).
- **Roleplay Exercise:** Assign roles (Data Engineer, ML Engineer, DevOps) and simulate an ML project kickoff meeting.
- **Mini Quiz:** Concepts of CI/CD, version control, automation in ML

Data Management and Preparation: Data Collection and Validation-gather, clean, and validate data for ML models, Feature Engineering and Selection-extracting meaningful features from data and selecting the most important ones for model training, Data Versioning-importance of managing data versions for reproducibility and auditing, Data Pipelines-Building robust data pipelines for automating data processing and feeding data to models.

Activities:

- **Data Cleaning Exercise:** Provide a raw dataset for students to clean and validate.
- **Feature Engineering Challenge:** Students extract and select relevant features from a dataset.
- **Data Versioning Task:** Use tools like DVC to version a dataset and demonstrate reproducibility.

Model Building and Experimentation: Model Selection and Training-Choosing the right ML algorithm and training models on the prepared data, Experiment Tracking-using tools like MLflow to track ML experiments and their results, Model Resource Management-Optimizing model resource usage and deployment strategies, Model

Evaluation and Tuning-Evaluating model performance and tuning hyper parameters for optimal results.

Activities:

- **Model Comparison Exercise:** Train 2–3 different ML models on the same dataset and compare results.
- **Leaderboard Competition:** Mini Kaggle-style competition where students track and tune their models.
- **Model Evaluation Drill:** Perform confusion matrix, precision, recall, ROC-AUC, etc., on a classification model.

Deployment and Monitoring: Model Serving-models serving patterns and infrastructure, Containerization with Docker- Using Docker to package and deploy ML models, CI/CD Pipelines: Automating ML model training and deployment using CI/CD pipelines, Model Monitoring: Implementing monitoring systems to track model performance and identify potential issues.

Cloud Deployment: Deploying ML models on cloud platforms like Google Cloud, Amazon AWS, or Azure.

Activities:

- **Cloud Lab:** Deploy a model to Google Cloud, AWS, or Azure (free tier).
- **Monitoring Simulation:** Set up basic logging and alerting for a deployed model.
- **Scaling Exercise:** Simulate traffic to a deployed model and scale it using Kubernetes (or simulated with Docker Compose).

Tools and Technologies: MLflow-MLflow for experiment tracking, model management, and deployment, TensorFlow/PyTorch-popular ML frameworks, Kubernetes-Kubernetes for deploying and managing ML applications, Cloud Platform Tools-tools offered by cloud providers for MLOps, Databricks- Exploring Databricks for data science and MLOps, Vertex AI: Using Vertex AI for MLOps on Google Cloud, Practical application and projects

Activities:

- **Tool Comparison Matrix:** Compare features and use cases of MLflow, Vertex AI, Databricks, TensorFlow, PyTorch, and Kubernetes.
- **Framework Implementation Task:** Build and train a model using both TensorFlow and PyTorch.
- **End-to-End Mini Project:** Students build, train, deploy, and monitor a small model using the full MLOps stack.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examination: 60%
	(i). Activities: 10% (ii). Internal Theory Examinations: 30%	

Mandated Activities with marks:

Assignments (30), Quiz (10), Virtual demonstration (25), Flipped Classroom (10), Review of GATE & IES questions (25).

Internal Examinations: TWO tests

References:

1. Yaron Haviv, Noah Gift, Implementing MLOps in the Enterprise 2023
2. Emmanuel Raj, Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale 2021.
3. Noah Gift, Alfredo Deza, Practical MLOps 2021.
4. Stephen Fleming, Accelerated DevOps with AI, ML & RPA: Non-Programmer's Guide to AIOPS & MLOPS 2020.
5. Mark Treveil, Nicolas Omont, Clément Stenac, Kenji Lefevre, Du Phan, Joachim Zentici, Adrien Lavoillotte, Makoto Miyazaki, Lynn Heidmann, Introducing MLOps, 2020.

E-resources:

1. MLOps Principles –MLflow, Blog <https://mlflow.org/docs/latest/index.html>
2. Machine Learning and Deep Learning Fundamentals and Applications (IIT Guwahati – Prof. M. K. Bhuyan)

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, workflows, and lifecycle management practices of MLOps for deploying and maintaining machine learning systems.	--	--
CO2	Analyze MLOps pipelines and automation practices to understand model deployment, monitoring, versioning, and performance management.	PO1 (3)	PSO1 (3)
CO3	Evaluate MLOps tools, platforms, and operational strategies to assess their effectiveness in managing scalable and reliable machine learning solutions.	PO3 (2)	PSO2 (2)
CO4	Design end-to-end MLOps solutions by selecting appropriate tools, architectures, and best practices for real-world machine learning deployments.	PO2 (1)	PSO1 (3)

SE25003	5G and Next-Gen Network Software	L	T	P	C
		3	0	0	3
<p>Course Objective: This course introduces the architecture, protocols, and software components underpinning 5G networks and future-generation programmable networks. It focuses on core technologies like 5G NR, SDN, NFV, MEC, and network slicing, enabling learners to understand the convergence of communication and software-defined systems.</p>					
<p>Evolution to 5G and Use Cases: Cellular generations: 1G to 5G, Key drivers for 5G: latency, bandwidth, density, 5G use cases: eMBB, URLLC, mMTC, ITU-T 5G architecture vision, and 5G spectrum Seminar: 5G use cases.</p> <p>5G System Architecture: 5G NR (New Radio) and gNB components, NG-RAN and 5G Core (5GC) overview, 5G protocol stack: RRC, PDCP, RLC, MAC, PHY, 5G deployment options: NSA vs. SA. Poster: 5G vs. LTE architecture</p> <p>Software Defined Networking (SDN):SDN principles and architecture, Separation of control and data plane, OpenFlow protocol and flow tables, SDN controllers: ONOS, OpenDaylight</p> <p>Simulation: Mininet</p> <p>Network Function Virtualization (NFV): NFV architecture and components, Virtual Network Functions (VNFs) and their lifecycle, ETSI MANO framework, NFV benefits, and deployment models.</p> <p>Paper Review of ETSI NFV whitepaper</p> <p>Multi-access Edge Computing (MEC): Concept and need for edge computing, MEC architecture and interfaces, Use cases: AR/VR, autonomous vehicles, smart factories, Integration with 5G and NFV</p> <p>Case study on smart healthcare</p> <p>Network Slicing and Orchestration: Introduction to network slicing, Slice lifecycle management, Intent-based networking and orchestration tools, Slice isolation, QoS, and SLA enforcement</p> <p>Project virtual slice creation</p>					

Security and Challenges in Next-Gen Networks: 5G security architecture and threats, Authentication, integrity, and privacy mechanisms, SDN/NFV vulnerabilities, Challenges in large-scale deployment and interoperability

Quiz: 5G threat models

Course Outcomes:

1. Explain the need and key technologies driving 5G and next-gen networks.
2. Illustrate the 5G system architecture and protocol stack components.
3. Apply principles of SDN and NFV to programmable network design.
4. Describe MEC, network slicing, and orchestration mechanisms.
5. Analyze security considerations and deployment challenges in 5G networks.

Assessment Weightage:

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	

Mandated Activities with Marks:

Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)

Internal Examinations: Two Tests

References

1. Martin Sauter, From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband, John Wiley & Sons Ltd, 2017.
<https://onlinelibrary.wiley.com/doi/book/10.1002/9781119346913>
2. Afif Osseiran, Jose F. Monserrat, Patrick Marsch, Mischa Dohler, 5G Mobile and Wireless Communications Technology, Cambridge Press, 2016.
3. Peterson, Cascone, O'Connor, Vachuska, and Davie, Software-Defined Networks: A Systems Approach, Systems Approach LLC (Publisher), 2022.
4. Rajendra Chayapathi, Syed Farrukh Hassan, Paresh Shah, Network Functions Virtualization (NFV) with a Touch of SDN, Addison-Wesley Professional, 2016.

