Appendix-90
Resolution No. 4-1-7

Department of Computer Science and Engineering
Faculty of Technology
University of Delhi

Detailed Course Structure and Curriculum of B.Tech. (CSE) Fourth Year

Department of Computer Science and Engineering
Faculty of Technology
University of Delhi

Course Structure of B.Tech. (CSE) Fourth Year

Semester VII						
S. No.	**Course Domain**	**Course Title**	**Credits***			**Total Credits**
			L	**T**	**P**	
1	DSC-19	Cyber Physical Systems	3	0	1	4
2	DSE-5	Select a course from the specified list of DSE-5				4
3	DSE-6 / GE-7	Select a course from the specified list of DSE-6 / GE-7				4
4	DSE-7 / GE-8	Select a course from the specified list of DSE-7 / GE-8				4
5	Dissertation on Major or Dissertation on Minor / Specialisation or Academic Project / Entrepreneurship					6
					Total Credits	**22**

Semester VIII						
S. No.	**Course Domain**	**Course Title**	**Credits***			**Total Credits**
			L	**T**	**P**	
1	DSC-20	Fairness, Bias & Responsible Ethics in AI	3	0	1	4
2	DSE-8	Select a course from the specified list of DSE-8				4
3	DSE-9 / GE-9	Select a course from the specified list of DSE-9 / GE-9				4
4	DSE-10 / GE-10	Select a course from the specified list of DSE-10 / GE-10				4
5	Dissertation on Major or Dissertation on Minor / Specialisation or Academic Project / Entrepreneurship					6
					Total Credits	**22**

**Credits*
L (01 Credit) is equivalent to 01 contact hour per week.
T (01 Credit) is equivalent to 01 contact hour per week.
P (01 Credit) is equivalent to 02 contact hours per week.

Department of Computer Science and Engineering
Faculty of Technology
University of Delhi

Pool of DSEs offered by the Department of CSE in Fourth Year

S. No.	Semester	Course Category	DSE
1.	VII	DSE-5	Web Analytics
2.			Wireless Communication
3.		DSE-6	Enterprise Application Programming
4.			Software Defined Networking
5.		DSE-7	Distributed Systems
6.			Foundations of Quantum Computing
7.	VIII	DSE-8	Bio-inspired Computing
8.			ML and DL for 3D Data
9.		DSE-9	Sustainable and Green Computing
10.			Computing for Smart Cities and Intelligent Infrastructure
11.		DSE-10	Human Centered Computing
12.			Quantum Computing and Applications

Department of Computer Science and Engineering
Faculty of Technology
University of Delhi

Specializations and Minor offered by the Department of CSE

S. No.	Sem	GE	Minor in CSE (Open only for ECE/EE)	Specializations (Open only for CSE) / Minors (Open for ECE/EE)				
				Artificial Intelligence & Machine Learning	Data Science	Software Engineering	Blockchain and Cybersecurity	Augmented Reality / Virtual Reality
1	VII	DSE-6 / GE-7	Introduction to Data Analytics & Data Visualization	Edge Computing	Deep Learning Models and Applications	Software Quality Assurance and Metrics	Smart Contracts and DApps	3D Animation Design
2	VII	DSE-7 / GE-8	Fundamentals of Algorithms	Reinforcement Learning	Big Data Analytics	Software Maintenance and Evolution	Digital Currencies and Blockchain	Multiuser and Social AR/VR
3	VIII	DSE-9 / GE-9	Low level Software Systems	Generative AI	Visual Computing	Program Analysis and Verification	Cyber Law and Digital Forensics	AR/VR application development
4			OOP Concepts	Scientific Machine Learning	Applied AI	Empirical Software Engineering	Cryptanalysis & Security Models	Computer vision for AR/VR
5	VIII	DSE-10 / GE-10	Fundamentals of Machine Learning	Automated Machine Learning	Applied Generative AI for Data Science	Secure Software Engineering	AI in Cybersecurity	AR/VR Hardware Systems: Concepts and Architectures
6			Fundamentals of Cloud Computing	Federated Learning	Data Privacy, Security & Regulatory Compliance	Build, Release and Deployment Engineering	Quantum Cryptography	Ethics, Privacy, and Safety in XR Systems

Detailed Syllabus of Discipline Specific Core (DSC) Courses of B. Tech. (CSE)
SEMESTER VII

Cyber Physical Systems (DSC-19)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Cyber Physical Systems	4	3	0	1	Computer Networks

Course Objectives:

1. Explain CPS concepts and characterize cyber-physical interaction using standard CPS terminology and viewpoints.
2. Develop CPS models using discrete, timed, and hybrid abstractions suitable for analysis and design.
3. Analyze timing concerns in CPS (latency, synchronization, timing-aware behavior) using NIST timing guidance and CPS timing models.
4. Design CPS data/telemetry and decision layers, including resource-aware intelligence placement.
5. Apply trustworthiness thinking (safety, security, updates, assurance) to CPS deployments.
6. Map CPS device/edge-node security requirements to the NIST IoT device cybersecurity capability baseline.

Course Outcomes:

At the end of this course, students will be able to:

1. Define CPS and justify CPS boundaries and stakeholder concerns using NIST CPS facets/aspects.
2. Construct discrete and hybrid CPS models and interpret safety requirements over system behaviors.
3. Evaluate CPS timing needs using timed models and apply scheduling/synchronization concepts to networked CPS.
4. Design CPS data/telemetry and decision layers, including resource-aware intelligence placement.
5. Design a CPS data/telemetry pipeline and assess discretization/quantization effects on decisions.
6. Position AI/ML within a CPS architecture and justify tradeoffs for reliability and resilience.
7. Produce a CPS assurance checklist: threats, secure update plan, and device capability mapping using NISTIR 8259A and CPS framework facets/aspects.

UNIT-I ***(12 hours)***

CPS Foundations and NIST CPS Framework Orientation: CPS meaning, motivation, and examples, CPS as an interaction of computational and physical dynamics, CPS modeling viewpoint (state-based and composition intuition), hybrid systems orientation, safety requirements as a CPS driver, NIST CPS Framework orientation-facets (conceptualization, realization, assurance) and aspects (functional, business, human, trustworthiness, timing, data, composition, boundaries, lifecycle), mapping a CPS scenario to facets/aspects and identifying key constraints and stakeholder needs.

UNIT-II ***(10 hours)***

CPS Modeling with Time and Real-Time Timing Foundations: Timed models for CPS, real-time scheduling fundamentals for CPS workloads, synchronization in distributed CPS and basic clock/time coordination notions, timing failures and their system impact, NIST timing themes - time awareness needs, timing/latency concerns in CPS.

UNIT-III ***(11 hours)***

CPS Data, Networks, and Decision-Making: Data vs information in CPS, information representations and limitations from data-to-model abstractions, network types and processes on networks for CPS communication and coordination, the CPS decision/action viewpoint: CPS "three-layer" framing (physical sensing/measuring layer, data/informing layer, decision/acting layer), control/decision logic for CPS: rule-based supervisors and optimization-based control; link decisions to sensing-actuation loops. Introduction to data-driven (ML-based) models in CPS as an enabling technology.

UNIT-IV ***(12 hours)***

Trustworthy CPS - Safety, Security, Assurance, and NIST-Aligned Deployment: CPS trustworthiness motivation and risk framing (cyber-physical attacks and cross-domain impacts), holistic risk management and why CPS security/safety/privacy must not be handled in silos, failures and layer-based attacks, CPS security topics, timing-security and resilient time as part of assurance, device/edge/IoT capability alignment using the NIST IR 8259A core baseline (capabilities needed from CPS-connected devices), and compiling an "assurance-ready" CPS deployment documentation set aligned to the NIST CPS facets/aspects vocabulary.

Practical Component: ***(30 hours)***

1. NIST facets/aspects mapping for a CPS scenario + constraints and stakeholders.
2. Specify inputs/outputs/state for 3 CPS components (sensor abstraction, controller abstraction, actuator abstraction) and define their interaction contract.
3. Model controller as FSM: Build a state machine for the controller logic.
4. Create a simulation of periodic sensing + control actions with variable compute/network delays; measure latency and jitter, and detect deadline misses under stress.
5. Implement a small scheduler simulation and compare missed deadlines as utilization increases.
6. Using a sensor time-series, vary sampling rate and show how discretization changes feature stability and decision confidence.
7. Compare edge vs cloud inference (latency, bandwidth, availability, privacy)
8. Implement basic data security aligned to NIST guidance
9. Runtime monitoring and audit logs, define safe-state behavior and trigger conditions
10. Integration test plan + final CPS demo + NIST-aligned system documentation (facets/aspects mapping)

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. W. M. Taha and J. Thunberg, *Cyber-Physical Systems: A Model-Based Approach*, 1st ed., Springer, 2021.
2. R. Alur, *Principles of Cyber-Physical Systems*, 1st ed., MIT Press, 2015.
3. E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed., MIT Press, 2017.

Suggested Readings:

1. R. Rajkumar, D. de Niz, and M. Klein, *Cyber-Physical Systems*, SEI Series, 1st ed., Addison-Wesley, 2017.
2. NIST CPS standards / guidance
    a. NIST SP 1500-201, Framework for Cyber-Physical Systems, Volume 1: Overview, 2017
    b. NIST SP 1500-202, Framework for Cyber-Physical Systems, Volume 2: Working Group Reports, 2017
    c. NIST SP 1500-203, Framework for Cyber-Physical Systems, Volume 3: Timing Annex, 2017
    d. NISTIR 8259A, IoT Device Cybersecurity Capability Core Baseline, 2020
    e. NIST SP 800-82 Rev.3, Guide to Operational Technology (OT) Security, 2023

Detailed Syllabus of Discipline Specific Elective (DSE) courses for B.Tech. (CSE)
SEMESTER VII

WEB ANALYTICS (DSE-5)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Web Analytics	4	3	0	1	NIL

Course Objectives:

1. To introduce students to the concepts, scope, and importance of web analytics and business intelligence.
2. To understand how user interaction data is collected, measured, and analyzed from websites.
3. To familiarize students with qualitative and quantitative web analytics techniques.
4. To enable students to use web analytics tools for decision-making and performance optimization.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain fundamental concepts, terminology, and evolution of web analytics.
2. Analyze web data using qualitative and quantitative analytics methods.
3. Interpret web metrics and key performance indicators (KPIs) for optimization.
4. Apply web analytics tools such as Google Analytics for real-world website analysis.

UNIT-I ***(11 hours)***

Introduction to Web Analytics: Definition and scope of web analytics; process of web analytics; key terminology: site references, keywords and key phrases; building block terms: visit characterization, content characterization, conversion metrics; categories of web analytics: off-site web analytics and on-site web analytics; web analytics platforms; evolution of web analytics; need for web analytics; advantages and limitations of web analytics

UNIT-II ***(11 hours)***

Web Data Collection and Analysis: Data collection techniques; clickstream data: web server logs, web beacons, JavaScript tags, packet sniffing; outcomes data: e-commerce data, lead generation, brand advocacy and support; research data: mindset, organizational structure, and timing; competitive data collection: panel-based measurement, ISP-based measurement, and search engine data.

UNIT-III ***(11 hours)***

Web Metrics and Optimization: Web analytics fundamentals; capturing data using web logs and JavaScript tags; data serving and data capture; type and size of web data; integration and innovation; selecting optimal web analytics tools; clickstream data quality; identifying unique page definitions; cookies and link coding issues; Common web metrics; Introduction to KPIs.

UNIT-IV ***(12 hours)***

Advanced Web Analytics and Tools: Web analytics 2.0: limitations of web analytics 1.0 gya and evolution to analytics 2.0; competitive intelligence analysis: CI data sources, toolbar data, panel data, ISP data, search engine data, hybrid data; website traffic analysis: long-term traffic trends and competitive site analysis; Google Analytics: introduction, working, AdWords integration, benchmarking, traffic categories (organic and paid).

Practical Component: ***(30 hours)***

1. Installation and configuration of web analytics tools.
2. Collection and analysis of website traffic data.
3. Measurement of key web metrics and KPIs.
4. Website performance analysis using Google Analytics.
5. Case studies on website optimization and business intelligence.

List of Experiments:

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. B. Clifton, *Advanced Web Metrics with Google Analytics*, 2nd ed., Wiley Publishing, 2010.
2. A. Kaushik, *Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity*, 1st ed., Wiley Publishing, 2009.
3. J. Sterne, *Web Metrics: Proven Methods for Measuring Web Site Success*, 1st ed., John Wiley & Sons, 2002.

Suggested Readings:

1. A. Kaushik, *Web Analytics: An Hour a Day*, 1st ed., Sybex, 2007.
2. J. Burby, S. Atchison and J. Sterne, *Actionable Web Analytics: Using Data to Make Smart Business Decisions*, 1st ed. Sybex, 2007.

WIRELESS COMMUNICATION (DSE-5)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Wireless Communication	4	3	0	1	Computer Networks

Course Objectives:

1. To understand key principles and challenges of wireless communication systems.
2. To explore radio wave propagation effects and channel modeling.
3. To study modulation, multiple access methods, and performance analysis in wireless environments.
4. To familiarize students with modern wireless standards and system architectures.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain the basic concepts and design principles of wireless communication systems.
2. Analyze wireless radio propagation and channel impairments.
3. Evaluate various modulation and multiple access schemes used in wireless systems.
4. Demonstrate understanding of cellular system architecture and modern wireless technologies.

UNIT-I ***(11 hours)***

Introduction and Cellular Fundamentals: Introduction to wireless communications; evolution of mobile communications; wireless communication systems and applications; cellular concept and frequency reuse; cell design fundamentals; structure of hexagonal cells; channel assignment and reuse pattern; handoff strategies; interference and system capacity; improving coverage and capacity.

UNIT-II ***(11 hours)***

Radio Propagation and Channel Models: Radio propagation mechanisms; free-space propagation; path loss models; large-scale vs small-scale fading; multipath propagation; Doppler effects; delay spread and coherence bandwidth; statistical channel models; shadowing and fading distributions; introduction to channel modeling.

UNIT-III ***(11 hours)***

Modulation and Multiple Access Techniques: Analog and digital modulation techniques for wireless communication: BPSK, QPSK, GMSK; spread spectrum techniques; introduction to OFDM; diversity and equalization concepts; multiple access techniques: FDMA, TDMA, CDMA, OFDMA; comparative analysis of access methods; overview of link level performance.