Suggested Tools and Platforms

1. Mininet / ONOS / OpenDaylight – for SDN simulation
GNS3 / NS-3 – for 5G protocol simulation
2. Docker / Kubernetes – for network function deployment
MATLAB / Python – for modeling and analysis

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, architectures, and software components of 5G and next-generation communication networks.	--	--
CO2	Analyze 5G network software architectures, protocols, and virtualization techniques to understand performance, latency, and scalability requirements.	PO1 (3)	PSO1 (3)
CO3	Evaluate 5G technologies, network slicing approaches, and software-defined networking solutions to assess their effectiveness for diverse communication services.	PO3 (2)	PSO2 (2)
CO4	Design 5G and next-generation network software solutions by selecting appropriate architectures, protocols, and management frameworks for real-world deployment scenarios.	PO2 (1)	PSO1 (3)

SE25004	Applied Machine Learning	L	T	P	C
		3	0	0	3

Course Objectives

- To understand the basic concepts of machine learning
- To learn supervised learning and its applications in Software Systems
- To know the unsupervised learning methods and their uses in Software Systems
- To learn Machine Learning tools to apply machine learning algorithms to Software
- Engineering Models

Feature Engineering: Overview of Machine Learning-Feature Selection-Overview of Feature Extraction-Principal Component Analysis-Linear Discriminant Analysis-Independent Component Analysis-Partial Least Squares Probability

Supervised Learning: Classification Models for Software Systems: Classification Process-Maximum likelihood estimator-Bayesian Learning-Decision Trees-K-Nearest Neighbour-Support Vector Machines-Logistic Regression-Performance Measures for Classification

Supervised Learning: Prediction Models for Software Systems: Need for Prediction Models-Regression-Linear Regression-Multiple Linear Regression-Polynomial Regression-Performance Measures for Prediction

Neural Networks for Software Systems: Artificial Neural Network-Perceptron-Multi-Layer Perceptron-Backpropagation-Application of Neural Networks in Software Engineering

Unsupervised Learning for Software Systems: Overview of Unsupervised Learning-Similarity Metrics for Types of Data-Clustering Methods-K-means-Hierarchical Clustering-DBSCAN

Ensemble Learning: Need of Ensemble Learning-Bagging-Boosting-Adaboost-Application of Ensemble Learning in Software Engineering

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References

1. Ethem Alpaydin, Introduction to Machine Learning, 4th edition, MIT Press 2020.

2. Tom M. Mitchell, Machine Learning, McGraw-Hill Education (India) Private Limited, 2013.
3. Bishop, Christopher M., Pattern Recognition and Machine Learning. Springer-Verlag, 2006.
4. Haykin S., "Neural Networks and Learning Machines", Prentice Hall, ISBN: 9780131471399.
5. Hastie, T., R. Tibshirani, J. Friedman, "The Elements of Statistical Learning", Springer, ISBN: 978-0387848587.
6. Shai Shalev-Shwartz and Shai Ben-David. "Understanding Machine Learning: From Theory to Algorithms", Cambridge University Press, ISBN:9781107057135

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, workflows, and practical considerations of applying machine learning techniques to real-world problems.	--	--
CO2	Analyse real-world datasets and machine learning models to understand data pre-processing, model performance, and result interpretation.	PO1 (3)	PSO1 (3)
CO3	Evaluate machine learning algorithms and application strategies to assess their effectiveness for solving domain-specific problems.	PO3 (2)	PSO2 (2)
CO4	Design applied machine learning solutions by selecting appropriate algorithms, features, and evaluation methods for practical deployment scenarios.	PO2 (1)	PSO1 (3)

Course Objectives:

1. To understand the theoretical foundations, algorithms and methodologies of Neural Network
2. To design and develop an application using specific deep learning models
3. To provide the practical knowledge in handling and analysing real world applications.
4. To understand how deep learning transforms key phases in the software development lifecycle.
5. To equip students with industry-ready skills in integrating DL into SE workflows through project-based learning.

Machine Learning Basics: Learning algorithms, Maximum likelihood estimation, Building machine learning algorithm, Neural Networks Multilayer Perceptron, Back-propagation algorithm and its variants Stochastic gradient decent, Curse of Dimensionality

Deep Learning Architectures: Machine Learning and Deep Learning, Representation Learning, Width and Depth of Neural Networks, Activation Functions: RELU, LRELU, ERELU, Unsupervised Training of Neural Networks, Restricted Boltzmann Machines, Auto Encoders, Deep Learning Applications

Convolutional Neural Networks: Architectural Overview, Motivation, Layers, Filters, Parameter sharing, Regularization, Popular CNN Architectures: ResNet, AlexNet - Applications

Transfer Learning: Transfer learning Techniques, Variants of CNN: DenseNet, PixelNet.

Sequence Modelling: Recurrent Neural Networks, Bidirectional RNNs, Encoder-decoder sequence to sequence architectures - BPTT for training RNN, Long Short Term Memory Networks.

DL for Requirements Engineering: RE tasks: elicitation, generation, analysis, classification, traceability; Models: BERT, Siamese networks, LSTM; Use cases, datasets, challenges

Code Representation and Generation: Code as data; Tokenization and ASTs; DL for code generation and completion; Pretrained models (CodeT5, GPT-2, CodeBERT); Benchmarks (APPS, CONCODE, CodeXGLUE) Code summarization; Code search with IR and DL; Graph-based models (GGNN, GraphCodeBERT); Contrastive learning for cross-modal representations Learning-based code refactoring, program repair; Dual learning between summarization and generation; Pre-trained vs fine-tuned models

Course Outcomes:

Upon completion of the course, the students will be able to

1. Recognize the characteristics of deep learning models that are useful to solve real-world problems.
2. Understand different methodologies to create application using deep nets.
3. Identify and apply appropriate deep learning algorithms for analyzing the data for variety of problems.
4. Explain key deep learning architectures and their applicability in software engineering.
5. Analyze and evaluate deep learning solutions across various software engineering subdomains.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References:

1. Ian Goodfellow, YoshuaBengio and Aaron Courville, " Deep Learning", MIT Press, 2017.
2. Josh Patterson, Adam Gibson "Deep Learning: A Practitioner's Chen X.P. et al. (2025). *Deep learning-based software engineering: progress, challenges, and opportunities*. Science China Information Sciences. [Open Access]
3. Zhang, L. et al., *Machine Learning for Software Engineers*, Springer, 2021.
4. Vaswani et al., *Attention is All You Need*, NeurIPS, 2017.
5. Goodfellow, Bengio & Courville, *Deep Learning*, MIT Press, 2016.
6. Allamanis, M. et al., *A Survey of Machine Learning for Big Code and Naturalness*, ACM Computing Surveys, 2018.
7. Feng, Z. et al., *CodeBERT: A Pre-Trained Model for Programming and Natural Languages*, EMNLP, 2020.

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, architectures, and applications of deep learning techniques in software engineering contexts.	--	--
CO2	Analyze software engineering problems using deep learning models to understand patterns in code, defects, and software behavior.	PO1 (3)	PSO1 (3)

CO	Description of CO	PO	PSO
CO3	Evaluate deep learning approaches and tools applied to software engineering tasks to assess their effectiveness and limitations.	PO3 (2)	PSO2 (2)
CO4	Design deep learning-based software engineering solutions by selecting appropriate models, data representations, and training strategies for real-world applications.	PO2 (1)	PSO1 (3)

SE25006	AI for Software Testing and Quality Assurance	L	T	P	C
		3	0	0	3

Course Objectives

- Understand the fundamentals of software testing and the role of AI in enhancing software quality assurance.
- Apply machine learning techniques for test case generation and prioritization.
- Analyze and implement AI-driven defect prediction and fault detection techniques.
- Utilize test automation frameworks and metrics for evaluating testing effectiveness.
- Leverage AI tools to improve usability, accessibility, and quality assurance decisions.

AI in Software Testing: Overview of Software Testing and QA- Testing levels: Unit, Integration, System, Acceptance- Limitations of manual and automated testing- Introduction to AI and Machine Learning- Role of AI in the SDLC-Types of ML: Supervised, Unsupervised, Reinforcement Learning

Test Case Generation Techniques: Black Box Testing: Equivalence Class Partitioning, Boundary Value Analysis (BVA)-Cause-effect graphing and decision tables-State transition testing, Use case and scenario-based test design-Combinatorial testing (pairwise, orthogonal array)-White Box Testing: Statement and branch coverage-Path testing, control flow graphs, Data flow testing-Loop testing and cyclomatic complexity-Condition and decision coverage

Test Case Design and Automation: Role and need for test case generation- Characteristics of good test cases-Fault models and their relationship to test design- Categories: Manual, Semi-automated, Automated-Overview of test oracles and assertions-Test case prioritization using ML (e.g., classification models)- Test automation frameworks: Selenium, Appium

Fault Detection and Defect Prediction: Definitions: Fault, Error, Defect, Failure (IEEE and ISO standards)-Types of Faults-Defect life cycle and bug tracking systems (Bugzilla, JIRA)-Static code analysis for early fault detection- Mutation testing, fault seeding, Fault Injection-Code-based metrics: LOC, Cyclomatic Complexity, Halstead Metrics

AI-Driven Quality Assurance: Test case selection and minimization-Redundancy detection and mutation testing-Code and model coverage metrics (statement, branch, path, transition)-Evaluating test effectiveness (fault detection rate, mutation score)- Software quality models (ISO 9126, SQuaRE)- Predictive analytics for QA decision-making- AI for usability and accessibility testing

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References

1. Srinivasan Desikan & Gopaldaswamy Ramesh. Software Testing: Principles and Practices. Pearson Education.
2. Ethem Alpaydin. Introduction to Machine Learning. MIT Press.