UNIT-IV ***(12 hours)***

Wireless Systems and Standards: Cellular system architecture; GSM system overview; introduction to 3G & 4G (WCDMA, LTE) principles; wireless LAN standards (IEEE 802.11 family); Bluetooth &

ZigBee fundamentals; introduction to 5G features; emerging paradigms (MIMO, mmWave concepts) and future directions.

Practical Component: ***(30 hours)***

1. Study of wireless system design parameters and link budget calculation.
2. Simulation of path loss and fading models.
3. Implementation of digital modulation techniques.
4. Analysis of multiple access schemes through simulation.
5. Study and configuration of wireless LANs (WiFi) in practical environments.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed., Pearson, 2010.
2. S. Haykin and M. Moher, *Modern Wireless Communications*, 1st ed., Pearson, 2011.
3. W. C. Y. Lee, *Mobile Communications Engineering: Theory and Applications*, 2nd ed., McGraw-Hill, 1997.

Suggested Readings:

1. A. Goldsmith, *Wireless Communications*, 1st ed., Cambridge University Press, 2012.
2. J. Schiller, *Mobile Communications*, 2nd ed., Pearson, 2008.

ENTERPRISE APPLICATION PROGRAMMING (DSE-6)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Enterprise Application Programming	4	3	0	1	Computer Networks

Course Objectives:

1. To introduce fundamentals of Enterprise Java Programming, concepts of program development using beans.
2. To familiarize students with the concept of multi threading
3. To impart knowledge about socket programming and datagrams
4. To familiarize students with the use of Servelets.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain the basic concepts and design principles of Java enterprise programming.
2. Analyze collections used for Java applications Programming as per the use case.
3. Implement multithreading and socket programming.
4. Demonstrate understanding of networking and server side, client side scripting.

UNIT-I ***(12 Hours)***

***Collections*:** Collection Interfaces, Concrete Collections, Collections Framework. Multithreading : Creating and running thread, Multiple thread synchronization, Thread communication, Thread group, Thread priorities, Daemon Thread, Life Cycle of Thread.

UNIT-II ***(15 Hours)***

Fundamentals in Networking: Sockets in Java - Internet Addressing - DNS – Ipv4, IPv6 - URL class - TCP/IP and Datagram. The interfaces and classes for networking: Interfaces and classes of java.net package; InetAddress class : IP address scope - Host name resolution - Methods of InetAddress class; Program to look up the IP addresses for a hostname - Factory methods - Creating and using Sockets : Socket class - constructors and methods of Socket class. Creating TCP servers &clients: TCP/IP server sockets - Constructors and methods of ServerSocket class - Program to create a TCP/IP server and client. Handling URL: URL class - constructors and methods of URL class URLConnection class - fields of URLConnection class - methods of URLConnection class. Working with Datagrams: DatagramPacket - Constructors for DatagramPacket class - Methods of DatagramPacket class - creating Datagram server and client.

UNIT-III ***(10 Hours)***

***JDBC Package*:** JDBC – JDBC versus ODBC – Types of JDBC drivers – Connection – Statement – PreparedStatement.ResultSet :Fields of ResultSet – Methods of ResultSet – Executing a query - ResultSetMetaData – DatabaseMetaData. Datatypes in JDBC : Basic datatypes in JDBC – Advanced datatypes in JDBC – fields of Statement – methods of Statement – CallableStatement Interface – BatchUpdates

UNIT-IV ***(08 Hours)***

***Servlets* :** Using Servlets - Servlet Package - Servlet lifecycle - init() method - service() method , doGet() method, doPost() method and destroy() method . Classes and interfaces of Servlet: Servlet - GenericServlet - ServletConfig - ServletContext - ServletException - ServletInputStream - ServletOutputStream - ServletRequest – ServletResponse. Classes and interfaces of HttpServlet: HttpServlet - HttpServletRequest - HttpServletResponse - Reading HTML form data from Servlets - Response Headers - Response Redirection. Handling Servlets : Servlet Chaining - HttpUtils - Database access with JDBC inside servlet. State and Session management : Cookies - HttpSession - Server Side includes - Request forwarding – RequestDispatcher.

Practical Component: ***(30 Hours)***

1. Study of collections in Java.
2. Use of database connectivity for real time scenarios.
3. Implementation of socket programming
4. Implementation of multiple networking packages.
5. Use of multithreading for multitasking and implementation of javabeans.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. S. Holzner, *Java 2 Programming Black Book*, Dreamtech Press, 2005.
2. Herbert Schildt, Java: The Complete Reference, McGraw Hill Education, Ninth Edition
3. J. O'Neil, *JavaBeans Programming from the Ground Up*, Osborne/ McGraw-Hill TMGH, New Delhi, 1998.

Suggested Readings:

1. K. Sierra and B. Bates, *Head First EJB*, O'Reilly Media, 2003.
2. A. Goncalves, *Beginning Java EE 6 Platform with GlassFish 3: From Novice to Professional*, Apress, 2009.

SOFTWARE DEFINED NETWORKING (DSE-6)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Software Defined Networking	4	3	0	1	Computer Networks

Course Objectives:

1. To understand the fundamentals and evolution of Software Defined Networking (SDN).
2. To study the architectural principles and design of SDN systems.
3. To analyze SDN controllers, protocols, and network programmability.
4. To explore real-world deployment scenarios and performance challenges of SDN.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the need for Software Defined Networking and its advantages over traditional networks.
2. Explain SDN architecture, planes, and control mechanisms.
3. Analyze SDN protocols and controller platforms.
4. Design and evaluate SDN-based network architectures.
5. Implement basic SDN applications and network configurations.

UNIT-I ***(11 hours)***

Introduction to Software Defined Networking*:* Overview of traditional networking and its limitations; evolution of SDN; SDN definition and characteristics; separation of control and data planes; SDN architecture; advantages and challenges; use cases and applications of SDN.

UNIT-II ***(11 hours)***

SDN Architecture and Interfaces: Network Operating System (NOS); SDN control plane and data plane; SDN architecture models; southbound and northbound interfaces; OpenFlow architecture; flow tables; SDN communication models.

UNIT-III ***(11 hours)***

SDN Protocols and Controllers: SDN protocols and standards; OpenFlow protocol architecture; message types and flow control; software-based vs hardware-based SDN switches; SDN controllers such as POX, NOX, Floodlight, OpenDaylight; SDN controller comparison.

UNIT-IV ***(12 hours)***

SDN Design and Applications: Network programmability; Network Function Virtualization (NFV); SDN-based data centers; SDN in enterprise and cloud environments; SDN for traffic engineering; security considerations in SDN; SDN deployment challenges.

Practical Component: ***(30 hours)***

1. Study of traditional networking versus Software Defined Networking architecture.
2. Installation and configuration of SDN environment using Mininet.
3. Implementation of basic SDN topology using OpenFlow switches.
4. Configuration and testing of SDN controllers (POX / Ryu / Floodlight).
5. Flow rule creation and traffic management using SDN controllers.
6. Performance analysis of SDN-based networks under varying traffic conditions.
7. Implementation of network virtualization using SDN.
8. Case study on SDN deployment in data centers or cloud environments.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. P. Goransson, C. Black, and T. Culver, *Software Defined Networks: A Comprehensive Approach*, Morgan Kaufmann, 2016.
2. T. D. Nadeau and K. Gray, *SDN-Software Defined Networks: An Authoritative Review of Network Programmability Technologies*,O'Reilly Media, 2013.
3. W. Stallings, *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*, 1st edition, Addison-Wesley, 2015.

Suggested Readings:

1. K. Gray and T. D. Nadeau, *Network Function Virtualization*, 1st ed. Morgan Kaufmann, 2016.
2. O. Coker and S. Azodolmolky, *Software Defined Networking with OpenFlow*, Packt Publishing, 2017.

DISTRIBUTED SYSTEMS (DSE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Distributed Systems	4	3	0	1	Operating System

Course Objectives:

1. To introduce the fundamental concepts, models, and challenges of distributed systems.
2. To explain communication, coordination, and synchronization mechanisms used in distributed environments.
3. To study classical algorithms for mutual exclusion, consensus, agreement, and fault handling.
4. To expose students to modern distributed platforms, large-scale systems, and security concerns.
5. To develop the ability to analyze, compare, and reason about distributed system designs.

Course Outcomes:

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

1. Explain the principles, models, and architectural foundations of distributed systems.
2. Apply communication and coordination mechanisms such as message passing, RPC, logical clocks, and snapshots in distributed environments.
3. Analyze and compare distributed algorithms for leader election, mutual exclusion, consensus, deadlock detection, and termination detection.
4. Describe the design and operation of large-scale distributed systems including P2P systems, DHTs, and distributed file systems.
5. Understand reliability, fault tolerance, and basic security mechanisms in distributed systems.

UNIT-I ***(10 hours)***

Foundations and Models: Concepts of distributed systems. Characteristics and Challenges. Case study: the world wide web. System models: Physical model, Architectural model, Fundamental models. Models of communication networks.

UNIT-II ***(12 hours)***

Communication and Coordination: Interprocess communication. Message passing systems. Remote Procedure Call (RPC). Remote Method Invocation (RMI). Sockets and message queues. Publish and Subscribe systems. Clock synchronization: Physical clocks, Logical clocks (Lamport clocks), Vector clocks. Distributed mutual exclusion, Leader election algorithms. Causal ordering of messages, Chandy-Lamport's Global State Recording Algorithm.

UNIT-III ***(12 hours)***

Mutual Exclusion, Consensus, and Agreement: Leader Election: Bully algorithm, Ring-based election. Distributed Mutual Exclusion: Requirements and challenges, Non-token-based approaches: Lamport's algorithm, Ricart-Agrawala algorithm. Quorum-based approaches, Token-based approaches: Suzuki-Kasami algorithm, Consensus and Agreement: Floodset algorithm, Byzantine General Agreement Problem. Deadlock detection. Termination Detection: Distributed termination problem, Dijkstra–Scholten algorithm.

UNIT-IV ***(11 hours)***

Modern Platforms and Case Studies: Self-Stabilization: Concept and motivation. Peer-to-Peer (P2P) Systems. Distributed Hash Tables (DHTs): Consistent hashing, Chord, Pastry. Large-Scale Distributed Systems: Google File System (GFS), Hadoop Distributed File System (HDFS), MapReduce programming model. Distributed Systems Security: Authentication, Secure communication

Practical Component: ***(30 hours)***

1. Exercises on client-server and message-passing models using socket libraries, RPC/RMI frameworks, or messaging systems.
2. Experiments on logical clocks, causal ordering, leader election, and distributed mutual exclusion using simulators or configurable distributed frameworks.
3. Hands-on exploration of consensus, deadlock detection, checkpointing, and termination detection using distributed system simulators.
4. Practical exploration of distributed file systems, MapReduce, P2P/DHTs, and security mechanisms using platforms such as Hadoop and network emulation tools.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. Jean Dollimore, Tim Kindberg, Gordon Blair and George Coulouris.. Distributed Systems: Concepts and Design. Pearson, 2017.
2. Maarten van Steen and Andrew S. Tanenbaum. Distributed Systems. Prentice Hall of India. Third edition.
3. Sukumar Ghosh. Distributed Systems: an Algorithmic Approach. CRC Press 2015. Second edition.

Suggested Readings:

1. Martin Kleppmann. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly. Second edition.
2. Tom White. Hadoop - The Definitive Guide.. O'Reilly. Fourth edition.

FOUNDATIONS OF QUANTUM COMPUTING (DSE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Foundations of Quantum Computing	4	3	0	1	NIL

Course Objectives:

1. Introduce the mathematical and physical foundations of quantum computation.
2. Develop intuition about quantum phenomena such as superposition, entanglement, and measurement.
3. Enable students to understand and design quantum circuits and algorithms.
4. Provide hands-on exposure to quantum programming frameworks.
5. Bridge concepts from linear algebra, probability, and algorithms to quantum computation.
6. Prepare students for advanced study or research in quantum information science.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the fundamental principles of quantum mechanics relevant to computation.
2. Explain how quantum information differs from classical information.
3. Model quantum states, quantum gates, and quantum circuits mathematically.
4. Design and analyze basic quantum algorithms such as Deutsch–Jozsa, Grover's, and Shor's algorithms.
5. Implement simple quantum programs using quantum simulators and real quantum hardware.
6. Critically assess the potential and limitations of quantum computing for real-world problems.

UNIT-I ***(11 hours)***

Introduction to Quantum Computing:Motivation for Quantum Computing; Classical Bits vs Quantum Bits (Qubits); Historical Overview and Applications; Postulates of Quantum Mechanics (computational perspective); State Vectors and Hilbert Spaces; Dirac Notation; Measurement and Probability Amplitudes; Bloch Sphere Representation.

UNIT-II ***(11 hours)***

Mathematical Foundations:Complex Vector Spaces; Inner Products and Norms; Tensor Products and Composite Systems; Multi-Qubit States; Quantum Measurement Theory; Density Matrices (introductory); Mixed vs Pure States; Partial Measurement and Reduced Density Matrices.

UNIT-III ***(11 hours)***

Quantum Gates and Circuits:Single-Qubit Gates (Pauli, Hadamard, Phase, Rotation Gates); Multi-Qubit Gates (CNOT, Toffoli); Universal Gate Sets; Quantum Circuit Model; Circuit Depth and Complexity; Entanglement and Bell States; No-Cloning Theorem; Quantum Teleportation and Superdense Coding.

UNIT-IV ***(12 hours)***

Quantum Algorithms and Applications:Deutsch and Deutsch–Jozsa Algorithms; Grover's Search Algorithm; Shor's Factoring Algorithm (conceptual overview); Quantum Fourier Transform; Complexity Classes (P, NP, BQP – overview); Noise, Decoherence, and Error Sources; Introduction to Quantum Error Correction; Near-Term Quantum Devices and NISQ Era Applications.

Practical Component: ***(30 hours)***

1. Representation and visualization of qubits and multi-qubit states.
2. Simulation of single- and multi-qubit quantum gates.
3. Construction and simulation of quantum circuits.
4. Implementation of basic quantum algorithms using Qiskit/Cirq.
5. Running quantum programs on simulators and cloud-based quantum hardware.
6. Mini-project on a quantum algorithm or application.

List of Experiments

Note: The course instructor will design experiments and mini-projects to complete the practical component of the course.

Essential Readings:

1. N. D. Mermin, *Quantum Computer Science: An Introduction*, Cambridge University Press, 2007.
2. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary ed., Cambridge University Press, 2010.
3. J. Preskill, *Lecture Notes on Quantum Computation*, California Institute of Technology, 2015.