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, techniques, and role of artificial intelligence in software testing and quality assurance processes.	--	--
CO2	Analyse software systems using AI-driven testing and quality assurance techniques to identify defects, risks, and quality issues.	PO1 (3)	PSO1 (3)
CO3	Evaluate AI-based testing tools, models, and quality assurance frameworks to assess their effectiveness in improving software reliability.	PO3 (2)	PSO2 (2)
CO4	Design AI-enabled software testing and quality assurance solutions by selecting appropriate algorithms, tools, and evaluation strategies for real-world projects.	PO2 (1)	PSO1 (3)

Course Objectives:

1. To introduce the fundamentals and principles of data visualization.
2. To expose students to various data representation techniques using modern tools.
3. To develop competency in interactive and exploratory data visualization.
4. To evaluate the effectiveness of visualization methods across domains.
5. To facilitate industry readiness through projects, seminars, and research-oriented activities.

Introduction to Data Visualization: Importance of data visualization, historical context, data-to-ink ratio, storytelling with data, human perception and cognition, ethical concerns.

Data Types, Design Principles and Visualization Grammar: Types of data (categorical, numerical, temporal, geospatial), marks and channels, Gestalt principles, color theory, Tufte's principles, Grammar of Graphics.

Charts, Graphs, and Mapping Techniques: Bar, line, pie, scatterplots, box plots, heatmaps, treemaps, network graphs, choropleths, and map overlays. When and how to use them effectively.

Tools and Programming for Data Visualization: Tableau, Microsoft Power BI, Python libraries (Matplotlib, Seaborn, Plotly), JavaScript & D3.js introduction. Hands-on with real datasets.

Interactive Visualization and Dashboards: User-driven exploration, filters, drilldowns, linking charts, dashboards in Tableau/Plotly/Dash, design for performance and UX.

Visual Analytics, Big Data, and Real-World Applications: Visualizing large datasets, streaming data visualization, case studies from domains like healthcare, finance, social media, and cybersecurity.

Research, Evaluation and Emerging Trends: Visualization in research papers, reproduction of visual analysis, critique of visualizations, current trends like AR/VR in visualization, narrative visualization.

Course Outcomes (COs):

Upon successful completion of this course, the students will be able to:

CO1 Understand the principles, purposes, and importance of data visualization.

CO2 Apply appropriate visualization techniques for various data types and structures.

CO3 Use modern tools and libraries (e.g., Tableau, Python, D3.js) to create effective visualizations.

- CO4 Design interactive dashboards and exploratory data visualizations for real-world datasets.
- CO5 Critically evaluate visualizations for their effectiveness, aesthetics, and integrity.
- CO6 Translate research insights or complex data into visual stories for decision-making.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References:

1. Colin Ware, *Information Visualization: Perception for Design*, Morgan Kaufmann, 4th Edition, 2020.
2. Alberto Cairo, *The Truthful Art: Data, Charts, and Maps for Communication*, New Riders, 2016.
3. Tamara Munzner, *Visualization Analysis and Design*, CRC Press, 2014.
4. Nathan Yau, *Data Points: Visualization That Means Something*, Wiley, 2013.
5. Jeffrey Heer, *Interactive Data Visualization for the Web*, O'Reilly Media (D3.js-based).
6. Claus O. Wilke, *Fundamentals of Data Visualization*, O'Reilly, 2019 (also available online).
7. Tutorials and documentation from:
 1. <https://www.tableau.com/learn>
 2. <https://seaborn.pydata.org/>
 3. <https://plotly.com/python/>
 4. <https://d3js.org/>

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, principles, and tools used for effective data visualization and visual analytics.	--	--
CO2	Analyse datasets and visualization techniques to understand data patterns, trends, and insights through appropriate visual representations.	PO1 (3)	PSO1 (3)
CO3	Evaluate data visualization methods, tools, and dashboards to assess their effectiveness in communicating insights and supporting decision-making.	PO3 (2)	PSO2 (2)

CO	Description of CO	PO	PSO
CO4	Design effective data visualizations and dashboards by selecting suitable visual encodings, tools, and interaction techniques for real-world data analysis tasks.	PO2 (1)	PSO1 (3)

SE25008	Formal Models of Software Systems	L 3	T 0	P 0	C 3
---------	--	--------	--------	--------	--------

Course Objective:

This course aims to introduce students to the foundational principles and mathematical techniques used to model, specify, verify, and reason about software systems formally. Students will explore various formal methods such as automata theory, process algebras, temporal logic, and model checking. The course aims to equip students with the ability to design reliable, secure, and verifiable software by applying formal models to analyse system behaviour and ensure correctness in both sequential and concurrent systems.

Fundamentals of Software Specification: Role of Specification in Software Development-Sources of Software Complexity: Size, Structure, Environment, Domain, Communication-Techniques to Control Complexity-Specification Activities in the Software Lifecycle-Integrating Formal Methods into SDLC-Specification Qualities and Process Quality Attributes-Models of Process Quality, Product Quality, and Utility-Conformance to Stated Goals

Suggested Activities:

- Analyse real-world software failure cases for missing specification qualities
- Evaluate multiple requirement versions using a formal checklist

Abstractions and Formal Methods: Fundamental Abstractions in Computing and Software Construction-Formalism Fundamentals and the Formalization Process-Components of a Formal System: Syntax, Semantics, Inference Mechanism-Properties of Formal Systems: Consistency, Soundness-Automata Theory: DFA, NFA, FST, Extended FSM-State Machine Modelling and Case Study: Elevator Control System-Classification of Formal Methods: Property-Oriented Specification

Suggested Activities: Analyse abstraction levels in a software system (e.g., ATM or food delivery)

Logic and Specification Reasoning: Propositional Logic and Truth Tables-Reasoning with Assumptions and Natural Deduction-Predicate Logic: Syntax and Semantics-Knowledge Representation and Policy Language Specification.

Suggested Activities: Construct truth tables and apply natural deduction to logical propositions

Specification Models and Systems: Mathematical Abstractions for Model-Based Specification-Set Theory, Relations, and Functions in Formal Specification-Algebraic Specification: Properties and Structured Specifications-Case Study: Multi-Window GUI Environment-Calculus of Communicating Systems (CCS)-Operational Semantics, Derivation Trees, and LTS

Suggested Activities: Develop formal specifications for window states, events, and transitions using sets, functions, or algebraic models

Formal Specification Languages: Z Notation: Abstractions, Types, Relations, Functions, Sequences, Bags-Operational Abstraction in Z: Schema, Schema Decorators, Generic Functions-Property Verification and Consistency in Z Specifications-Object-Z Language: Structure, Classes, Composition, Parallelism-The B-Method and Abstract Machine Notation (AMN)-Structures, Arrays, Statements in B Specifications

Suggested Activities:Case study-based modelling of an Automated Billing System using Z/Object-Z

References:

1. “Vulnerability Detection: From Formal Verification to Large Language Models and Hybrid Approaches: A Comprehensive Overview”
2. **Authors:** Norbert Tihanyi, Tamas Bisztray, Mohamed Amine Ferrag, Bilel Cherif, Richard A. Dubniczky, Ridhi Jain, Lucas C. Cordeiro
Year: 2025 (Mar 13)**Link:**<https://arxiv.org/abs/2503.10784>
3. “**Lessons from Formally Verified Deployed Software Systems (Extended Version)**”
4. **Authors:** Li Huang, Sophie Ebersold, Alexander Kogtenkov, Bertrand Meyer, Yinling Liu **Year:** 2023 (Jan 5) **Link:**<https://arxiv.org/abs/2301.02206>

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

E- Sources

E- Books:

1. “Principles of Model Checking” - Christel Baier (Germany), Joost-Pieter Katoen (Netherlands) (MODULE 2)
2. “Formal Methods: State of the Art and New Directions” - Paul Boca (UK), Jonathan Bowen (UK), Jawed Siddiqi (UK) (MODULE 5)

Course Outcomes:

1. Analyze software-specification principles to manage complexity and assure quality.
2. Apply formal abstractions and automata-based models for system behaviour representation.

3. Evaluate logical arguments using propositional, predicate, and temporal logic for correctness proofs.
4. Design and Validate model-based and algebraic specifications for concurrent, interactive software systems.
5. Develop and Verify complete specifications in Z, Object-Z, and B-Method to ensure consistency and conformance to requirements.