Suggested Readings:

1. E. G. Rieffel and W. H. Polak, *Quantum Computing: A Gentle Introduction*, MIT Press, 2011.
2. Eric R. Johnston, Nic Harrigan, Mercedes Gimeno-Segovia, Programming Quantum Computers: Essential Algorithms and Code Samples, 2019.

Detailed Syllabus of Discipline Specific Core (DSC) courses for B.Tech. (CSE)
SEMESTER VIII

FAIRNESS, BIAS & RESPONSIBLE ETHICS IN AI (DSC-20)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Fairness, Bias & Responsible Ethics in AI	4	3	0	1	Artificial Intelligence & Machine Learning

Course Objectives:

1. Provide comprehensive understanding of fairness, bias, and discrimination in AI systems.
2. Examine mathematical definitions and trade-offs of fairness.
3. Introduce bias detection and mitigation techniques.
4. Study explainable, transparent, and accountable AI frameworks.
5. Explore governance, legal, and societal implications of AI deployment

Course Outcomes:

At the end of this course, students will be able to:

1. Understand foundational ethical theories and their relevance to AI systems.
2. Identify different types of bias in datasets and machine learning models.
3. Apply fairness metrics to evaluate AI systems.
4. Analyze and mitigate bias using algorithmic and data-level techniques.
5. Evaluate AI systems under responsible AI and governance frameworks.

UNIT-I ***(10 hours)***

Foundations of AI Ethics and Fairness: Introduction to AI ethics; Evolution of ethical concerns in AI; Stakeholders in AI systems; Ethical theories - Utilitarianism, Deontology, Virtue Ethics; Concepts of justice and distributive fairness; Sources of bias in AI systems - data bias, sampling bias, measurement bias, algorithmic bias, deployment bias; Individual vs group fairness; Trade-offs between fairness and accuracy.

UNIT-II ***(13 hours)***

Measuring and Mitigating Bias in AI*:* Mathematical foundations of fairness; Confusion matrix and fairness metrics; Demographic parity; Equal opportunity; Equalized odds; Predictive parity; Impossibility theorem in fairness; Bias detection through exploratory data analysis; Pre-processing methods (re-sampling, re-weighting); In-processing techniques (fair regularization, adversarial debiasing); Post-processing corrections; Fairness in classification and regression models.

UNIT-III ***(12 hours)***

Responsible, Explainable and Privacy-Preserving AI*:* Principles of Responsible AI – transparency, accountability, safety, inclusivity; Interpretability vs explainability; Model-agnostic explanation

techniques; SHAP and LIME concepts; Counterfactual explanations; Privacy concerns in AI; Differential privacy basics; Federated learning overview; Risk assessment in AI systems; Auditing AI systems; Ethical AI design frameworks and lifecycle management.

UNIT-IV ***(10 hours)***

Governance, Regulation and Societal Impact of AI: AI governance models and standards; Risk-based AI regulation; Global AI ethics guidelines; Accountability and liability in AI systems; AI in healthcare, criminal justice, finance, education, and agriculture; Human-centered AI and participatory design; Generative AI risks and misinformation; Sustainability and green AI; Case studies of AI failures and responsible redesign strategies.

Practical Component: ***(30 hours)***

1. Bias analysis on real-world datasets (e.g., hiring, loan approval).
2. Implementation of fairness metrics (demographic parity, equalized odds).
3. Comparative study of accuracy vs fairness trade-offs.
4. Application of bias mitigation techniques (pre-, in-, post-processing).
5. Explainable AI experiments using SHAP/LIME tools.
6. Mini-project: Audit and evaluate an AI system for fairness and responsibility.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning: Limitations and Opportunities*, 1st ed., MIT Press, 2023.
2. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson, 2022.
3. L. Floridi, *Ethics of Artificial Intelligence:Principles, Challenges, and Opportunities*, 1st Ed., Oxford University Press, 2019.

Suggested Readings:

1. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed., MIT Press, 2016.
2. C. O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*, 1st ed., Crown Publishing, 2016.
3. IEEE, *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems*, 1st ed., IEEE, 2019.

Detailed Syllabus of Discipline Specific Elective (DSE) courses for B.Tech. (CSE)
SEMESTER VIII

BIO-INSPIRED COMPUTING (DSE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Bio-inspired Computing	4	3	0	1	Analysis and Design of Algorithms

Course Objectives:

1. Introduce the biological motivation behind modern computational intelligence techniques.
2. Provide a strong foundation in evolutionary and swarm-based optimization algorithms.
3. Enable students to understand and implement neuro-inspired and physics-based algorithms.
4. Develop hands-on skills in applying bio-inspired techniques to engineering and data-driven problems.
5. Encourage analytical comparison of bio-inspired approaches with classical optimization techniques.
6. Prepare students for research and advanced applications in intelligent systems and optimization.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the fundamental principles of bio-inspired computing and how biological systems inspire computational models.
2. Analyze and apply evolutionary algorithms such as Genetic Algorithms and Differential Evolution for optimization problems.
3. Implement swarm intelligence techniques including Particle Swarm Optimization and Ant Colony Optimization for real-world problems.
4. Apply neural and neuro-inspired computing concepts for learning and pattern recognition tasks.
5. Design hybrid bio-inspired solutions combining multiple algorithms for complex optimization and decision-making problems.

UNIT-I ***(08 hours)***

Introduction to Bio-Inspired Computing: Definition and scope of bio-inspired computing; Historical evolution and motivation; Biological systems as computational models; Classification of bio-inspired algorithms: evolutionary, swarm-based, neural-inspired, and physics-based algorithms; Fitness functions and optimization landscapes; Performance measures and convergence behavior; Applications of bio-inspired computing.

UNIT-II ***(14 hours)***

Evolutionary Algorithms*:* Introduction to evolutionary computation; Genetic Algorithms (GA): encoding schemes, population initialization, fitness evaluation, selection methods, crossover operators, mutation operators, and elitism; Genetic Programming; Evolution Strategies; Differential Evolution; Constraint handling techniques; Applications of evolutionary algorithms to function optimization, scheduling, and combinatorial problems.

UNIT-III ***(11 hours)***

Swarm Intelligence*:* Swarm intelligence principles and collective behavior; Particle Swarm Optimization (PSO): velocity and position update equations, parameter tuning, convergence characteristics; Ant Colony Optimization (ACO): pheromone model, probabilistic path selection, pheromone update rules; Bee Colony Optimization; Applications of swarm intelligence to routing, clustering, load balancing, and optimization problems.

UNIT-IV ***(12 hours)***

Neural and Hybrid Bio-Inspired Systems: Neuro-inspired computing fundamentals; Artificial Neural Networks as bio-inspired models; Learning rules and adaptation; Hybrid bio-inspired algorithms (evolutionary-neural and swarm-neural systems); Physics- and chemistry-inspired algorithms: Simulated Annealing, Bacterial Foraging Optimization; Case studies in optimization, pattern recognition, and decision support systems; Emerging trends and research challenges in bio-inspired computing.

Practical Component: ***(30 hours)***

1. Implementation of Genetic Algorithms for optimization problems such as function minimization or Travelling Salesman Problem.
2. Implementation of Differential Evolution and comparison with Genetic Algorithms.
3. Simulation of Particle Swarm Optimization for continuous optimization problems.
4. Implementation of Ant Colony Optimization for shortest path or routing problems.
5. Design and training of a simple neural network for classification using bio-inspired learning principles.
6. Implementation of a hybrid bio-inspired algorithm combining swarm intelligence and neural networks.
7. Mini-project applying a bio-inspired algorithm to a real-world optimization problem.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. X.S. Yang, *Nature-Inspired Optimization Algorithms*, 2nd ed., Elsevier, 2021.
2. A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed., Springer, 2015.
3. N. Siddique and H. Adeli, *Nature-Inspired Computing: Physics and Chemistry-Based Algorithms*, 1st ed., Chapman & Hall/CRC, 2017.

Suggested Readings:

1. Y. Xiao, *Bio-Inspired Computing and Networking*, 1st ed., CRC Press / Taylor & Francis, 2011.
2. S. S. Bhoi et al., *Bio-Inspired Neurocomputing*, 1st ed., Springer Nature, 2020.
3. X.-S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, 2nd ed., Wiley, 2010.

ML AND DL FOR 3D DATA (DSE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
ML and DL for 3D Data	4	3	0	1	Artificial Intelligence & Machine Learning

Course Objectives:

1. Provide strong probabilistic and statistical foundations for 3D data learning.
2. Introduce geometric representations and theoretical foundations of 3D shape analysis.
3. Study deep learning architectures tailored for 3D data modalities.
4. Develop practical competence in solving real-world 3D vision problems.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain mathematical foundations underlying 3D data analysis.
2. Apply probabilistic machine learning techniques to 3D shape representations.
3. Design deep neural architectures for voxel, point cloud, and mesh data.
4. Implement 3D object recognition, segmentation, and reconstruction models.
5. Evaluate and compare learning models for 3D computer vision applications.

UNIT-I ***(12 hours)***

Mathematical and Probabilistic Foundations for 3D Learning: Probability theory and distributions; Bayesian inference and decision theory; linear models for regression and classification; mixture models and EM algorithm; dimensionality reduction including PCA; graphical models; density estimation and latent variable models as discussed in Pattern Recognition and Machine Learning; geometric transformations, coordinate systems, projective geometry and camera models; image formation and multi-view geometry fundamentals as presented in Computer Vision: Models, Learning, and Inference.

UNIT-II ***(10 hours)***

Fundamentals of 3D Shape Analysis and Geometric Representations*:* 3D shape representations including point sets, surfaces, meshes and volumetric grids; differential geometry concepts; curvature and surface descriptors; shape similarity measures; feature extraction and invariant representations; registration and alignment techniques; statistical shape models; spectral methods for shape analysis; theoretical foundations and applications of 3D shape matching and retrieval as covered in 3D Shape Analysis: Fundamentals, Theory, and Applications.

UNIT-III ***(13 hours)***

Deep Learning Architectures for 3D Data*:* Introduction to deep learning for structured and unstructured 3D data; 3D convolutional neural networks for volumetric representations; multi-view CNN approaches; point-based neural networks; hierarchical feature learning for point clouds; graph neural networks for

mesh processing; loss functions and optimization strategies for 3D tasks; regularization and generalization in deep models; self-supervised and transfer learning approaches for 3D data as discussed in Deep Learning for 3D Data.

UNIT-IV ***(10 hours)***

Advanced Topics in 3D Understanding and Applications: 3D object classification and segmentation; scene understanding and semantic labeling; 3D object detection and LiDAR-based perception; shape completion and reconstruction; generative models for 3D shapes; probabilistic graphical models for spatial reasoning; performance evaluation metrics for 3D tasks; integration of probabilistic learning with deep 3D architectures; applications in robotics, autonomous systems, medical imaging and augmented reality.

Practical Component: ***(30 hours)***

1. Implementation of probabilistic models(for eg PCA) for 3D feature datasets.
2. 3D data preprocessing including mesh handling, point cloud normalization and voxelization.
3. Feature-based 3D shape classification using classical machine learning methods.
4. Implementation of 3D convolutional neural networks.
5. Point cloud classification and segmentation using deep learning frameworks.
6. 3D object segmentation and reconstruction experiments.
7. Mini-project involving development of an end-to-end 3D object recognition, detection, or reconstruction system.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., Springer, 2006.
2. S. J. D. Prince, *Computer Vision: Models, Learning, and Inference*, 2nd ed., Cambridge University Press, 2012.
3. H. Shao and X. Wang, *Deep Learning for 3D Data*, 1st ed., 2023.

Suggested Readings:

1. D. W. Jacobs and R. Basri, *3D Shape Analysis: Fundamentals, Theory, and Applications*, 1st ed., 2017.
2. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, 2004.
3. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed., MIT Press, 2016.

SUSTAINABLE AND GREEN COMPUTING (DSE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Sustainable and Green Computing	4	3	0	1	Cloud Computing

Course Objectives:

1. To introduce the concepts and goals of sustainable and green computing.
2. To understand the environmental impact of computing systems and IT infrastructure.
3. To study energy-efficient techniques in software, hardware, data centers, and cloud computing.
4. To analyze the role of IT in achieving environmental sustainability.

Course Outcomes:

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

1. Analyze the environmental impact of IT and propose sustainable solutions.
2. Develop software and systems that prioritize energy efficiency and sustainability.
3. Design green data centers and cloud architectures that support sustainable practices.
4. Understand and contribute to the achievement of Sustainable Development Goals through technology.
5. Engage in responsible and ethical computing practices that promote digital sustainability.

UNIT-I ***(12 hours)***

Introduction to Green and Sustainable Computing: Introduction to Green Computing, Environmental Concerns and the importance of Sustainable Development, Environmental Impact of Information Technology, Carbon Footprint of Computing Devices, Green IT concepts: Green IT 1.0 and Green IT 2.0 (overview), Life Cycle of Computing Devices, Three R's of Sustainability: Reduce, Reuse, Recycle, Green Hardware Concepts: Energy-Efficient Devices, Reuse, Recycling, and Disposal, Green IT standards and eco-labeling, Challenges and Future Trends in Sustainable Computing.

UNIT-II ***(10 hours)***

Sustainable Software and Energy-Efficient Systems: Introduction to sustainable software development, Energy consumption in software and applications, Software sustainability attributes: efficiency, usability, maintainability, reusability, Measuring Energy Consumption in Computing Systems, Power-aware Operating Systems, Energy-Efficient Processors, Memory, and Storage, Introduction to Virtualization and its role in optimizing resources, Role of algorithms in energy-efficient computing.

UNIT-III ***(11 hours)***

Green Data Centers and Cloud Computing: Introduction to Data Centers, Energy Challenges in Modern Data Centers, Green Data Center Concepts and Design, Cooling techniques and thermal management, Green Cloud Computing and Environmental Sustainability, Features of Clouds Enabling Green Computing, Energy Efficiency of Cloud Computing, Green Cloud Architecture, Role of virtualization in

green cloud computing, Benefits and challenges of green cloud computing, Enterprise Green IT strategy, Organizational considerations for Green IT Implementation.