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, mathematical foundations, and types of formal models used in the specification and analysis of software systems.	--	--
CO2	Analyse software system behaviours using formal modelling techniques to verify correctness, consistency, and reliability.	PO1 (3)	PSO1 (3)
CO3	Evaluate formal models, verification methods, and analysis tools to assess their effectiveness in validating complex software systems.	PO3 (2)	PSO2 (2)
CO4	Design formal specifications and models for software systems by selecting appropriate formalisms and verification techniques for real-world applications.	PO2 (1)	PSO1 (3)

CP25C17	Edge and Fog Computing	L	T	P	C
		3	0	0	3
Course Objective:					
<p>This course aims to provide students with a comprehensive understanding of designing computing systems that process data closer to where it's created, rather than sending everything to distant cloud servers. Students will learn to build smart applications that work across different computing layers - from powerful cloud servers to small edge devices like sensors and smartphones. By the end, students can create practical solutions for smart cities, healthcare, and industry that respond quickly and work efficiently.</p>					
Cloud to Edge Computing Paradigms: Evolution from Cloud to Fog and Edge Computing - IoT Architecture Requirements and Latency Limitations - Fog vs Edge					

vs Cloud: Differences in Computation and Storage - Multi-access Edge Computing (MEC) Concepts - Multi-tier Architectural Models for Distributed Systems

Suggested Activities:

- Case Study: IoT solution using fog or edge
- Group Discussion: Business opportunities in edge markets

Edge Offloading and System Optimization: Edge Offloading Strategies and Placement Issues - Challenges in Latency, Energy Efficiency, QoS, and Privacy - Queuing Theory and System Performance Metrics - Optimization Techniques using Machine Learning for Slice Orchestration

Suggested Activities:

- Performance Comparison: Cloud vs Edge
- Mathematical Modeling: Latency or offloading simulation

Resource Federation and Network Slicing: Cloud-to-Fog-to-Things (C2F2T) Integration - Network Slicing for 5G/6G: Slice Types and Lifecycle - Resource Virtualization: SDN, NFV, and Container-Based Federation - Modeling and Simulation for Resource Management

Suggested Activities:

- Implementation Case Study: 5G network slicing for verticals
- Quiz: Virtualization and resource slicing concepts

Middleware for Edge Systems: Middleware Design Goals: Heterogeneity, Discovery, Scalability – Service Orchestration at the Edge - Lightweight Containers: Docker and Kubernetes on Constrained Devices - Edge Cluster Formation and Node Coordination - Communication Protocols: MQTT, CoAP, and Microservices - Distributed Storage, Synchronization, and Fault Recovery

Suggested Activities:

- Prototype: Middleware architecture on edge devices
- Containerization Project: Docker/Kubernetes deployment
- Security Analysis: Vulnerabilities in edge middleware

Intelligence and Analytics at the Edge: Data Management at the Fog Layer: Caching, Placement, Lifecycle - Federated Learning and Distributed ML for Edge Analytics - Stream Processing and Real-Time Analytics Frameworks - Use Cases: Smart Cities, Healthcare, Smart Homes - Security Challenges: ML-Based Threat Detection, Behavioral Analysis - Privacy Techniques: Differential Privacy, Federated Anonymization

Suggested Activities:

- Implement Predictive Analytics: Apache Kafka + Flink
- Develop ML Model: IoT threat detection
- Case Study: Privacy-preserving fog computing

Advanced Technologies for Secure and Efficient Edge Computing: Blockchain for Fog Resource Security and Transparency - Software-defined Edge Networks and Intent-Based Networking - Green and Sustainable Edge Computing-Security and Privacy Challenges in Edge Environments

Suggested Activities:

- Case Study: Edge security or sustainability
- Technical Report: Designing a secure, green edge architecture

Future Directions and Emerging Trends: Edge AI and TinyML: Neural Compression, Quantization, Model Deployment - 6G Vision: Ultra-Low Latency, Terahertz Communications - Advanced Paradigms: Digital Twins, Serverless Edge, Cognitive Networks

Suggested Activities:

- Research Project: Emerging tech integration with edge computing
- Presentation: Future research directions and innovation trends
- Technical presentation on future research directions in edge computing

Weightage: Continuous Assessment: 40%, End Semester Theory Examinations: 60%

Assessment Methodology: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)

References:

1. Rajiv Pandey, Sunil Kumar Khatri, Neeraj Kumar Singh, Parul Verma, Artificial Intelligence and Machine Learning for EDGE Computing, 1st Edition,2022, Academic Press
2. S. Goundar, Ed., 'Edge Computing - Technology, Management and Integration'. IntechOpen,2023.
3. Ahmed Banafa, Secure And Smart Internet of Things (IoT) using Blockchain and Artificial Intelligence (AI).2018, River publications
4. Javid Taheri, Schahram Dustdar, Albert Zomaya, Shuiguang Deng, Edge Intelligence From Theory to Practice, 2023, Springer
5. Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications Deepak Gupta, Aditya Khamparia ,2021, Wiley-IEEE Press

E-Resources:

1. Kubernetes on Edge: K3s Documentation and Tutorials - <https://k3s.io/>
2. Foundation of Cloud IoT Edge ML, NPTEL, Prof. Rajiv Misra, IIT Patna - <https://archive.nptel.ac.in/courses/106/104/106104242/>
3. Open5GS Documentation - <https://open5gs.org/>

4. Eclipse IoT Working Group - <https://iot.eclipse.org/>
5. TensorFlow Lite for Microcontrollers - <https://www.tensorflow.org/lite/microcontrollers>
6. Apache Flink Training - <https://flink.apache.org/>
7. Hyperledger Fabric Tutorials - <https://hyperledger-fabric.readthedocs.io/>
8. Subedi, P., Alsadoon, A., Prasad, P.W.C. et al. Network slicing: a next generation 5G perspective. J Wireless Com Network 2021, 102 (2021). <https://doi.org/10.1186/s13638-021-01983-7>

Description of Course Outcomes and Mapping

CO	CO Description	PO	PSO
CO1	Explain the evolution of cloud-to-edge paradigms and describe the role of edge and fog computing in real-time IoT applications.	PO2, PO3	PSO1
CO2	Design and optimize network slicing strategies for integrated cloud–fog–edge systems using mathematical modeling and optimization techniques.	PO1, PO3	PSO1, PSO2
CO3	Develop middleware architectures and container-based solutions for distributed edge computing environments.	PO1, PO3	PSO1
CO4	Implement intelligent data management systems and machine learning solutions at the fog layer for real-time analytics.	PO1, PO3	PSO1, PSO2
CO5	Evaluate emerging technologies and research directions in edge and fog computing for future system design.	PO1, PO2, PO3	PSO2

SE25009	Smart and Secure Software System	L	T	P	C
		3	0	0	3

Course Objectives

1. To understand the integration of smart technologies (AI/ML, IoT) in software systems.
2. To explore secure software development practices from design to deployment.
3. To evaluate threats and vulnerabilities in intelligent systems.
4. To build awareness of privacy, ethics, and compliance in smart software ecosystems.

Introduction to Smart and Secure Software Systems: Overview of smart systems: features and examples-Characteristics of secure software-Security problems in software-Risk Management framework-Secure software-development lifecycle (SSDLC)-Design principles: CIA triad, Zero Trust, privacy-by-design

Software Security Touchpoints: Code review (Tools)-Architectural risk analysis-Software penetration testing- integration with CI/CD Pipelines- Security Requirement Mapping to Touchpoints

System Security and Resilience: Risk based security testing-Abuse cases-Software security operations-Input validation, authentication, and access control-Secure configuration and logging-Defensive programming and exception handling-Software patching, updating, and rollback mechanisms

Privacy, Compliance, and Ethics: Data protection regulations (GDPR, HIPAA, IT Act)-Privacy-enhancing technologies (PETs)-Ethics in intelligent systems (bias, transparency, accountability)-Secure DevOps (DevSecOps practices)

Case Studies and Future Directions: Role of AI/ML in software: personalization, anomaly detection-Smart APIs, agents, and autonomous decision-making-Security issues in IoT-enabled systems-Attacks on ML models (adversarial examples, model inversion)-Blockchain and secure software ecosystems-Trends: Explainable AI (XAI), Trusted Execution Environments (TEE)

Course Outcomes

Upon successful completion, students will be able to:

CO1: Analyze the architecture of intelligent and connected software systems.

CO2: Apply AI/ML concepts for building adaptive software systems.

CO3: Integrate secure coding, threat modeling, and privacy-preserving techniques.

CO4: Evaluate system risks and ensure resilience against software and network-based attacks.

CO5: Design systems that meet security, privacy, and ethical standards.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References:

1. "Software Security: Building Security In" by Gary McGraw, Addison-Wesley, 2006.
2. "Secure and Resilient Software Development" by Mark Merkow, 2020, CRC Press
3. "Security Engineering" by Ross Anderson, 3rd Edition, Wiley, 2020
4. "Building Intelligent Systems: A Guide to Machine Learning Engineering" by Geoff Hulten, Springer, 2018.
5. "Designing Machine Learning Systems" by Chip Huyen, O'Reilly, 2022
6. "Privacy Engineering: A Dataflow and Ontological Approach", by Ian Oliver, Createspace Independent, 2024.
7. International Journal Articles

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, architectures, and security principles of smart and secure software systems.	--	--
CO2	Analyse smart software systems to identify security requirements, vulnerabilities, and risk mitigation strategies.	PO1 (3)	PSO1 (3)
CO3	Evaluate security mechanisms, smart system architectures, and assurance techniques to assess their effectiveness in protecting software systems.	PO3 (2)	PSO2 (2)
CO4	Design smart and secure software solutions by selecting appropriate architectures, security controls, and intelligent features for real-world applications.	PO2 (1)	PSO1 (3)

Course Objectives

1. To introduce the architecture and programming model of Graphics Processing Units (GPUs).
2. To enable students to develop parallel programs using CUDA/OpenCL.
3. To provide practical exposure to optimizing algorithms for GPU architectures.
4. To explore high-performance computing applications using GPUs.

Introduction to Parallel Computing: Motivation for parallel computing-Flynn's taxonomy-Multi-core vs. many-core systems-Parallel Programming Languages and Models-Introduction to GPU architecture (History, CUDA cores, SIMD model)

CUDA Programming Basics: Overview of NVIDIA CUDA and OpenCL-Thread hierarchy: grid, block, warp, thread-Memory hierarchy: global, shared, constant, texture memory-Execution model and synchronization-CUDA program structure-Kernel launches and memory management-Thread indexing and data parallelism-Simple CUDA examples (vector addition, matrix multiplication)

Memory and Performance Optimization: CUDA Device Memory Types-Memory coalescing and shared memory optimization-Data Prefetching and Instruction Mix-Avoiding bank conflicts-Thread Granularity-Occupancy and warp divergence-Use of streams and asynchronous operations

Advanced GPU Programming and Case Studies: Floating-Point Format -Parallel programming and computational thinking-CUDA Unified Memory and dynamic parallelism-Performance profiling using tools (nvprof, Nsight)-Introduction to Tensor Cores and GPU acceleration in AI-GPU use cases in scientific computing, image processing, deep learning

Introduction to OPENCL: Data Parallelism Model-Device Architecture-Kernel Functions-Device Management and Kernel Launch

Course Outcomes

Upon successful completion of this course, students will be able to:

- CO1:** Understand GPU architectures and the parallel computing model.
CO2: Understanding memory management and thread execution
CO2: Write and optimize programs using CUDA or OpenCL.
CO3: Evaluate and enhance the performance of GPU-based algorithms.
CO4: Apply GPU computing for scientific and real-time applications.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

Textbook

1. "Programming Massively Parallel Processors: A Hands-on Approach" by David B. Kirk and Wen-mei W. Hwu, 3rd Edition, Morgan Kaufmann, 2016.

References

1. "CUDA by Example: An Introduction to General-Purpose GPU Programming" by Jason Sanders and Edward Kandrot, Addison-Wesley, 2011.
2. "GPU Computing Gems – Emerald Edition" by Wen-mei W. Hwu (Editor), Morgan Kaufmann, 2011.
3. "CUDA Programming: A Developer's Guide to Parallel Computing with GPUs" by Shane Cook, Morgan Kaufmann, 2013.
4. "Parallel Programming in C with MPI and OpenMP" by Michael J. Quinn, McGraw-Hill, 2003. *(for comparative perspective)*
5. NVIDIA CUDA Toolkit Documentation & Developer Zone – <https://developer.nvidia.com/cuda-zone>

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, architectures, and programming models of GPU computing and its role in high-performance computing systems.	--	--
CO2	Analyze GPU architectures and parallel programming paradigms to understand performance optimization, memory management, and computational efficiency.	PO1 (3)	PSO1 (3)
CO3	Evaluate GPU-based algorithms, frameworks, and hardware configurations to assess their effectiveness in accelerating computational tasks.	PO3 (2)	PSO2 (2)
CO4	Design efficient GPU computing solutions by selecting appropriate parallelization strategies, memory layouts, and programming techniques for real-world applications.	PO2 (1)	PSO1 (3)

Course Objectives

The students will be enabled to:

1. Explore the architecture and evolution of modern web services.
2. Understand API-first development using REST, GraphQL, and gRPC.
3. Learn advanced API management including gateways, observability, and lifecycle.
4. Implement secure, scalable, and maintainable APIs using cloud-native tools.
5. Apply real-time integration, containerization, and CI/CD for API deployment.

Evolution and Foundations of Web Services: History: SOAP → REST → GraphQL/gRPC-HTTP/2 and HTTP/3 for APIs-Webhooks and real-time APIs (SSE, WebSocket)-REST: Richardson Maturity Model-API-first vs. code-first design

RESTful and GraphQL API Design: REST constraints, best practices-Designing REST endpoints and URI schemes-GraphQL:-concepts, schema design (SDL), resolvers-N+1 problem and batching in GraphQL-REST vs. GraphQL vs. gRPC: performance and use cases

API Security, Governance, and Documentation: OpenAPI 3.1 (Swagger), GraphQL Docs-OAuth 2.0, OpenID Connect, API keys, JWT-RBAC and ABAC for APIs-CORS, rate limiting, DoS protection-API lifecycle management and governance

Cloud-Native API Deployment and Management: API Gateways: Kong, Apigee, AWS API Gateway-Service mesh and API mesh (Istio, Linkerd)-Load balancing and circuit breakers-Docker, Kubernetes for API containerization-CI/CD pipelines for API deployment (GitHub Actions, Jenkins)

Observability, Monitoring & Advanced Topics: Logging, metrics, tracing (ELK, Prometheus, Jaeger)-API analytics and monetization-Versioning strategies: URI, headers, media types-API Orchestration and Choreography-Case study: Microservice-based API architecture in fintech/healthcare

Course Outcomes

Upon successful completion, the student will be able to:

CO1: Compare and contrast modern API paradigms (REST, GraphQL, gRPC).

CO2: Build and document production-grade APIs with OpenAPI and GraphQL SDL.

CO3: Implement API security, rate limiting, and versioning strategies.

CO4: Deploy and manage APIs using API gateways and service meshes.

CO5: Monitor, scale, and orchestrate APIs in a cloud-native ecosystem.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References

1. Mark Masse, *REST API Design Rulebook*, O'Reilly, 2011.
2. Leonard Richardson et al., *RESTful Web APIs*, O'Reilly, 2013.
3. Alex Banks & Eve Porcello, *Learning GraphQL*, O'Reilly, 2018.
4. *API Security in Action*, Neil Madden, Manning
5. *Microservices Patterns*, Chris Richardson, Manning
6. OWASP API Security Top 10
7. API Gateway & Service Mesh Docs: Istio, Kong, Apigee
8. GraphQL Official Docs: <https://graphql.org>
9. OpenAPI Specification: <https://swagger.io/specification/>

CO Description of CO PO Mapping PSO Mapping

CO1	Describe the fundamentals of web services, RESTful and SOAP architectures, and API design principles.	PO1 (3)	PSO1 (3)
CO2	Design and implement RESTful APIs and web services using modern frameworks and tools.	PO3 (3)	PSO2 (3)
CO3	Analyze and evaluate API performance, security, and scalability for real-world applications.	PO2 (2)	PSO2 (2)
CO4	Apply best practices in API documentation, versioning, and integration to build maintainable and robust web services.	PO3 (2)	PSO1 (3)

Course Objective:

The course on Decentralized Systems is designed to provide students with a comprehensive understanding of the fundamental principles and architectures that underpin decentralized computing. It introduces key concepts such as peer-to-peer models, distributed trust, and consensus mechanisms, which are essential for building and maintaining decentralized networks. As students engage with these topics, they gain valuable insights into how decentralized systems operate without a central authority, helping them to understand the practical advantages and limitations of such models. A major focus is on blockchain technology, its structure, and how it enables the creation of smart contracts and decentralized applications, which helps students connect theoretical knowledge with real-world innovations.

Introduction to Decentralized Systems: Definition, Characteristics, Components-Comparison: Centralized vs Distributed vs Decentralized-Benefits: Fault Tolerance, Autonomy, Transparency-Challenges: Consistency, Latency, Security-Data Models: Eventual vs Strong Consistency-Fault Tolerance, Scalability Issues-Peer-to-Peer Systems: Structured and Unstructured-Blockchain Basics: Blocks, Nodes, Transactions, Consensus-Federated Learning & Hybrid Models

Activities:

- Role-play of replicated databases and client behavior under network failures
- Weekly lightning talks on real-world decentralized tech (e.g., Web3, DePIN)

Core Technologies and Algorithms: Consensus Mechanisms and Distributed Algorithms-Coordination and Fault Tolerance-Smart Contracts: Automation, Trust, Use Cases-Cryptography: Symmetric/Asymmetric, Hashing, Digital Signatures-Zero-Knowledge Proofs and Security in Decentralized Systems

Activities:

- Build simple encryption/decryption tools (AES, RSA)
- Debate: Smart Contracts vs Legal Contracts

Design and Implementation: Autonomous Agents and Robotics Architecture-Rule-based Systems and Decision Making-Basics of Reinforcement Learning-Sensor Integration, Calibration, and Fusion Techniques

Activities:

- Design a rule-based robot (e.g., smart cleaner)
- Model components of a self-driving vehicle

Ethical and Legal Considerations: Ethics in Autonomy: Accountability, Bias, Transparency-Legal Issues: Liability, Data Privacy, Regulatory Compliance-Social Impact: Employment, Human–Machine Interaction, Inclusion

Activities:

- Analyze redundancy and failure mitigation in real systems
- Study ethical and legal issues in a deployed autonomous system

Safety and Reliability in Systems: Fail-safe Design Principles-Redundancy and Fault Tolerant Architectures-Risk Assessment, Hazard Analysis, Safety Certification

Activities:

- Case study of a system failure with reliability analysis
- Propose improvements to enhance system safety

Assessment Weightage:

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References

1. "A Survey on Decentralized Federated Learning: A Survey and Perspective" Liangqi Yuan *et al.* (2023) [Link to arXiv PDF](#)
2. Recent advances in reinforcement learning-based autonomous ...", 2024, ScienceDirect.
3. A Survey of Security Strategies in Federated Learning" MDPI (2022) <https://www.mdpi.com/1999-5903/16/10/374>
4. A Comparative Survey of Centralised and Decentralised Identity Management" Castiglione *et al.* (2023) <https://www.mdpi.com/1999-5903/17/1/1>

E- Sources**E-Books :**

1. "Foundations of Distributed Consensus and Blockchains" – Elaine Shi
2. "Distributed Consensus Protocols and Algorithms" – Ning & Yan (2020 paper)

Course Outcomes:

CO1: Analyze the structure, components, and benefits of decentralized systems compared to centralized and distributed systems.