UNIT-IV ***(12 hours)***

Sustainable Development Goals, Socio-Ethical Aspects: Introduction to Sustainability and the United Nations Sustainable Development Goals (SDGs), Role of Green and Sustainable Computing in achieving SDGs, Sustainable Cities and Communities: Smart Cities and Green ICT solutions, Intelligent Transportation Systems and Energy-aware Urban Systems, Sustainable Hardware Lifecycle, E-waste Management and Circular Economy in IT, Socio-cultural impact of Green IT, Responsible user practices and digital sustainability, Ethics, privacy, and security in sustainable information systems, Green IT governance, policies, and standards, Emerging technologies for sustainable computing.

Practical Component: ***(30 hours)***

1. To study the concept of Green Computing and analyze the environmental impact of computing systems.
2. To study the Three R's of Sustainability (Reduce, Reuse, Recycle) in IT systems.
3. To analyze power management features in operating systems.
4. To evaluate the energy efficiency of different algorithms.
5. To study the concept of Virtualization for resource optimization.
6. To analyze the energy consumption in data centers.
7. To study the architecture of Green Cloud Computing.
8. To study the role of green computing in sustainable urban systems.
9. To design a Green IT strategy for an organization.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. S. Murugesan and G. R. Gangadharan, *Harnessing Green IT: Principles and Practices*, 1st ed., Wiley-IEEE Computer Society, 2012.
2. J. Lamb, *The Greening of IT*, 1st ed., IBM Press, 2009.
3. J. Harris, *Green Computing and Green IT: Best Practices on Regulations and Industry*, 1st ed., Emero, 2008.
4. W. Leonhard and K. Murray, *Green Home Computing for Dummies*, 1st ed., For Dummies, 2009.

Suggested Readings:

1. S. K. Das, *Cloud Computing*, 1st ed., Dominant Pub and Dist, 2015.
2. P. Pattnaik and M. Kabat, *Fundamentals of Cloud Computing*, 1st ed., Vikas Publishing House, 2014.

COMPUTING FOR SMART CITIES AND INTELLIGENT INFRASTRUCTURE (DSE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Computing for Smart Cities and Intelligent Infrastructure	4	3	0	1	NIL

Course Hours:L-03, T-00, P-02

Course Objectives:

1. To understand the multi-layered architecture of Smart City solutions.
2. To explore the role of edge and cloud computing in managing urban data.
3. To analyze intelligent infrastructure systems like smart grids, ITS, and waste management.
4. To address the security, privacy, and sustainability challenges in hyper-connected cities.

Course Outcomes:

By the end of this course, the student will be able to:

1. Design a communication framework for large-scale sensor networks in urban settings.
2. Apply machine learning algorithms to predict urban trends (traffic, energy demand).
3. Evaluate the efficiency of intelligent transport and energy systems.
4. Develop prototypes for smart urban services using IoT and Cloud platforms.

UNIT-I ***(10 hours)***

Smart City Framework and Connectivity: The Smart City Ecosystem: Definition, dimensions and the 4-layer architecture. Communication Protocols: Short-range vs. Long-range. 5G and Beyond: The role of ultra-low latency in vehicle-to-everything (V2X) communication. Urban Sensing: Mobile crowdsensing, fixed sensor networks, and satellite imagery integration.

UNIT-II ***(12 hours)***

Intelligent Infrastructure Systems: Smart Energy & Grids: Demand-response management, integration of renewable energy, and smart metering. Intelligent Transportation Systems: Real-time traffic monitoring, adaptive signal control, and autonomous shuttle integration. Smart Water & Waste: Automated leak detection, quality monitoring, and sensor-based waste collection routing. Structural Health Monitoring: Using IoT to monitor the integrity of bridges, buildings, and tunnels.

UNIT-III ***(13 hours)***

Data Analytics and Urban Intelligence: Edge vs. Cloud Computing: Distributed processing for real-time urban decision-making. Urban Big Data: Challenges of volume, velocity, and variety in city-scale data. Predictive Modeling: Time-series analysis for air quality forecasting and energy consumption prediction. Digital Twins: Creating virtual replicas of physical city infrastructure for simulation and stress testing.

UNIT-IV ***(10 hours)***

Governance, Security, and Sustainability: Smart Governance: Open data portals, e-participation, and blockchain for transparent city records. Cybersecurity in Infrastructure: Protecting critical assets from hacking. Anonymization techniques for citizen data. Circular economy models and the role of computing in reducing the carbon footprint of cities.

Practical Component : ***(30 hours)***

1. Sensor Integration: Interface an air quality sensor (MQ series) with an ESP32 and push data to a cloud dashboard.
2. Traffic Simulation: Use SUMO to model traffic flow and test adaptive signal timing.
3. Develop a Python script to predict household energy consumption using a public Smart Meter dataset and LSTM networks.
4. Design a system using ultrasonic sensors and an MQTT broker to display real-time parking availability.
5. Configure a LoRa gateway and node to measure signal strength (RSSI) across different urban obstacles.
6. Use a platform to visualize real-time sensor data overlaid on a 3D building model.
7. Implement a simplified smart contract for a community-based "Participatory Budgeting" system.

List of Experiments

Note: Course instructor may add/delete/update new experiments in addition to the above Suggested practical exercise

Essential Readings:

1. Smart Cities: Foundations, Principles, and Applications — by H. Song, R. Srinivasan, T. Sookoor, S. Jeschke
2. Internet of Things: A Hands-On Approach — by A. Bahga & V. Madisetti
3. Smart Cities: Big Data, Civic Hackers, and the Quest for a New Utopia — by Anthony M. Townsend

Suggested Readings:

1. Shalli, et al. (2022). IoT and WSN based Smart Cities: A Machine Learning Perspective. Springer.
2. Pardo, T. A., & Nam, T. (2011). Conceptualizing smart city with dimensions of technology, people, and institutions.

HUMAN CENTERED COMPUTING (DSE-10)

CREDIT DISTRIBUTION, ELIGIBILITY, AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Human Centered Computing	4	3	0	1	NIL

Course Objectives:

1. Develop a strong foundation in Human-Centered Design (HCD) for building usable, useful, and inclusive computing systems.
2. Train students to perform user research, translate insights into requirements, and iteratively design through prototyping.
3. Build practical capability in interaction design for modern computing contexts (web/mobile, data-intensive products, AI-assisted interfaces).
4. Introduce accessibility, human factors, ethics, and privacy so students can design responsibly for diverse users and real-world constraints.
5. Enable students to evaluate and improve designs using standard Human Computer Interaction (HCI) / (User Experience) UX evaluation methods, metrics, and evidence-based iteration.

Course Outcomes:

At the end of this course, students will be able to:

1. Plan and conduct user research and synthesize findings into personas, scenarios, task flows, and measurable usability goals.
2. Produce interaction designs and prototypes (low and high fidelity) for web/mobile and other interactive computing experiences.
3. Apply accessibility and inclusive design principles and identify/remediate common usability barriers.
4. Design user experiences for data-driven and AI-assisted features with transparency, user control, and calibrated trust.
5. Conduct usability evaluations (expert and user-based), analyze results, and propose prioritized design improvements.
6. Incorporate privacy-by-design and ethical considerations into interface requirements and product decisions.

UNIT-I ***(10 hours)***

Human Factors as System Constraints and Interaction as Computation: Human-Centered Computing (HCC) scope as socio-technical systems engineering, cognition and attention limits, mental models and conceptual models, human error types and recovery design, responsiveness and perceived performance, interaction as event-driven computation, modeling interaction with state machines and invariants, multimodal inputs (touch/voice/gesture) and sensor/actuator feedback loops in interactive devices, accessibility engineering fundamentals (semantic UI, keyboard-first flows, screen-reader compatibility),

inclusive design constraints for diverse abilities and contexts, ethics primer for user impact and manipulation resistance (dark patterns, consent-as-interaction).

UNIT-II ***(15 hours)***

Engineering Interaction Architecture for Modern Computing Systems: User-work modeling as engineering artifacts (personas, hierarchical task inventory, task sequences, flow models), information architecture as graphs (navigation, findability, search), interaction architecture from task-flow → screen-flow → API contract, component-based UI architecture, design systems and reusable components (tokens, theming, consistency), client-server interaction design (API affordances, pagination/infinite scroll, optimistic UI, offline-first, caching), performance engineering (render cost, network waterfalls, critical path, caching/CDN considerations), collaboration and social computing primitives (shared state, concurrency, consistency, presence/awareness cues, moderation primitives), ubiquitous/IoT and XR interaction considerations (context, safety boundaries, sensor uncertainty, ergonomic constraints), domain-driven HCC examples (health, education, finance tech) - constraints, compliance, and real-world workflows.

UNIT-III ***(10 hours)***

Data-Driven HCC-Instrumentation, Experiments, Visualization, and Decision Support: Telemetry and event schema design, product analytics and behavioral measurement limits, metrics taxonomy (UX quality metrics, guardrails, business metrics) and target setting, funnel/cohort/retention analysis, causal reasoning intuition for product decisions, online controlled experiments (A/B tests), experimentation platform architecture (feature flags, staged rollouts, canaries), observability for human outcomes (error budgets for UX, rage clicks, abandonment, crash loops), dashboards and visualization principles for trustworthy decision support (cognitive load, uncertainty display, misleading charts).

UNIT-IV ***(10 hours)***

Trustworthy Human-Centered Computing-Usable Security, Privacy, Fairness, Human-AI Interaction, and Governance: Usable security engineering (auth flows, permissions, least-privilege UX, phishing-resistant and abuse-resistant design), safety and misuse-case analysis (harassment, fraud, coercion, unsafe automation), privacy engineering in systems and products (consent, notice design, minimization, retention, sensitive data handling), trustworthy AI characteristics and risk framing for AI-enabled features (valid/reliable, safe, secure/resilient, accountable/transparent, explainable / interpretable, privacy-enhanced, fair), human-AI interaction patterns (set expectations, feedback, control, graceful failure), explanation UX and uncertainty communication by user role, feedback loops and human-in-the-loop operations (labeling UX, escalation), recommender/ranking interaction risks (manipulation, filter bubbles, over-personalization), fairness concepts and tradeoffs.

Practical Component: ***(30 hours)***

1. Choose a computing product problem (web/mobile/data/AI-based), define target users, context-of-use, and measurable success criteria
2. Conduct user research (minimum: 1 interview/observation + 1 survey), synthesize findings into key insights and pain points
3. Create personas, scenarios, user journeys, and task flows, write clear UX requirements and acceptance criteria
4. Build low-fidelity prototypes (paper/wireframes), run quick feedback sessions, iterate
5. Build a clickable/high-fidelity prototype (Figma or a simple front-end), ensure consistency using a mini design system
6. Define error states and recovery flows, design micro-interactions and feedback for key tasks
7. Accessibility audit and improvement plan (contrast, typography, keyboard navigation, focus order, labels), implement fixes in the prototype

8. Design a data-centric screen (dashboard/analytics/search) with usable filters, progressive disclosure, and clear hierarchy
9. Design an AI-assisted feature (recommendation/autocomplete/copilot-style help) with transparency, user control, and fallback behavior
10. Implement a human-in-the-loop flow for the AI feature (confirm/override/feedback capture), define the feedback data structure and logging
11. Run a usability test (5-8 users), compute SUS + task metrics, analyze results and prioritize fixes
12. Add privacy UX elements (consent, settings, data controls, retention notice), justify choices using NIST Privacy Framework thinking, final demo + report

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. Y. Rogers, H. Sharp, and J. Preece, *Interaction Design: Beyond Human-Computer Interaction*, 6th ed., Wiley, 2023.
2. R. Hartson and P. S. Pyla, *The UX Book: Agile UX Design for a Quality User Experience*, 3rd ed., Morgan Kaufmann / Elsevier, 2025.
3. W3C, *Web Content Accessibility Guidelines (WCAG) 2.2 and ARIA Authoring Practices Guide (APG)*, World Wide Web Consortium, latest edition.
4. R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*, Cambridge University Press, 2020.

Suggested Readings:

1. Microsoft, *Human-AI Interaction Guidelines*, Microsoft Research, latest edition.
2. Google PAIR Team, *People + AI Guidebook and PAIR Design Patterns*, Google Research, latest edition.
3. A. Mathur, M. Kshirsagar, and J. Mayer, "Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites," *Proceedings of the ACM on Human-Computer Interaction*, ACM, 2019.
4. NIST standards / guidance
    a. NIST, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, National Institute of Standards and Technology, 2023.
    b. NIST, *NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management*, Version 1.0, National Institute of Standards and Technology, 2020.

QUANTUM COMPUTING AND APPLICATIONS (DSE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Quantum Computing and Applications	4	3	0	1	Foundations of Quantum Computing

Course Objectives:

1. Build advanced theoretical and practical skills in quantum computing.
2. Introduce cutting-edge quantum algorithms and application domains.
3. Enable students to work with noisy intermediate-scale quantum (NISQ) devices.
4. Provide hands-on experience with quantum programming and optimization.
5. Encourage interdisciplinary applications across AI, cryptography, and physical sciences.
6. Prepare students for research or industry roles in quantum technologies.

Course Outcomes:

At the end of this course, students will be able to:

1. Analyze and design advanced quantum algorithms for computational problems.
2. Apply quantum machine learning techniques to data-driven tasks.
3. Understand quantum error correction and fault-tolerant computation.
4. Evaluate quantum hardware architectures and noise models.
5. Implement hybrid quantum–classical algorithms on real quantum devices.

UNIT-I ***(11 hours)***

Advanced Quantum Algorithms*:* Amplitude Amplification and Variants; Advanced Applications of Grover's Algorithm; Quantum Phase Estimation; Quantum Walks; Variational Quantum Algorithms (VQE, QAOA); Hamiltonian Simulation; Linear Systems Algorithms (HHL – conceptual understanding).

UNIT-II ***(11 hours)***

Quantum Error Correction and Noise*:* Sources of Noise and Decoherence; Noise Models (Depolarizing, Dephasing, Amplitude Damping); Quantum Error Correction Codes (Bit-flip, Phase-flip, Shor Code, Steane Code); Fault-Tolerant Quantum Computation; Threshold Theorem; Error Mitigation Techniques for NISQ Devices.

UNIT-III ***(11 hours)***

Quantum Machine Learning and Optimization*:* Quantum Data Encoding Techniques; Quantum Kernels; Variational Quantum Classifiers; Quantum Neural Networks; Hybrid Quantum–Classical Learning; Quantum Optimization Algorithms; Applications in Pattern Recognition, Chemistry, and Finance.

UNIT-IV ***(12 hours)***

Applications and Emerging Directions*:* Quantum Cryptography and Post-Quantum Security; Quantum Simulation in Chemistry and Materials Science; Quantum Computing for Optimization and Logistics; Quantum Sensing and Metrology (overview); Cloud-Based Quantum Computing Platforms; Ethical, Societal, and Security Implications of Quantum Technologies.