CO2: Apply cryptographic techniques such as encryption, digital signatures, and hash functions to ensure data security and trust.

- CO3:** Demonstrate the architecture and behavior of autonomous agents and robots in decentralized systems.
- CO4:** Discuss the ethical and legal challenges in deploying autonomous and decentralized systems.
- CO5:** Analyze risk in autonomous systems through risk modeling, hazard analysis, and safety certification practices.

CO	Description of CO	PO	PSO
CO1	Analyze the structure, components, and benefits of decentralized systems compared to centralized and distributed systems.	PO1 (3) PO2 (2) PO9 (2) PO11 (2)	PSO1 (2) PSO2 (2)
CO2	Apply cryptographic techniques such as encryption, digital signatures, and hash functions to ensure data security and trust.	PO1 (3) PO2 (3) PO3 (2) PO5 (2) PO10 (2)	PSO1 (3) PSO2 (2)
CO3	Demonstrate the architecture and behavior of autonomous agents and robots in decentralized systems.	PO1 (3) PO2 (3) PO4 (3) PO6 (2) PO10 (2)	PSO1 (2) PSO2 (2)
CO4	Discuss the ethical and legal challenges in the deploying autonomous and decentralized systems.	PO1 (3) PO2 (2) PO3 (2) PO11 (2)	PSO1 (2) PSO2 (2)
CO5	Analyze risk in autonomous systems through risk modeling, hazard analysis, and safety certification practices.	PO1 (3) PO2 (3) PO4 (3) PO6 (2)	PSO1 (2) PSO2 (2)

Course Objectives

1. To understand the foundational concepts of cloud computing, service models, and deployment architectures.
2. To explore the core technologies including virtualization, cloud networking, and distributed storage.
3. To gain hands-on experience in designing and managing cloud-based solutions using AWS, Azure, and GCP.
4. To evaluate the challenges, security concerns, and advanced applications in modern cloud environments.

Cloud Foundations & Service Models: Cloud characteristics, service models (IaaS, PaaS, SaaS), deployment models, benefits, case studies from AWS, Azure, GCP

Virtualization & Cloud Architecture: Hypervisors, containers (Docker), Kubernetes, VM management, multitenancy, resource pooling

Cloud Networking and Geo-Distribution: VPCs, Subnetting, Load balancers, CDNs, DNS in cloud, Cloud networking in AWS/Azure/GCP, Geo-replication, Global app deployment

Fault Tolerance, Availability, and Debugging: Fault tolerance vs. availability, High Availability (HA), Load balancing, Failover strategies, Diagnosing/debugging cloud services, Health checks, monitoring

Cloud Programming & Storage: MapReduce, Hadoop/Spark, Serverless (FaaS), Cloud storage (S3, Blob, GCS), Object vs. block storage, Data consistency, Geo-replicated storage

Cloud Security and Compliance: IAM, Data encryption, Secure containers & VMs, Audit trails, Security in AWS/Azure/GCP, Compliance (GDPR, HIPAA, SOC2)

Cloud Native Apps & Emerging Trends: Microservices, CI/CD pipelines, DevOps in cloud, Observability, AI/ML workloads in cloud, Fog/Edge computing, Sustainable cloud, Quantum Cloud

Course Outcomes (COs)

CO1: Explain the fundamental concepts, evolution, and different models of cloud computing including IaaS, PaaS, and SaaS.

CO2: Identify and describe the key components of cloud infrastructure such as virtualization, containers, and cloud storage.

CO3: Analyze cloud networking architectures, geo-replication strategies, and techniques for achieving fault tolerance and high availability.

CO4: Demonstrate the ability to diagnose and debug distributed cloud applications using monitoring and logging tools in AWS, Azure, or GCP.

CO5: Implement secure cloud solutions by applying IAM, encryption techniques, and compliance policies across different cloud platforms.

CO6: Assess real-world applications, trends, and research directions in cloud-native development, serverless computing, and AI/ML integration in cloud.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

Reference Books

1. Dan C. Marinescu, *Cloud Computing: Theory and Practice*,
Publisher: Morgan Kaufmann (Elsevier Inc.)
Edition: 2nd Edition
Year: 2018
2. Rajkumar Buyya, *Cloud Computing: Principles and Paradigms*,
Publisher: Wiley
Year: 2011
3. Michael J. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*,
Publisher: Wiley
Year: 2014
4. Tom Laszewski, Kamal Arora, Erik Farr, Piyum Zonooz, *Cloud Native Architectures*,
Publisher: Packt Publishing
Year: 2018
5. Tim Mather, Subra Kumaraswamy, Shahed Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*,
Publisher: O'Reilly Media
Year: 2009
6. Cloud Platform Documentation and Tutorials:
 - o Amazon Web Services (AWS) Documentation – <https://docs.aws.amazon.com>
 - o Microsoft Azure Documentation – <https://learn.microsoft.com/en-us/azure>
 - o Google Cloud Documentation – <https://cloud.google.com/docs>
(Updated regularly, used for labs and real-world implementation)

Mapped Books / Tools

Marinescu Ch. 1-2, Buyya Ch. 1, AWS/Azure/GCP Docs
 Marinescu Ch. 3, Kavis Ch. 3-4, GCP/AWS Compute Docs
 Laszewski Ch. 5-6, AWS VPC/Route53, Azure Traffic Manager, GCP Network
 Marinescu Ch. 4, AWS CloudWatch, Azure Monitor, GCP Stackdriver
 Marinescu Ch. 5, Buyya Ch. 8, AWS Lambda, Azure Functions
 Mather Ch. 2-6, Marinescu Ch. 8, AWS IAM, Azure AD, GCP IAM
 Laszewski Ch. 9-10, AWS SageMaker, Azure ML, GCP Vertex AI

Description of Course Outcomes and Mapping

CO	Description of CO	PO	PSO
CO1	Describe the advanced concepts, architectures, and services in cloud computing and their significance in scalable and distributed applications.	--	--
CO2	Analyse cloud platforms, virtualization techniques, and service models to understand performance, resource management, and scalability issues.	PO1 (3)	PSO1 (3)
CO3	Evaluate cloud computing technologies, deployment strategies, and security mechanisms to assess their effectiveness for enterprise and real-world applications.	PO3 (2)	PSO2 (2)
CO4	Design advanced cloud-based solutions by selecting appropriate architectures, services, and deployment strategies to meet specific application requirements.	PO2 (1)	PSO1 (3)

Course Objectives:

1. To understand the foundational concepts of AR and VR.
2. To explore the key technologies behind AR and VR systems.
3. To gain hands-on experience in creating AR/VR content.
4. To evaluate the applications and challenges in AR/VR development.

Introduction to AR and VR: Importance and applications of AR and VR – History and evolution – Differences between AR and VR – Basics of Computer Vision – Multimodal interaction – Components and features of VR systems – Recent developments in AR/VR.

Augmented Reality Concepts: Displays – Taxonomy, technology, and features of Augmented Reality – Challenges with AR – AR systems and functionality – Major software and hardware components for AR – Software architectures – Creating Augmented Reality content.

Principles and Practices of Augmented Reality: Augmented Reality methods – Visualization techniques for AR – Wireless displays in educational AR applications – Mobile projection interfaces – Marker-less tracking for AR – Enhancing interactivity in AR environments – Evaluating AR systems.

Introduction to Virtual Reality: Computer graphics – Real-time computer graphics – Flight simulation – The virtual world space – Positioning the virtual observer – The perspective projection – Human vision – Stereo perspective projection – 3D clipping – Color theory – Simple 3D modeling – Illumination models – Reflection models – Shading algorithms – Radiosity – Hidden surface removal – Realism – Stereographic image.

Interactive Techniques in Virtual Reality: Introduction to 2D and 3D concepts – From 2D to 3D – 3D space curves – 3D boundary representation – Geometrical transformations – Frames of reference – Modeling transformations – Instances – Picking – Flying – Scaling the virtual environment – Collision detection – Generic VR system – Introduction to virtual environment – Computer environment – VR technology – Model of interaction – VR systems.

Visual Computation in Virtual Reality: Animating the virtual environment – The dynamics of numbers – Linear and nonlinear interpolation – The animation of objects – Linear and nonlinear translation – Shape and objects – Free-form deformation – Particle systems – Physical simulation – Introduction to simulation concepts – Objects falling in a gravitational field – Rotating wheels – Elastic collisions – Projectiles – Simple pendulum – Springs – Flight dynamics of an aircraft.