Practical Component: ***(30 hours)***

1. Implementation of variational quantum algorithms (VQE, QAOA).
2. Simulation and mitigation of noise in quantum circuits.
3. Quantum machine learning models using hybrid frameworks.
4. Execution of quantum programs on real quantum hardware.
5. Performance benchmarking of quantum vs classical algorithms.
6. Capstone mini-project focused on a real-world quantum application.

List of Experiments

Note: The course instructor will design experiments and mini-projects to complete the practical component of the course.

Essential Readings:

1. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary ed., Cambridge University Press, 2010.
2. M. Schuld and F. Petruccione, *Machine Learning with Quantum Computers*, 1st ed., Springer, 2018.
3. J. Preskill, *Quantum Computing in the NISQ Era and Beyond*, Quantum, vol. 2, 2018.

Suggested Readings:

1. A. Montanaro, *Quantum Algorithms: An Overview*, npj Quantum Information, vol. 2, 2016.
2. Jack D. Hidary, *Quantum Computing: An Applied Approach*, Springer, 2nd Edition, 2019.

Department of Computer Science and Engineering
Faculty of Technology
University of Delhi

List of Discipline Specific Elective (DSE)/ Generic Elective (GE) courses offered for Minors / Specializations by the Department of CSE in Fourth Year

1. Minor in CSE (Offered to ECE and EE)

a. GE-7: Introduction to Data Analytics & Data Visualization
b. GE-8: Fundamentals of Algorithms
c. GE-9: Low Level Software Systems
d. GE-9: OOP Concepts
e. GE-10: Fundamentals of Machine Learning
f. GE-10: Fundamentals of Cloud Computing

2. Minor/Specialization in Artificial Intelligence & Machine Learning (Offered to CSE, ECE, and EE)

a. DSE-6 / GE-7: Edge Computing
b. DSE-7 / GE-8: Reinforcement Learning
c. DSE-9 / GE-9: Generative AI
d. DSE-9 / GE-9: Scientific Machine Learning
e. DSE-10 / GE-10: Automated Machine Learning
f. DSE-10 / GE-10: Federated Learning

3. Minor/Specialization in Data Science (Offered to CSE, ECE, and EE)

a. DSE-6 / GE-7: Deep Learning Models and Applications
b. DSE-7 / GE-8: Big Data Analytics
c. DSE-9 / GE-9: Visual Computing
d. DSE-9 / GE-9: Applied AI
e. DSE-10 / GE-10: Applied Generative AI for Data Science
f. DSE-10 / GE-10: Data Privacy, Security & Regulatory Compliance

4. Minor/Specialization in Software Engineering (Offered to CSE, ECE, and EE)

a. DSE-6 / GE-7: Software Quality Assurance and Metrics
b. DSE-7 / GE-8: Software Maintenance and Evolution
c. DSE-9 / GE-9: Program Analysis and Verification
d. DSE-9 / GE-9: Empirical Software Engineering
e. DSE-10 / GE-10: Secure Software Engineering
f. DSE-10 / GE-10: Build, Release and Deployment Engineering

5. Minor/Specialization in Blockchain and Cybersecurity (Offered to CSE, ECE, and EE)

a. DSE-6 / GE-7: Smart Contract and DApps

b. DSE-7 / GE-8: Digital Currencies and Blockchain
c. DSE-9 / GE-9: Cyber Law and Digital Forensics
d. DSE-9 / GE-9: Cryptanalysis & Security Models
e. DSE-10 / GE-10: AI in Cybersecurity
f. DSE-10 / GE-10: Quantum Cryptography

6. Minor/Specialization in Augmented Reality / Virtual Reality (Offered to CSE, ECE, and EE)

a. DSE-6 / GE-7: 3D Animation Design
b. DSE-7 / GE-8: Multiuser and Social AR/VR
c. DSE-9 / GE-9: AR/VR Application Development
d. DSE-9 / GE-9: Computer Vision for AR/VR
e. DSE-10 / GE-10: AR/VR Hardware Systems: Concepts and Architectures
f. DSE-10 / GE-10: Ethics, Privacy, and Safety in XR Systems

Detailed Syllabus of Generic Elective (GE) courses offered for Minors / Specializations by Department of CSE in SEMESTER VII

INTRODUCTION TO DATA ANALYTICS & DATA VISUALIZATION (GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Introduction to Data Analytics & Data Visualization	4	3	0	1	NIL

Course Objectives:

1. Introduce the principles and workflow of data analytics.
2. Develop statistical thinking for analyzing engineering data.
3. Teach data cleaning, transformation, and exploratory analysis techniques.
4. Provide practical skills in data visualization tools and libraries.
5. Demonstrate applications of analytics in electrical and electronics engineering domains.

Course Outcomes:

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

1. Understand fundamental concepts of data analytics and its role in engineering applications.
2. Apply statistical and exploratory techniques to analyze engineering datasets.
3. Perform data preprocessing and transformation for real-world data.
4. Create effective data visualizations for technical interpretation and decision-making.
5. Analyze domain-specific datasets from ECE and EE applications.

UNIT-I ***(09 hours)***

Foundations of Data Analytics: Introduction to data analytics and its importance in engineering; types of data including structured and unstructured data; data analytics lifecycle; descriptive, diagnostic, predictive and prescriptive analytics; data sources in real world such as sensor data, power system measurements and communication signals; basic probability concepts; measures of central tendency and dispersion; introduction to data-driven decision making.

UNIT-II ***(14 hours)***

Data Preprocessing and Exploratory Data Analysis: Data collection and data formats; handling missing data and outliers; data cleaning and normalization; data transformation and feature engineering; correlation analysis; covariance and regression basics; exploratory data analysis techniques; visualization for exploration; time-series data analysis for power and signal data; introduction to simple statistical modeling.

UNIT-III ***(10 hours)***

Data Visualization Techniques: Principles of effective visualization; types of plots including line plots, bar charts, histograms, scatter plots and box plots; visualization of multivariate data; time-series visualization; heatmaps and correlation matrices; dashboards and reporting; storytelling with data; visualization best practices for engineering reports; introduction to visualization tools.

UNIT-IV ***(12 hours)***

Analytics Applications in Real World: Signal data analysis; power system load analysis; fault detection using data analytics; predictive maintenance basics; basic machine learning concepts for engineers including supervised and unsupervised learning; clustering and classification overview; case studies involving communication networks, smart grids and IoT systems; ethical considerations in data analytics.

Practical Component: ***(30 hours)***

1. Data collection and preprocessing using Python or equivalent tools.
2. Implementation of statistical measures and exploratory data analysis.
3. Visualization of engineering datasets using plotting libraries.
4. Time-series analysis of power or sensor data.
5. Mini-project involving analysis and visualization of a real-world dataset.

List of Experiments

Note: The course instructor will design experiments and mini-projects to complete the practical component of the course.

Essential Readings:

1. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann, 2011.
2. D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 7th ed., Wiley, 2018.
3. W. McKinney, *Python for Data Analysis*, 3rd ed., O'Reilly Media, 2022.

Suggested Readings:

1. C. N. Knaflic, *Storytelling with Data: A Data Visualization Guide for Business Professionals*, 1st ed., Wiley, 2015.
2. F. Provost and T. Fawcett, *Data Science for Business*, 1st ed., O'Reilly Media, 2013.

EDGE COMPUTING (DSE-6 / GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Edge Computing	4	3	0	1	NIL

Course Objectives:

1. Introduce edge computing as a distributed computing paradigm spanning the cloud–edge continuum, driven by latency, bandwidth, privacy, and reliability needs.
2. Build skills to design edge system architectures, choose appropriate platforms (containers, orchestration, MEC), and reason about deployment constraints.
3. Develop competency in edge resource management (placement, offloading, caching, scheduling) and performance evaluation.
4. Enable students to build data/streaming and AI-at-edge solutions, including model deployment and monitoring under edge constraints.
5. Introduce security, privacy, and operational concerns for edge deployments, including standards-based edge / MEC architectural thinking.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain edge computing concepts, motivations, and architectures, and differentiate edge, fog, cloudlets, and MEC-style deployments.
2. Design an end-to-end edge solution including edge nodes, connectivity, data pipeline, and cloud integration, and justify architectural choices using performance and locality constraints.
3. Implement and deploy edge services using containers and basic orchestration patterns, and evaluate latency/throughput/resource usage.
4. Build an edge analytics/AI workflow (stream ingestion to inference to feedback/monitoring) and apply suitable optimization patterns for edge constraints.
5. Identify major edge security/privacy risks and apply baseline mitigation strategies and operational practices in an edge prototype.

UNIT-I ***(10 hours)***

Edge Computing Fundamentals and the Cloud–Edge Continuum: edge computing motivations (low latency, bandwidth reduction, data locality, resilience, privacy), edge vs cloud vs fog vs cloudlets vs MEC, edge deployment scenarios (industrial IoT, smart cities, AR/VR, video analytics, connected vehicles) and workload characteristics, edge reference architectures (device-gateway-edge node–regional cloud-central cloud), data locality and context awareness, service models (edge as platform, edge as cache, edge analytics), edge application decomposition into microservices, introduction to edge performance metrics (end-to-end latency, tail latency, jitter, availability, energy), overview of canonical edge vision and drivers from foundational literature.

UNIT-II ***(12 hours)***

Edge Platforms, Virtualization, and Orchestration: edge hardware and system constraints (heterogeneity, intermittent connectivity, limited power), virtualization choices (VMs vs containers, lightweight isolation), container-based deployment and packaging, edge orchestration concepts (service discovery, placement, scaling, rolling updates), state management at the edge (stateless vs stateful services, local storage, synchronization), networking for edge (service meshes at a high level, NAT constraints, secure tunnels), mobility and session continuity concepts, MEC framework and reference architecture notions (MEC system and host, platform services, reference points), integration of edge with 4G/5G ecosystems and SDN/NFV ideas at a conceptual level.

UNIT-III ***(12 hours)***

Edge Resource Management, Offloading, and Data Pipelines: computation offloading motivations and models (partial vs full offload, device–edge–cloud split), placement and scheduling basics (latency-aware placement, resource-aware scheduling, multi-tenancy), edge caching and content distribution concepts, stream processing at the edge (windowing, event-time vs processing-time intuition, backpressure basics), data management and consistency tradeoffs (eventual consistency, synchronization, conflict handling), QoS/QoE thinking (SLA-like targets, admission control), benchmarking and evaluation methodology (workload profiling, measurement pitfalls, tail latency focus), overview of research directions and system challenges highlighted in surveys.

UNIT-IV ***(11 hours)***

Edge Intelligence, Security, Privacy, and Operational Reliability: edge analytics and edge AI (on-device/near-device inference, pipeline design for sensor/video/text workloads), model deployment patterns (batch vs streaming inference, model versioning at a high level), optimization themes (model compression/quantization concepts, hardware acceleration awareness, latency budgeting), federated/distributed learning intuition and privacy-aware learning concepts, trust and transparency in edge AI decisions, security threat landscape (insecure edge nodes, rogue gateways, data leakage, model theft/poisoning at a high level), security controls (identity, authentication, secure communication, isolation, secure updates, monitoring and logging), reliability patterns (graceful degradation, offline operation, failover, observability), reading and discussing modern edge+AI research directions from recent survey/tutorial literature.

Practical Component: ***(30 Hours)***

1. Edge lab setup: configure an edge node (Raspberry Pi / x86 mini-PC / VM), basic Linux hardening, networking, time sync, benchmarking tools
2. Containerization: package a microservice (REST/gRPC) into Docker, run locally, measure latency and resource usage
3. Edge messaging pipeline: MQTT/Kafka-lite/NATS-style pub-sub (any one), sensor/event simulation, logging and replay
4. Edge gateway pattern: device → gateway → edge node, local buffering during connectivity loss, replay on reconnection
5. Simple MEC-style decomposition exercise: split an application into edge vs cloud components, justify split using latency/bandwidth constraints
6. Orchestration mini-lab: deploy services on a lightweight edge orchestrator (e.g., k3s or docker-compose), rolling update and basic service discovery
7. Offloading experiment: implement and compare local vs edge vs cloud execution for a CPU-bound task, measure end-to-end and tail latency
8. Edge caching experiment: cache hot content/results at edge, compare hit rate, latency, bandwidth savings
9. Edge stream analytics: implement windowed aggregation/anomaly rule over streaming data, handle backpressure via batching/throttling

10. Edge AI inference: deploy a lightweight model for classification/anomaly detection, measure inference latency and accuracy tradeoff
11. Edge AI robustness: simulate drift/noise and log model confidence/false alarms, propose mitigation (thresholding, retraining trigger, fallback)
12. Final mini-project: build an end-to-end edge application (video analytics / industrial telemetry / smart campus / health monitoring simulator), include deployment scripts, evaluation report, and a short discussion of security and reliability controls.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. R. Buyya and S. N. Srirama (eds.), *Fog and Edge Computing: Principles and Paradigms*, 1st ed., Wiley, 2019.
2. J. Taheri and S. Deng (eds.), *Edge Computing: Models, Technologies and Applications*, 1st ed., Institution of Engineering and Technology (IET), 2020.
3. L. Xu and W. Shi (eds.), *Edge Computing: Systems and Applications*, 1st ed., Wiley / Wiley-IEEE Press, 2025.

Suggested Readings:

1. ETSI, *Multi-access Edge Computing (MEC): Framework and Reference Architecture (ETSI GS MEC 03)*, latest ed., European Telecommunications Standards Institute (ETSI).
2. M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, IEEE Computer Society, vol. 50, no. 1, pp. 30–39, 2017.
3. W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge Computing: A Survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, Elsevier, 2019.
4. H. Hua et al., "Edge Computing with Artificial Intelligence: A Machine Learning Perspective," ACM Computing Surveys, vol. 55, no. 9, ACM, 2023.

DEEP LEARNING MODELS AND APPLICATIONS (DSE-6 / GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Deep Learning Models and Applications	4	3	0	1	Fundamentals of Data Analytics, Artificial Intelligence and Machine Learning

Course Objectives:

1. Introduce foundations and mathematical principles of deep learning.
2. Understand and implement major deep learning architectures.
3. Apply deep learning models to vision, language, and sequence problems.
4. Analyze real-world applications and challenges of deep learning systems.

Course Outcomes:

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

1. Explain core concepts and learning mechanisms of deep neural networks.
2. Design and implement deep learning models using standard architectures.
3. Apply deep learning techniques to real-world vision and language problems.
4. Evaluate performance and address challenges in deep learning applications

UNIT-I ***(12 hours)***

Foundations of Deep Learning: Introduction to Deep Learning, ML vs DL, Biological inspiration, Perceptron, MLP, Activation functions, Loss functions, Gradient descent, Backpropagation, Regularization and optimization challenges.

UNIT-II ***(12 hours)***

Deep Learning Architectures: CNN architecture, convolution, pooling, padding, stride, CNN variants, RNN, vanishing gradient, LSTM, GRU, Attention mechanism, Transformer overview.

UNIT-III ***(12 hours)***

Deep Learning Models for Vision and Language: Computer vision tasks, AlexNet, VGG, ResNet (overview), NLP with deep learning, word embeddings, sequence modeling, Vision Transformers, transfer learning.

UNIT-IV ***(09 hours)***

Applications and Advanced Topics in Deep Learning: Applications in speech, vision, healthcare, recommender systems, autonomous systems, Autoencoders, VAEs, GANs, evaluation metrics, ethical issues, case studies.

Practical Component: ***(30 Hours)***

1. Implementation of basic neural networks using Python
2. CNN-based image classification

3. Sequence modeling using RNN / LSTM
4. Transfer learning using pre-trained models
5. Mini-project on deep learning application

List of Experiments

Note: The course instructor will design experiments and mini-projects to complete the practical component of the course.

Essential Readings:

1. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed., MIT Press, 2016.
2. F. Chollet, *Deep Learning with Python*, 2nd ed., Manning Publications, 2021.
3. A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed., O'Reilly Media, 2022.

Suggested Readings:

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., Springer, 2006.
2. S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., Pearson Education, 2009.

SOFTWARE QUALITY ASSURANCE AND METRICS (DSE-6 / GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Software Quality Assurance and Metrics	4	3	1	0	Software Engineering, Object Oriented Software Engineering

Course Objectives:

1. To introduce concepts of software quality and quality assurance.
2. To develop a quantitative understanding of software measurement and metrics.
3. To apply product, process, and maintenance metrics for quality evaluation.
4. To analyze object-oriented software quality using standard metrics.
5. To expose students to software quality estimation tools and computer-aided quality engineering.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain software quality assurance principles and standards.
2. Apply software quality metrics for product, process, and maintenance evaluation.
3. Analyze object-oriented software quality using standard OO metrics.
4. Estimate and analyze software quality using quantitative models and tools.
5. Apply metrics methodology and CAQE concepts in software quality evaluation

UNIT-I ***(10 hours)***

Software Quality Concepts and Quality Assurance: Introduction to software quality; definitions and perspectives of quality; software quality attributes; software quality assurance (SQA): objectives, activities, and role in software development; verification and validation; quality planning; quality assurance vs quality control; introduction to software quality standards (ISO 9001, ISO/IEC 25010).

UNIT-II ***(12 hours)***

Software Quality Metrics and Measurement: Software measurement concepts; metrics and measurement theory; software quality metrics; product quality metrics; process quality metrics; size metrics (LOC, Function Point Analysis); complexity metrics (Cyclomatic Complexity); productivity metrics; metrics for software maintenance; limitations and misuse of metrics.

UNIT-III ***(13 hours)***

Object-Oriented Metrics and Quality Estimation*:* Object-oriented software engineering principles; need for object-oriented metrics; Chidamber and Kemerer (CK) metrics suite; MOOD metrics; coupling, cohesion, inheritance, and polymorphism metrics; software quality metrics methodology; Goal–Question–Metric (GQM) approach; defect density and reliability metrics; software quality estimation models.

UNIT-IV ***(10 hours)***

Software Quality Tools and Computer-Aided Quality Engineering: Software quality estimation tools: objectives and scope; desirable features of software quality estimation tools; static and dynamic analysis tools; defect tracking and metrics tools; comparative study of existing tools; Computer-Aided Quality Engineering (CAQE) concepts; design techniques for CAQE; integration of CAQE with the software development life cycle; case studies.

Tutorial Component: ***(15 hours)***

1. Identification and analysis of software quality attributes for a given system.
2. Numerical problems on LOC, Function Point Analysis, and Cyclomatic Complexity.
3. Computation and interpretation of product and process quality metrics.
4. Analysis of maintenance metrics using defect and effort data.
5. Evaluation of object-oriented designs using CK and MOOD metrics.
6. Designing a metrics plan using the Goal–Question–Metric (GQM) approach.
7. Case study on software quality estimation using metrics and models.
8. Comparative study of software quality estimation tools.
9. Conceptual design of a Computer-Aided Quality Engineering (CAQE) framework.

List of tutorial tasks

Note: The course instructor will design tasks to complete the tutorial component of the course.

Essential Readings:

1. R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed., McGraw-Hill Education, 2019.
2. Y. Singh and R. Malhotra, *Object-Oriented Software Engineering*, 1st Edition, PHI Learning Private Limited, 2010.
3. N. E. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*, 3rd Edition, CRC Press, 2014.
4. S. H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd Edition, Addison-Wesley, 2002.

Suggested Readings:

1. I. Sommerville, *Software Engineering*, 10th ed., Pearson, 2015.
2. ISO/IEC 25010, *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE)*, 2011.
3. ISO 9001, *Quality Management Systems – Requirements*, 2015.

SMART CONTRACTS AND DAPPS (DSE-6 / GE-7)

CREDIT DISTRIBUTION, ELIGIBILITY, AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Smart Contracts and DApps	4	3	0	1	Blockchain Essentials

Course Objectives:

1. To understand the fundamentals of blockchain and decentralized systems.
2. To introduce the concepts of smart contracts, decentralized applications, and blockchain platforms.
3. To develop the ability to design and deploy smart contracts using Solidity.
4. To understand the architecture, tools, and security aspects of decentralized applications.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain the working principles of blockchain and Ethereum platforms.
2. Develop smart contracts using Solidity programming language.
3. Deploy and interact with smart contracts in a blockchain environment.
4. Design and implement decentralized applications using modern blockchain tools.
5. Analyze security issues and best practices in smart contract development.

UNIT-I ***(11 hours)***

Introduction to Blockchain and Ethereum: Introduction to Blockchain Technology; Evolution of Distributed Ledger Systems; Centralized vs Decentralized Architectures; Blockchain Components; Cryptographic Foundations; Ethereum Overview; Ethereum Architecture; Ethereum Virtual Machine (EVM); Accounts, Transactions, and Gas; Use Cases of Blockchain and Ethereum.

UNIT-II ***(11 hours)***

Smart Contracts and Ethereum Environment: Smart Contracts: Definition and Characteristics; Smart Contract Lifecycle; Account Types and Transactions; Gas and Gas Optimization; Ethereum Development Environment; Tools and Frameworks; Web3 and Ethereum APIs; Writing, Compiling, Testing, and Deploying Smart Contracts.

UNIT-III ***(11 hours)***

Solidity Programming: Introduction to Solidity Language; Structure of a Solidity Program; Data Types and Variables; Control Structures; Functions and Modifiers; Visibility Specifiers; Events and Logging; Error Handling (require, assert, revert); Memory, Storage, and Calldata; Contract Creation and Interaction.

UNIT-IV ***(12 hours)***

Decentralized Applications (DApps): Decentralized Application Architecture; DApp Components; Front-end and Back-end Integration; Smart Contract Interaction using Web3.js and Ethers.js; Wallet Integration (MetaMask); DApp Design Patterns; Use Cases such as Voting, Token Systems, and Auctions; Security Considerations and Best Practices.

Practical Component: ***(30 hours)***

1. Installation and configuration of blockchain development environment (Ganache, Remix, MetaMask).
2. Creation and management of cryptocurrency wallets and accounts.
3. Writing, compiling, and deploying smart contracts using Solidity.
4. Implementing token standards (ERC-20 / ERC-721) and executing transactions.
5. Developing and testing decentralized applications (DApps) using Web3.js.
6. Demonstrating smart contract security features and common vulnerabilities.
7. Case studies on real-world blockchain applications and deployments.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. A. M. Antonopoulos and G. Wood, *Mastering Ethereum*, 2nd ed., O'Reilly Media, 2018.
2. G. Shrivastava, D.-N. Le, and K. Sharma, *Cryptocurrencies and Blockchain Technology Applications*, 1st ed., Wiley & Sons, 2020.
3. Josh Thompson, *Blockchain: The Blockchain for Beginners - Guide to Blockchain Technology and Programming*, 1st ed., Independently Published, 2017.

Suggested Readings:

1. K. Solorio, R. Kanna, and D. H. Hoover, *Hands-On Smart Contract Development with Solidity and Ethereum: From Fundamentals to Deployment*, O'Reilly Media, 2019.
2. A. M. Antonopoulos and D. A. Harding, *Mastering Bitcoin: Programming the Open Blockchain*, 3rd ed. O'Reilly Media, 2023.

3D ANIMATION DESIGN (DSE-6 / GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
3D Animation Design	4	3	0	1	NIL

Course Objectives:

1. To introduce the standard 3D production pipeline (Pre-production, Production, Post-production).
2. To master the technical aspects of polygonal modeling, UV mapping, and digital sculpting.
3. To apply the "12 Principles of Animation" within a digital 3D environment.
4. To understand the physics of lighting, shading, and rendering engines.

Course Outcomes:

By the end of this course, the student will be able to:

1. Construct complex 3D assets using industry-standard modeling techniques.
2. Apply realistic textures and materials using UV unwrapping and PBR workflows.
3. Animate characters and objects with believable physics and personality.
4. Render high-quality sequences using global illumination and ray-tracing techniques.

UNIT-I ***(15 hours)***

Modeling and Geometry: Introduction to 3D Interface: Navigating the viewport, coordinate systems (Local vs. World), and basic primitives.;Polygonal Modeling: Extrude, Bevel, Bridge, and Boolean operations; maintaining "All-Quads" topology for animation.;NURBS vs. Polygons: Understanding mathematical surfaces vs. mesh-based surfaces; Digital Sculpting: High-poly detailing and the concept of multiresolution modeling.

UNIT-II ***(11 hours)***

Surfacing and Texturing: UV Unwrapping: Projecting 3D surfaces onto 2D planes; managing texture seams and stretching; Shading Networks: Introduction to PBR (Physically Based Rendering) materials; Albedo, Roughness, and Normal maps; Procedural Texturing: Creating materials using nodes and noise functions without external images; Baking Maps: Transferring high-poly detail to low-poly models via Normal and Ambient Occlusion maps.

UNIT-III ***(11 hours)***

Rigging and Animation: Character Rigging: Skeleton creation (Joints/Bones), Forward Kinematics (FK), and Inverse Kinematics (IK); Skinning: Weight painting to define how the mesh deforms during movement; The 12 Principles: Implementing Squash and Stretch, Anticipation, Staging, and Follow-through in 3D; Graph Editor: Fine-tuning animation curves, interpolation types (Linear vs. Bézier), and keyframe timing.

UNIT-IV ***(8 hours)***

Lighting, Dynamics, and Rendering: Lighting Techniques: Three-point lighting (Key, Fill, Rim), HDRI environments, and Global Illumination.;Particle Systems: Simulating fire, smoke, and basic fluid dynamics; Rendering Engines: Differences between Rasterizers (Real-time) and Path Tracers (Arnold/Cycles); Compositing: Basic multi-pass rendering (Render Layers) and final video output.

Practical Component: ***(30 hours)***

List of Experiments:

1. Hard Surface Modeling: Design and model a detailed mechanical object (e.g., a vintage camera or a sci-fi drone).
2. Character Sculpting: Create a basic humanoid or creature bust using digital sculpting tools.
3. UV & Texturing Lab: Perform a "clean" UV unwrap of a complex object and apply hand-painted or PBR textures.
4. The Bouncing Ball: Animate a ball with different weights (Bowling ball vs. Ping-pong ball) to master Squash and Stretch.
5. Character Walk Cycle: Create a seamless 24-frame looping walk cycle for a bipedal character.
6. Interior Lighting: Setup an architectural visualization scene and render it using Physical Sky and Area lights.

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. R. Williams, *The Animator's Survival Kit*, Faber & Faber, 2001.
2. D. Derakhshani, *Introducing Autodesk Maya*, Sybex, 2014.
3. A. Chopine, *3D Art Essentials: The Fundamentals of 3D Modeling, Texturing, and Animation*, Focal Press, 2011.

Suggested Readings:

1. W. Vaughan, *Digital Modeling*, New Riders, 2011.
2. A. Beane *3D Animation Essentials: New Directions for Evaluation,,* John Wiley, 2012.

FUNDAMENTALS OF ALGORITHMS (GE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Fundamentals of Algorithms	4	3	0	1	Data Structures

Course Objectives:

1. Introduce core principles of algorithms in an application-oriented manner.
2. Develop the ability to estimate computational efficiency using asymptotic notations.
3. Familiarize students with commonly used algorithms in engineering analysis.
4. Enable practical problem solving using structured algorithmic techniques.
5. Strengthen logical reasoning and analytical thinking skills across disciplines.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand basic concepts of algorithms and their role in engineering problem solving.
2. Analyze algorithm efficiency using simple time and space complexity measures.
3. Apply searching and sorting techniques to engineering datasets.
4. Use fundamental algorithmic approaches to solve optimization and graph-based engineering problems.

UNIT-I ***(10 hours)***

Introduction to Algorithms and Complexity: Definition and characteristics of algorithms; algorithm design steps; flowcharts and pseudocode; performance analysis; time and space complexity; best case, worst case and average case analysis; asymptotic notations including Big O, Big Theta and Big Omega; practical examples from signal processing, power systems and numerical computation.

UNIT-II ***(11 hours)***

Searching, Sorting and Recursion: Linear search and binary search; introduction to sorting concepts; bubble sort, selection sort and insertion sort; overview of merge sort and quick sort (conceptual understanding); comparison of sorting algorithms; basic idea of recursion and recursive algorithms; simple applications in engineering data processing; complexity comparison of searching and sorting techniques.

UNIT-III ***(12 hours)***

Algorithmic Strategies for Problem Solving*:* Introduction to divide and conquer with practical examples; greedy method fundamentals; applications such as activity selection and minimum spanning tree (conceptual); introduction to dynamic programming with simple examples such as Fibonacci and shortest path in grid; comparison of greedy and dynamic programming approaches; engineering optimization problems modeled algorithmically.