Applications of AR, VR, and Mixed Reality: Augmented Reality applications – Future of AR – Present and future state of VR – Convergence of AR and VR – Mixed Reality platforms and use-cases – Emerging industrial and research applications.

Course Outcomes (COs):

CO1: Explain the fundamental concepts, history, and differences between Augmented Reality and Virtual Reality systems.

CO2: Identify and describe the software and hardware components required for developing AR systems, and evaluate their functionalities and limitations.

CO3: Analyze various visualization techniques, tracking methods, and interactive features used in Augmented Reality applications.

CO4: Demonstrate knowledge of core computer graphics principles and interactive 3D modeling techniques used in Virtual Reality environments.

CO5: Implement and simulate basic animation and physical simulation models for interactive VR environments.

CO6: Assess real-world applications and emerging trends in AR, VR, and Mixed Reality and propose potential areas of innovation.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References:

1. Deiter Schmalstieg, Tobias Hollerer, Augmented Reality, Principles and Practices. 2014, Addison Wesley - 40%.
2. Burdea, G. C. and P. Coffet. Virtual Reality Technology, Second Edition. Wiley-IEEE Press, 2003/2006. 60%.
3. Tom Dieck, M. Claudia, Jung, Timothy, Correia Loureiro, Sandra Maria, Augmented Reality and Virtual Reality, New Trends in Immersive Technology. Springer publications. (Edited Book), 2021.
4. Alan Craig, William Sherman and Jeffrey Will, Developing Virtual Reality Applications, Foundations of Effective Design, Morgan Kaufmann, 2009.
5. Anand R., "Augmented and Virtual Reality", Khanna Publishing House, Delhi.
6. Alan B. Craig, Understanding Augmented Reality, Concepts and Applications, Morgan.
7. *Learning Augmented Reality Development* – Joseph Hocking, Packt Publishing.
8. *3D User Interfaces: Theory and Practice* – Doug Bowman, Ernst Kruijff et al.

CO Description of CO PO Mapping PSO Mapping

CO1	Describe the fundamental concepts, architectures, and devices used in AR and VR systems.	PO1 (3)	PSO1 (3)
CO2	Analyze AR/VR applications to identify requirements, interaction techniques, and usability challenges.	PO2 (3)	PSO2 (3)
CO3	Design and implement AR/VR experiences using modern development tools and platforms.	PO3 (3)	PSO1 (3)
CO4	Evaluate the effectiveness, performance, and user experience of AR/VR applications for real-world scenarios.	PO2 (2)	PSO2 (2)

SE25015 Software Requirements Engineering L T P C
3 0 0 3

Course Objectives

- To understand the role of requirements in software development.
- To explore various types of requirements and how to elicit, analyze, model, and validate them.
- To learn techniques for requirements documentation and management.
- To understand requirements engineering in agile and traditional processes.
- To apply formal and semi-formal techniques in real-world scenarios.

Introduction to Requirements Engineering: Definition – Importance of Requirements – Types of Requirements – Requirements Engineering Process – Role of Requirements in Software Engineering – Requirements Problems – Stakeholders – Requirement Qualities – IEEE Standards – Introduction to Requirements Documentation.

Requirements Elicitation: Elicitation Techniques – Interviews, Workshops, Observation, Scenarios, Prototyping – Elicitation Challenges – Requirements Gathering vs Elicitation – Context Modeling – Rich Pictures – Use Cases – Personas.

Requirements Analysis and Modelling: Requirements Prioritization – Conflicts and Negotiation – Consistency and Feasibility – Modeling Requirements using UML – Goal-Oriented Modeling – Data Modeling – Functional and Non-Functional Requirements Analysis.

Requirements Documentation and Validation: SRS Structure – Writing Good Requirements – Review Techniques – Validation and Verification – Prototyping – Requirements Inspections – Acceptance Criteria – Traceability – Requirements Reuse.

Requirements Management and Tools: Change Management – Requirements Baseline – Version Control – Tools for Requirements Management – Managing Requirements in Agile – Case Studies – Introduction to Tools like IBM DOORS, JIRA, RequisitePro.

Course Outcomes (COs)

CO Code	Course Outcome
CO1	Understand and explain the importance and principles of Software Requirements Engineering.
CO2	Apply suitable techniques to elicit requirements from stakeholders.
CO3	Analyze and model functional and non-functional requirements effectively.
CO4	Prepare well-structured and quality Software Requirements Specification (SRS) documents.
CO5	Manage and validate requirements using tools and agile-compatible practices.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References

- Ian Sommerville, "Software Engineering", 10th Edition, Pearson Education, 2015.
- Klaus Pohl, "Requirements Engineering: Fundamentals, Principles, and Techniques", Springer, 2010.
- Elizabeth Hull, Ken Jackson, Jeremy Dick, "Requirements Engineering", Springer, 4th Edition, 2022.
- Karl Wieggers and Joy Beatty, "Software Requirements", Microsoft Press, 3rd Edition, 2013.
- Dean Leffingwell and Don Widrig, "Managing Software Requirements", Addison Wesley, 2nd Edition, 2003.

CO Description of CO PO Mapping PSO Mapping

CO1	Describe the fundamentals, processes, and techniques of software requirements engineering.	PO1 (3)	PSO1 (3)
CO2	Elicit and analyze software requirements using appropriate techniques to understand stakeholder needs.	PO2 (3)	PSO2 (3)
CO3	Specify and model software requirements using standard notations and tools.	PO3 (3)	PSO1 (3)
CO4	Validate, verify, and manage software requirements to ensure completeness, consistency, and quality.	PO2 (2)	PSO2 (2)

CP25C19	Cognitive Computing	L	T	P	C
		3	0	0	3
<p>Course Objectives:</p> <ul style="list-style-type: none"> • To understand the core principles of cognitive computing and how they emulate human thought processes. • To apply cognitive computing to solve complex, unstructured, and real-world problems. • Develop cognitive computing applications. 					
<p>Human cognition - Introduction to Cognitive Computing – Elements of a Cognitive System - Design Principles for Cognitive Systems - Artificial Intelligence, Natural Language Processing and Machine Learning in Cognitive Computing.</p> <p>Activities:</p> <ol style="list-style-type: none"> 1. Concept Quizzes: Human cognition and Cognitive Computing 2. Research paper review on cognitive systems 					
<p>Cognitive architectures: ACT-R, Soar, OpenCog - Knowledge Representation and Reasoning - Machine Learning models in cognitive systems - Human-like Learning: Supervised, Unsupervised, Reinforcement</p> <p>Activities: Demo on IBM Watson APIs (Language Translator)</p>					
<p>Role of NLP in a Cognitive System - NLP pipeline: POS tagging, NER, parsing - Sentiment Analysis and Question Answering</p> <p>Debate: Sentiment analysis and its challenges in Regional Languages.</p> <p>Semantic Web - Semantic Understanding: Ontologies and Knowledge Graphs</p> <p>Activities: Ideathon: Innovative solution to health care problems.</p>					
<p>Cognitive Agents - Rule-based Agents and Expert Systems - Decision Support Systems -Emotion-aware computing and human-computer interaction- Cognitive bias and ethics in decision making</p> <p>Activities: Mini project: Develop health care chatbot. Prototype Demonstrations.</p>					
<p>Process of Building a Cognitive Application - IBM's Watson as a Cognitive System- Business Implications of Cognitive Computing</p> <p>Activities: Research Paper Review: Cognitive Computing</p>					

Cognitive Computing Applications: Healthcare, Finance, Education, Legal, Robotics- Building a Cognitive Healthcare Application - Emerging Cognitive Computing Areas - Cognitive cloud- Cognitive IoT - Cognitive Chatbots - Neuromorphic computing and brain-inspired models.

Activities:

1. Prototype Demonstrations.
2. Project demo and peer review.

Weightage: Continuous Assessment: 40%, End Semester Theory Examinations: 60%

Assessment Methodology: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)

References

1. Velankar, Mahalle & Shinde – Cognitive Computing for Machine Thinking, CRC Press, 2024.
2. Judith S. Hurwitz, Marcia Kaufman, Adrian Bowles – Cognitive Computing and Big Data Analytics, Wiley, 2015.
3. Peter Fingar , Cognitive Computing: A Brief Guide for Game Changers, Meghan Kiffer, 2015.
4. Building Cognitive Applications with IBM Watson Services, Redbooks, 2017
5. Steven Bird, Ewan Klein, Edward Loper, Natural Language Processing with Python, O’Reilly, 2009.
6. Fuchun Sun, Qinghu Meng, Zhumu Fu, Bin Fang (Editors), Communications in Computer and Information Science (CCIS, volume 1918) – Springer Series, 2024.