UNIT-IV ***(11 hours)***

Graphs and Engineering Applications: Introduction to graphs and their representation; traversal techniques including BFS and DFS; shortest path algorithms including Dijkstra's algorithm; minimum spanning trees and their applications in network design; topological sorting (conceptual); applications of graph algorithms in communication networks, power grids, transportation and circuit analysis; basic introduction to algorithmic problem modeling in engineering systems.

Practical Component: ***(30 hours)***

1. Implementation of searching and sorting algorithms.
2. Complexity comparison using runtime measurement.
3. BFS and DFS implementation on engineering network graphs.
4. Dijkstra's algorithm for shortest path in power/communication networks.
5. Simple dynamic programming problems.
6. Mini-project: Modeling an engineeing system as a graph problem.

List of Experiments

Note: The course instructor will design experiments and mini-projects to complete the practical component of the course

Essential Readings:

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed., MIT Press, 2009.
2. J. Kleinberg and É. Tardos, *Algorithm Design*, 1st ed., Pearson Education, 2005.
3. E. Horowitz, S. Sahni, and S. Rajasekaran, *Fundamentals of Computer Algorithms*, 2nd ed., Universities Press, 2008.

Suggested Readings:

1. A. Levitin, *Introduction to the Design and Analysis of Algorithms*, 3rd ed., Pearson Education, 2012.
2. R. K. Shukla, *Analysis and Design of Algorithms: A Beginner's Approach*, Wiley India Pvt. Ltd., 2015.

REINFORCEMENT LEARNING (DSE-6 / GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Reinforcement Learning	4	3	0	1	NIL

Course Hours: L: 03, T: 00, P: 02

Course Objectives:

1. Introduce the mathematical and computational foundations of reinforcement learning for sequential decision-making problems.
2. Develop the ability to formulate computing tasks as MDPs/bandits and define reward signals, states, and actions precisely.
3. Enable students to implement and compare core RL algorithms for prediction and control in tabular settings.
4. Train students to build deep RL agents with function approximation, stable training pipelines, and reproducible experiments.
5. Develop competence in policy optimization and actor-critic methods for discrete and continuous control.
6. Explain how RL principles extend to preference-based learning and RLHF-style alignment used in instruction-following LLM pipelines (pretraining vs alignment fine-tuning).

Course Outcomes:

At the end of this course, students will be able to:

1. Explain RL formulations (MDP, return, policy, value functions) and derive Bellman equations for prediction and control.
2. Implement and evaluate tabular RL algorithms (DP, Monte Carlo, TD, SARSA, Q-learning) on standard toy environments.
3. Implement deep value-based RL with function approximation and evaluate stability using appropriate metrics and seeds.
4. Design and implement policy-gradient and actor-critic methods (including PPO-style objectives) for benchmark control tasks.
5. Apply preference-based learning concepts to build a simple reward model and use it to optimize a policy (RLHF-style loop at small scale).
6. Evaluate RL systems for robustness and risk (reward design, safety constraints, dataset shift in offline settings) and propose mitigations aligned with AI risk frameworks.

UNIT-I ***(11 hours)***

Foundations of Reinforcement Learning and MDP Modeling: agent–environment interaction and reward hypothesis, episodic vs continuing tasks and returns, Markov property and Markov decision processes (MDPs), policies and policy improvement idea, state-value and action-value functions, Bellman expectation and optimality equations, dynamic programming methods (policy evaluation, policy iteration,

value iteration), Monte Carlo prediction and control, temporal-difference learning (TD(0), n-step intuition), exploration vs exploitation and ε-greedy strategies, on-policy vs off-policy learning, SARSA and Q-learning, contextual bandits as simplified RL.

UNIT-II ***(12 hours)***

Function Approximation and Deep Value-Based Reinforcement Learning: motivation for generalization in large state spaces, feature representations and linear function approximation, gradient-descent view of prediction, instability sources with bootstrapping and off-policy learning, experience replay and target networks, deep Q-networks (DQN) and loss design, overestimation bias and mitigation (Double Q intuition), architectural enhancements (dueling/value-advantage decomposition), exploration in deep RL (noise and schedules), practical evaluation methodology (learning curves, variance across random seeds, sample efficiency vs wall-clock efficiency), debugging signals and failure patterns (divergence, reward scaling, value blow-up).

UNIT-III ***(11 hours)***

Policy Optimization and Actor-Critic Methods: motivation for direct policy optimization, stochastic policies and log-likelihood gradients, REINFORCE with baselines and variance reduction, advantage functions and actor-critic architectures, entropy regularization for exploration, trust-region motivation and clipped surrogate objectives (PPO-style), continuous-action control and reparameterization intuition, off-policy actor–critic overview with replay buffers, practical training issues (normalization, stability, sensitivity to hyperparameters, monitoring and early stopping).

UNIT-IV ***(11 hours)***

RL Systems in Practice, Offline/Model-Based RL, and RLHF-Style Preference Learning: model-based RL overview (learning dynamics models, planning vs learning trade-offs), offline RL fundamentals, multi-agent RL basics (self-play, cooperation vs competition, non-stationarity), reward design and reward hacking, safe RL concepts, preference-based RL (pairwise comparisons, learning a reward model from preferences), RLHF-style alignment loop for instruction-following LLMs as an application of policy optimization and reward modeling (pretraining by next-token prediction followed by supervised instruction tuning and RLHF-style fine-tuning), KL-regularized optimization intuition for keeping a policy close to a reference model, reproducibility practices for RL systems (seeding, versioning, reporting).

Practical Component: ***(30 hours)***

1. Environment and tooling setup: Python RL stack, Gymnasium-style environment interaction, seeding for reproducibility, logging and experiment folder structure.
2. Implement multi-armed bandits (ε-greedy, UCB-style, Thompson-style) and compare regret on multiple reward distributions.
3. Implement a Gridworld MDP and solve it via policy evaluation + value iteration; visualize value functions and greedy policies.
4. Implement Monte Carlo prediction/control on an episodic environment; analyze variance and sensitivity to episode count.
5. Implement TD prediction (TD(0), n-step extension) and compare learning curves against Monte Carlo under identical seeds.
6. Implement SARSA (on-policy) and Q-learning (off-policy) for control; study the impact of exploration schedules and step sizes.
7. Build an evaluation harness: multiple seeds, confidence intervals/summary statistics, checkpointing, and standardized reporting plots.
8. Implement value-function approximation (linear or small neural network) and demonstrate at least one instability mode and a mitigation.

9. Implement DQN with replay buffer + target network on CartPole (or equivalent); add one improvement (e.g., Double DQN or dueling head).
10. Implement an actor–critic baseline (A2C-style) and then a PPO-style clipped objective; compare stability and sample efficiency.
11. Preference learning lab (RLHF-style at small scale): collect pairwise preferences over short trajectories or text outputs, train a reward model, then optimize a policy against the learned reward with a constraint to stay close to a reference policy (e.g., KL-penalty), and evaluate reward hacking risks.
12. Mini-project: choose a computing/system problem (game, scheduling, resource allocation, simulated control, or small text-policy task), define state/action/reward clearly, implement two algorithm families (value-based vs policy/actor–critic or RLHF-style preference loop), run ablations, and submit a reproducible report with code and logs.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.
2. M. Lapan, *Deep Reinforcement Learning Hands-On*, 2nd ed., Packt Publishing, 2020.
3. A. Zai and B. Brown, *Deep Reinforcement Learning in Action*, Manning Publications, 2020.
4. N. Lambert, *Reinforcement Learning from Human Feedback*, online book / PDF, 2026.

Suggested Readings:

1. Canonical papers / surveys
2. V. Mnih et al., “Human-Level Control through Deep Reinforcement Learning,” *Nature*, 2015.
3. J. Schulman et al., “Proximal Policy Optimization Algorithms,” arXiv:1707.06347, 2017.
4. P. Christiano et al., “Deep Reinforcement Learning from Human Preferences,” *Advances in Neural Information Processing Systems (NIPS)*, 2017.
5. L. Ouyang et al., “Training Language Models to Follow Instructions with Human Feedback,” 2022.
6. M. Alshiekh et al., “Safe Reinforcement Learning via Shielding,” *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
7. S. Gu et al., “A Review of Safe Reinforcement Learning: Methods, Theories, and Applications,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
8. S. Levine et al., “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems,” 2020.

BIG DATA ANALYTICS (DSE-7 / GE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Big Data Analytics	4	3	0	1	NIL

Course Objectives:

1. Understand fundamentals and ecosystem of Big Data.
2. Implement distributed storage and processing using Hadoop and MapReduce.
3. Explore NoSQL and data warehousing tools.
4. Apply analytics frameworks to extract insights from big data.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain Big Data concepts and tools.
2. Develop programs using Hadoop and MapReduce.
3. Use MongoDB, Hive, and Pig for big data processing.
4. Apply Spark for large-scale data analytics.

UNIT-I ***(12 Hours)***

Introduction to Big Data and Analytics: Classification of Data, Characteristics of Big Data, Evolution of Big Data, Traditional BI vs Big Data, Hadoop Ecosystem, Types of Analytics, NoSQL overview.

UNIT-II ***(12 Hours)***

Hadoop and MapReduce Framework: Hadoop architecture, HDFS, YARN, MapReduce programming model, Mapper, Reducer, Combiner, Partitioner, Searching and Sorting, Compression.

UNIT-III ***(12 Hours)***

NoSQL and Data Processing Tools: NoSQL databases, MongoDB architecture and query language, Hive architecture, HQL, File formats, User Defined Functions.

UNIT-IV ***(9 Hours)***

Big Data Analytics Frameworks and Applications: Pig architecture and Pig Latin, Pig vs Hive, Apache Spark architecture, Data analysis with Spark, Text and Web Analytics, PageRank, Case studies.

Practical Component: **(30 Hours)**

1. Installation and configuration of Hadoop ecosystem components and execution of HDFS file operations.
2. Development of MapReduce programs such as Word Count and Matrix Multiplication.

3. Implementation of CRUD (Create, Read, Update, Delete) and aggregation operations using MongoDB.
4. Writing and execution of data processing scripts using Hive and Pig.
5. Implementation of data analytics tasks using Apache Spark.
6. Mini-project involving the design and development of a Big Data Analytics application.

List of Experiments

Note: The course instructor will design experiments and mini-projects to complete the practical component of the course

Essential Readings:

1. S. Acharya and S. Chellappan, *Big Data and Analytics*, 1st ed., Wiley India, 2015.
2. R. Kamal and P. Saxena, *Big Data Analytics*, 1st ed., McGraw-Hill Education, 2017.
3. T. White, *Hadoop: The Definitive Guide*, 4th ed., O'Reilly Media, 2015.

Suggested Readings:

1. T. Erl, W. Khattak, and P. Buhler, *Big Data Fundamentals: Concepts, Drivers & Techniques*, 1st ed., Pearson Education, 2016.
2. A. Shook and D. Miner, *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems*, 1st ed., O'Reilly Media, 2015.

SOFTWARE MAINTENANCE AND EVOLUTION (DSE-7/GE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Software Maintenance and Evolution	4	3	1	0	Software Engineering, Object Oriented Software Engineering

Course Objectives:

1. To introduce the concepts and importance of software maintenance and evolution.
2. To understand different types of software maintenance and their impact on cost and quality.
3. To study software evolution laws, models, and processes.
4. To analyze program comprehension, reengineering, and refactoring techniques.
5. To expose students to modern challenges in maintaining large and long-lived software systems.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain the role of software maintenance in the software life cycle.
2. Classify and analyze different types of software maintenance activities.
3. Apply software evolution models and laws to real software systems.
4. Analyze legacy systems and apply reengineering and refactoring techniques.
5. Evaluate maintenance strategies using quantitative and qualitative measures.

UNIT-I ***(09 hours)***

Introduction to Software Maintenance: Software maintenance: definition and need; maintenance as a phase of software life cycle; types of maintenance: corrective, adaptive, perfective, and preventive; cost of maintenance; maintenance process models; maintenance planning and management; maintenance standards and documentation.

UNIT-II ***(12 hours)***

Program Comprehension and Maintenance Techniques: Program understanding and reverse engineering; legacy systems and maintenance challenges; code analysis techniques; documentation recovery; impact analysis; change management; configuration management in maintenance; tools and environments for software maintenance.

UNIT-III ***(11 hours)***

Software Evolution: Software evolution concepts; Lehman's laws of software evolution; evolution processes and models; software aging and decay; evolution of requirements and architecture; managing continuous evolution; metrics for software evolution and maintainability.

UNIT-IV ***(13 hours)***

Reengineering, Refactoring, and Modern Maintenance Issues: Software reengineering: objectives and process; reverse engineering and forward engineering; code restructuring and refactoring techniques;

design recovery; migration and modernization of legacy systems; maintenance of object-oriented systems; maintenance challenges in agile, DevOps, and open-source environments; case studies.

Tutorial Component: ***(15 hours)***

1. Identification and classification of maintenance activities for a given software system.
2. Case study on cost and effort distribution in software maintenance.
3. Program comprehension exercise on a medium-sized codebase.
4. Impact analysis for proposed changes in an existing system.
5. Study and discussion of Lehman's laws using real software examples.
6. Analysis of software evolution metrics and maintainability indices.
7. Refactoring exercise: identifying code smells and proposing improvements.
8. Legacy system reengineering case study.
9. Critical review of maintenance challenges in agile or DevOps-based projects.

List of tutorial tasks

Note: The course instructor will design tasks to complete the tutorial component of the course.

Essential Readings:

1. T. M. Pigoski, *Practical Software Maintenance: Best Practices for Managing Your Software Investment*, IEEE Computer Society Press, 2001.
2. M. M. Lehman and J. F. Ramil, *Software Evolution*, Springer, 2006.
3. M. Fowler, *Refactoring: Improving the Design of Existing Code*, 2nd ed., Addison-Wesley, 2018.

Suggested Readings:

1. R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed., McGraw-Hill Education, 2019.
2. Y. Singh and K. K. Aggarwal, *Software Engineering*, 2nd ed., New Age International Publishers, 2018.