E-Resources

- Prof. Sharba Bandyopadhyay, Prof. Rajlakshmi Guha | IIT Kharagpur, IIT Kharagpur
Link https://onlinecourses.nptel.ac.in/noc22_ee122/preview
- Cognition and its Computation, IIT Kharagpur
Prof. Sharba Bandyopadhyay, Prof. Rajlakshmi Guha
Link “<https://nptel.ac.in/courses/108105185>”

CO	Description of CO	PO	PSO
CO1	Describe the fundamental concepts, architectures, and paradigms of cognitive computing and their role in intelligent decision-making systems.	--	--
CO2	Analyze cognitive computing models and learning mechanisms to understand perception, reasoning, and knowledge representation in intelligent systems.	PO1 (3)	PSO1 (3)
CO3	Evaluate cognitive computing techniques, frameworks, and applications to assess their effectiveness in solving complex real-world problems.	PO3 (2)	PSO2 (2)
CO4	Design cognitive computing solutions by selecting appropriate models, algorithms, and system architectures for human-centric intelligent applications.	PO2 (1)	PSO1 (3)

Course Objectives:

Students are able to

1. Understand the Foundations of Language Models
2. Master the Design and Training of LLMs
3. Apply Advanced Techniques for Fine-Tuning and Alignment
4. Evaluate and Deploy LLMs Responsibly
5. Explore Emerging Trends and Applications

Fundamentals of NLP and Language Models: Introduction to Natural Language Processing-Statistical Language Models-Word Representation Techniques: Word2Vec, GloVe-Tokenization Strategies: Whitespace, Subword Tokenization-Introduction to Transformer Architecture-Self-Attention Mechanism

Transformer Architecture Deep Dive: Encoder Layers and Encoder Stack-Decoder Layers and Decoder Stack-Position Encoding-Teacher Forcing and Masked Attention-Batch Normalization and Layer Normalization

Architectures of Large Language Models: Decoder-only LLMs (e.g., GPT)-Encoder-only LLMs (e.g., BERT)-Encoder-Decoder Models (e.g., BART, T5)-Pre-training vs Fine-tuning-Pre-Training Strategies: Encoder-Decoder and Decoder-only Models-Training Objectives: Masked Language Modeling, Causal Language Modeling

Tokenization and Decoding Strategies: Tokenization Methods: Byte Pair Encoding (BPE), WordPiece, SentencePiece-Decoding Strategies:Greedy Search-Beam Search-Top-k Sampling-Top-p (Nucleus) Sampling

Instruction Tuning and Prompt Engineering: Types of Training Methods for LLMs-Instruction Tuning-Prompt-based Learning-Advanced Prompting Techniques-Prompt Sensitivity and Prompt Injection

Alignment and Optimization of LLMs: RLHF: Aligning LLMs with Human Feedback-Learning Schedules and Gradient Clipping-Typical Training Failures and Debugging Strategies-Overview of Training Pipelines: Gopher, OPT, LLaMA, Falcon, SETU-Clean Data Importance and Data Curation Studies

Comparative Study of LLMs and Ecosystem Tools: Major LLMs: GPT-2, GPT-3, T5, BART, BERT, LLaMA, Falcon, OPT, Gopher-Differences and Similarities Between Architectures-Data Sources: C4, mC4, Pile, RedPajama, SlimPajama, Stack, Sangraha-Tools and Ecosystem: Introduction to HuggingFace-Pipelines for Pre-training and Fine-tuning LLMs

Course Outcomes (COs)

CO Code	Course Outcome Statement
CO1	Understand and explain core NLP concepts, transformer models, and training strategies.
CO2	Analyze and apply tokenization, decoding, and deep learning methods for text understanding.
CO3	Design and implement transformer-based NLP applications using pre-trained models.
CO4	Investigate and evaluate training techniques, prompting strategies, and alignment approaches.
CO5	Compare LLM architectures and assess their effectiveness in real-world NLP applications.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

Learning Resources

References

1. Tanmoy Chakraborty, Introduction to Large Language Models, Generative AI for Text, Wiley India, 1st Edition, 2025.
2. Sanket Subhash Khandare, Mastering Large Language Models, Advanced Techniques, Applications, Cutting-edge Methods, and Top LLMs (English Edition), 1st Edition, Bpb Publications, 2024.
3. Jay Alamar and Maarten Grootendorst, Hands-on Large Language Models, Language Understanding and Generation, O'Reilly, 1st Edition, 2024.
4. Sinan Ozdemir, Quick Start Guide to Large Language Models, Strategies and Best Practices for Using ChatGPT and Other LLMs, Pearson Education, 1st Edition, 2023.
5. Raj Arun R, Mastering Large Language Models with Python, AVA, 1st Edition, 2024.

CO1	Describe the fundamentals, architectures, and training techniques of large language models.	PO1 (3)	PSO1 (3)
CO2	Analyze and evaluate the performance, limitations, and ethical considerations of LLMs in various applications.	PO2 (3)	PSO2 (3)

CO3	Design and fine-tune LLM-based solutions for natural language processing tasks and real-world problems.	PO3 (3)	PSO1 (3)
CO4	Implement and optimize LLMs using modern frameworks and tools while ensuring scalability and efficiency.	PO2 (2)	PSO2 (2)

Course Objective:

This course introduces the software platforms, operating systems, middleware, and programming models that power Internet of Things (IoT) systems. It focuses on embedded OS, real-time constraints, device interoperability, middleware frameworks, edge-cloud integration, and service management in IoT ecosystems.

Introduction to IoT Software Stack: Overview of IoT software architecture, Sensors, actuators, and device firmware, Role of microcontrollers and real-time OS, Introduction to software stack layers: device, network, service, and application.

Concept Mapping: IoT Architectures

Embedded Systems and IoT OS: Characteristics of embedded and constrained systems, Real-Time Operating Systems (RTOS) for IoT: FreeRTOS, RIOT, Contiki, Scheduling, memory management, and inter-process communication, Power-aware programming and sensor interfacing. **Quiz:** RTOS

Middleware for IoT: IoT middleware architecture and functions, Service discovery and device management, Data acquisition, storage, and edge analytics, Middleware examples: Kaa, FIWARE, Eclipse IoT

Poster: Middleware comparisons

IoT Communication and Protocol Stack: Protocols: MQTT, CoAP, HTTP, AMQP, XMPP, Publish-subscribe vs. request-response models, Security mechanisms in constrained protocols, Transport and application layer adaptations for IoT, **Reproduction of research paper:** papers on IoT Protocols stack optimization in smart environment

Cloud and Edge Software Integration: Fog computing and edge cloud platforms, Device-to-cloud communication patterns, IoT platforms: AWS IoT Core, Azure IoT Hub, Google Cloud IoT, Data stream processing and visualization tools

Flipped classroom: Cloud Platforms for IoT Data Processing

IoT Application Development Frameworks: Node-RED, Blynk, Arduino, and PlatformIO, Integration with APIs and cloud services, Event-driven programming and rule engines,

Case study/Model making: Smart Home / Smart Health system implementation using IoT Application Development Frameworks

Challenges in IoT Software Systems: Interoperability and standardization issues, Scalability and maintainability of IoT applications, Security & privacy concerns in the software stack, Lifecycle and update management in IoT deployments.

Seminar: Scalability and maintainability of IoT applications

Course Outcomes:

1. Identify software components and the architecture of IoT systems.
2. Develop embedded software for resource-constrained IoT devices using RTOS.
3. Apply middleware solutions for device communication and management.
4. Evaluate and implement cloud/edge-based IoT software systems.
5. Use development frameworks and tools to prototype IoT applications.

Weightage:	Continuous Assessment: 40%	End Semester Theory Examinations: 60%
	(i) Activity: 10% (ii) Internal Theory Examination: 30%	
Mandated Activities with Marks: Assignments (30), Quiz (10), Virtual Demo (25), Flipped Class Room (10), Review of Gate and IES Questions (25)		
Internal Examinations: Two Tests		

References:

1. Raj Kamal, Internet of Things: Architecture and Design Principles, McGraw-Hill, 2nd Edition, 2022.
2. Honbo Zhou, The Internet of Things in the Cloud: A Middleware Perspective, CRC Press, 2012.
3. Adrian McEwen & Hakim Cassimally, Designing the Internet of Things, Wiley, 2014.
4. Dieter Uckelmann, Mark Harrison, Florian Michahelles, Architecting the Internet of Things, Springer, 2011.
5. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stamatias Karnouskos, From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, Academic Press, 2014.
6. Eclipse IoT Project Resources, <https://iot.eclipse.org>
7. An Introduction to Programming the Internet of Things (IOT) Specialization, <https://www.coursera.org/specializations/iot>

CO Description of CO PO Mapping PSO Mapping

CO1	Describe the architecture, protocols, and software components used in IoT systems.	PO1 (3)	PSO1 (3)
CO2	Analyze IoT requirements and design software solutions for sensor networks, data collection, and communication.	PO2 (3)	PSO2 (3)
CO3	Develop and implement IoT applications using cloud integration, edge computing, and middleware platforms.	PO3 (3)	PSO1 (3)
CO4	Evaluate IoT software systems for performance, security, scalability, and reliability in real-world scenarios.	PO2 (2)	PSO2 (2)