DIGITAL CURRENCIES AND BLOCKCHAIN (DSE-7/GE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Digital Currencies and Blockchain	4	3	0	1	Blockchain Essentials, Cryptography Essentials

Course Objectives:

1. To introduce concepts and the evolution of cryptocurrencies and blockchain systems.
2. To provide in-depth knowledge of blockchain architecture and cryptographic primitives.
3. To develop skills in cryptocurrency design, scripting, and transaction validation.
4. To understand consensus mechanisms and decentralized trust models.
5. To explore real-world applications, challenges, and future directions of blockchain technology.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the evolution, principles, and significance of digital currencies and blockchain technologies.
2. Analyze blockchain architectures, consensus mechanisms, and cryptographic foundations.
3. Apply cryptographic and scripting techniques to design cryptocurrency systems.
4. Evaluate security, privacy, and trust issues in blockchain-based platforms.
5. Design and assess blockchain-based solutions for real-world applications.

UNIT-I ***(11 hours)***

Fundamentals of Digital Currency and Blockchain*:* Cryptocurrency - origin and evolution, importance and use cases, legal and regulatory aspects, blockchain structure and components, interaction between blockchain and cryptocurrencies, public, private, and consortium blockchains, and hardware and software requirements for blockchain systems.

UNIT-II ***(11 hours)***

Functional Aspects of Cryptocurrency: Bitcoin architecture and transaction lifecycle; distributed consensus and atomic broadcast; alternatives to Bitcoin consensus; Byzantine fault-tolerant mechanisms; alternative cryptocurrencies; blockchain-based cryptocurrency applications; technologies adopted in blockchain platforms.

UNIT-III ***(11 hours)***

Cryptographic Foundations and Bitcoin Scripting: Cryptographic hash functions; puzzle-friendly and collision-resistant hashes; digital signatures and public key cryptography; Bitcoin scripting language; transaction scripts; pay-to-public-key-hash (P2PKH); pay-to-script-hash (P2SH); Segregated Witness; hash time-locked contracts; atomic swaps.

UNIT-IV ***(12 hours)***

Security, Privacy, and Advanced Blockchain Applications: Security threats and vulnerabilities in blockchain systems; building secure cryptocurrency platforms; privacy-preserving mechanisms; zero-knowledge proofs; blockchain-based payment systems; smart contracts and decentralized applications; e-governance and enterprise blockchain solutions; emerging trends including DeFi, NFTs, and CBDCs.

Practical Component: ***(30 hours)***

1. Installation and configuration of the blockchain development environment.
2. Creation and management of cryptocurrency wallets.
3. Writing and deploying smart contracts.
4. Implementing token standards and transactions.
5. Case studies on blockchain platforms and real-world deployments.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. N. Daskalakis and P. Georgitseas, *An Introduction to Cryptocurrencies: The Crypto Market Ecosystem*, 1st ed., Routledge & CRC Press, 2020.
2. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction,* Princeton University Press, 2016.
3. A. M. Antonopoulos and D. A. Harding, *Mastering Bitcoin: Programming the Open Blockchain*, 3rd ed., O'Reilly Media, 2023.

Suggested Readings:

1. M. Grabowski, *Cryptocurrencies: A Primer on Digital Money*, 1st ed., Routledge, 2019.
2. D. Tapscott and A. Tapscott, *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World,* Penguin, 2016.

MULTIUSER AND SOCIAL AR/VR (DSE-7/ GE-8)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Multiuser and Social AR/VR	4	3	0	1	Foundations of Augmented and Virtual Reality, Computer Networks

Course Objectives:

1. To understand the architecture required to host multiple users in a shared virtual environment.
2. To implement real-time synchronization of avatars, spatial audio, and interactive objects.
3. To explore the psychological aspects of social presence and "embodiment" in XR.
4. To address the unique security and moderation challenges in social AR/VR spaces.

Course Outcomes:

1. Develop networked XR applications using frameworks like Photon (PUN/Fusion), Mirror, or Normcore.
2. Optimize network traffic for smooth interactions in low-bandwidth scenarios.
3. Design expressive avatars with synchronized lip-sync and hand-tracking.
4. Evaluate user experience (UX) based on spatial social cues and telepresence metrics.

UNIT-I ***(15 hours)***

Foundations of Networked XR: Architecture for XR: Client-Server vs. Peer-to-Peer vs. Related Architectures; Introduction to "Authoritative Servers"; Synchronization Challenges: State synchronization, Remote Procedure Calls (RPCs), and Network Observables; Latency Mitigation: Dead Reckoning, Linear Interpolation (Lerp), and Client-Side Prediction; AR Cloud: Shared spatial anchors and persistent world-mapping for multi-user AR.

UNIT-II ***(12 hours)***

Social Presence and Avatar Systems: Representation: High-fidelity vs. Low-poly avatars; The "Uncanny Valley" in social VR.; Embodiment: IK (Inverse Kinematics) for body tracking; Synchronizing head and hand movements.; Social Cues: Eye tracking, facial expression mapping, and procedural lip-syncing; Spatial Audio: 3D sound attenuation, HRTF (Head-Related Transfer Functions), and voice-chat occlusion.

UNIT-III ***(08 hours)***

Shared Interaction and Physics: Object Ownership: Handling collisions and interactions when multiple users grab the same object.; Concurrency Control: Locking mechanisms for shared UI and 3D assets; Multi-user AR: Collaborative design, remote assistance (See-What-I-See), and shared holograms; Navigation: Synchronized teleportation vs. continuous movement; Preventing motion sickness in shared spaces.

UNIT-IV ***(10 hours)***

Ethics, Privacy, and Scalability: **Social Safety:** Personal space bubbles, hand-gesture filtering, and real-time moderation tools; Privacy: Data security in spatial computing; Risks of biometric tracking (eye/gaze data); Massive Multiuser XR: Techniques for sharding and interest management (NetCulling); Case Studies: Analyzing VRChat, Rec Room, Horizon Worlds, and Microsoft Mesh.

Practical Component: ***(30 hours)***

1. Basic Connection: Set up a "Network Manager" to allow two users to join a shared VR room as simple primitive shapes.
2. Avatar Sync: Implement a full-body avatar that synchronizes the movement of a VR user's head and hands to remote players.
3. Spatial Voice Chat: Integrate a Voice-over-IP (VoIP) system where volume decreases as users move away from each other in 3D space.
4. Shared Interaction: Create a "Collaborative Whiteboard" where multiple users can draw or move 3D objects simultaneously.
5. Persistent AR: Use Azure Spatial Anchors or Niantic Lightship to place a digital object in a physical room that remains visible to multiple mobile AR users.
6. Safety Zone Logic: Script a "Proximity Bubble" that makes other players invisible/transparent if they get too close to the user's avatar.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. B. Shneiderman, *Human-Centered AI*, Oxford University Press, 2022.
2. J. Jerald, *The VR Book: Human-Centered Design for Virtual Reality,* Morgan & Claypool Publishers, 2015
3. T. Parisi, *Learning Virtual Reality: Developing Immersive Experiences and Applications for Desktop, Web, and Mobile*, O'Reilly Media, 2015.

Suggested Readings:

1. Unity Technologies, *Unity Multiplayer Networking Documentation (Netcode for GameObjects / Fusion)*, Unity Manual, latest edition.
2. M. Ball, *The Metaverse: And How It Will Revolutionize Everything,* Liveright Pub Corp, 2022

Detailed Syllabus of Generic Elective (GE) courses offered for Minors / Specializations by Department of CSE in SEMESTER VIII

LOW LEVEL SOFTWARE SYSTEM (GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Low Level Software System	4	3	0	1	Data Structures, Fundamentals of Operating Systems

Course Objectives:

1. Introduce system-level thinking bridging hardware and software.
2. Provide foundational understanding of firmware, kernel, and user-space interaction.
3. Develop awareness of performance and memory behavior in modern processors.
4. Familiarize students with real-time and embedded system considerations.
5. Introduce basic low-level security mechanisms in modern systems.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain how software interacts with hardware from boot process to application execution.
2. Understand basic kernel responsibilities including interrupts, scheduling, and memory management.
3. Analyze performance implications of low-level programming decisions.
4. Apply fundamental concepts of synchronization, real-time constraints, and system security.
5. Interpret compilation, linking, and execution flow at a system level.

UNIT-I ***(10 hours)***

Bootstrapping and Control Transfer: System Boot Process: Firmware concepts (BIOS/UEFI, U-Boot overview), Secure Boot and basic trust concepts, Memory Layout Basics: Stack, Heap, Data, BSS, Introduction to MMU and virtual memory, Binary formats (ELF) and program loading, Introduction to HAL (Hardware Abstraction Layer),Calling conventions and stack fundamentals

UNIT-II ***(13 hours)***

Kernel Runtime Evolution and Application Integration: Kernel roles and execution context, Basics of interrupts and interrupt handling, Timers and system clock concepts, Introduction to device drivers and I/O subsystems, System calls and user–kernel boundary, Overview of process creation and program execution, The exec Pipeline, Embedded vs general-purpose OS (Linux, FreeRTOS overview).

UNIT-III *(12 hours)*

Microarchitectural Performance, Memory Models, and Determinism: Hardware Atomics, Compare-and-Swap (CAS) vs. Load-Link/Store-Conditional (LL/SC), Memory Consistency Models, Cache Coherency Interaction, The Pipeline-Software Interface, Instruction-Level Parallelism (ILP), Synchronization problems, Real-time systems: Hard vs Soft real-time, Scheduling basics: Rate Monotonic and EDF (conceptual overview), Introduction to Worst-Case Execution Time (WCET)

UNIT-IV *(10 hours)*

Systems Toolchains, Optimization, and Low-Level Security: Compilation process, Static vs Dynamic linking (conceptual understanding), Compiler optimizations overview (-O levels), Cache-aware programming basics, Stack protection and memory protection concepts, Introduction to ASLR and DEP, Overview of Overview of speculative execution vulnerabilities (Spectre and Meltdown),Secure coding principles in system software.

Practical Component: *(30 hours)*

1. Analyze memory layout using C programs and tools (objdump, nm).
2. Study stack frames and calling conventions via simple assembly inspection.
3. Writing and analyzing interrupt-driven programs (simulation or embedded board).
4. Basic driver-style interaction (GPIO or simulated I/O).
5. Measuring execution time and observing compiler optimization effects.
6. Implementing simple synchronization examples using threads.
7. Mini-project: System-level performance or synchronization case study.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. J. L. Hennessy, D. A. Patterson, and C. Kozyrakis, *Computer Architecture: A Quantitative Approach*, 7th ed., Morgan Kaufmann, 2022.
2. D. Lacamera, *Embedded Systems Architecture: Explore architectural concepts, pragmatic design patterns, and best practices to produce robust systems*, 2nd ed., Packt Publishing, 2021.
3. R. E. Bryant and D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 3rd ed., Pearson Education, 2016.
4. R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Version 1.10, Arpaci-Dusseau Books, 2018.

Suggested Readings:

1. D. P. Bovet and M. Cesati, *Understanding the Linux Kernel*, 3rd ed., O'Reilly Media, 2005.
2. A. S. Tanenbaum and H. Bos, *Modern Operating Systems*, 5th ed., Pearson, 2022.
3. R. Jain, *The Art of Computer Systems Performance Analysis*, Wiley, 1991.
4. G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 4th ed., Springer, 2020.
5. Official white papers on Spectre and Meltdown vulnerabilities (Intel, ARM, and related security advisories).
6. Processor ISA manuals (e.g., Intel® 64 and IA-32 Architectures Software Developer's Manual, ARM Architecture Reference Manual) and ELF specification documents.

OOP CONCEPTS (GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
OOP Concepts	4	3	0	1	Fundamentals of Computer Programming

Course Objectives:

1. Introduce fundamental concepts of object-oriented programming and design.
2. Enable students to model real-world problems using classes and objects.
3. Provide hands-on experience with object-oriented languages and tools.
4. Promote good programming practices using modularity, reuse, and maintainability.
5. Familiarize students with exception handling, file I/O, and generic programming techniques

Course Outcomes:

At the end of this course, students will be able to:

1. Understand core object-oriented concepts such as abstraction, encapsulation, inheritance, and polymorphism.
2. Design and implement object-oriented solutions using classes, objects, and interfaces.
3. Apply principles of inheritance and polymorphism to develop reusable and extensible software.
4. Implement exception handling and file handling mechanisms for robust applications.
5. Use object-oriented design principles to model real-world systems.
6. Develop complete object-oriented applications using Java / C++.

UNIT-I ***(12 Hours)***

Introduction to Object Oriented Programming: Programming Paradigms; Procedural vs Object-Oriented Programming; Objects, Classes, Data Abstraction, Encapsulation; Benefits of OOP; Java/C++ Basics; Control Statements; I/O Operations.`

UNIT-II ***(10 Hours)***

Classes, Objects and Constructors: Class definition; Access Specifiers; Constructors and Destructor; Constructor Overloading; this pointer; Static Members; Memory Allocation.

UNIT-III ***(12 Hours)***

Inheritance and Polymorphism: Types of Inheritance; Method Overriding; Polymorphism; Function and Operator Overloading; Virtual Functions; Abstract Classes; Interfaces.

UNIT-IV ***(11 Hours)***

Advanced OOP Concepts and Applications: Exception Handling; Generic programming- Templates in C++.

Practical Component: ***(30 Hours)***

1. Development of programs demonstrating classes and objects.
2. Implementation of constructors and destructors in object-oriented programs.
3. Programs illustrating inheritance and polymorphism concepts.
4. Implementation of exception handling mechanisms.
5. File handling programs for reading and writing data.
6. Mini project applying object-oriented design principles.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. E. Balagurusamy, *Object Oriented Programming with Java*, 6th ed., McGraw Hill Education, 2019.
2. H. Schildt, *Java: The Complete Reference*, 11th ed., McGraw Hill Education, 2018.
3. R. Lafore, *Object-Oriented Programming in C++*, 4th ed., Sams Publishing, 2001.

Suggested Readings:

1. B. McLaughlin, G. Pollice, and D. West, *Head First Object-Oriented Analysis and Design*, 1st ed., O'Reilly Media, 2006.
2. R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 1st ed., Prentice Hall, 2008.
3. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed., Addison-Wesley, 1994.

GENERATIVE AI (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Generative AI	4	3	0	1	NIL

Course Objectives:

1. Introduce the mathematical and conceptual foundations of generative modeling.
2. Enable hands-on experience with modern generative architectures.
3. Provide practical exposure to training and fine-tuning large generative models.
4. Familiarize students with prompt engineering and alignment techniques.
5. Develop critical understanding of safety, bias, and responsible AI usage.
6. Prepare students for research and industry applications of Generative AI.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the theoretical foundations of generative modeling and representation learning.
2. Implement and train generative models such as autoencoders, VAEs, GANs, and diffusion models.
3. Apply large language models (LLMs) and transformer architectures for text generation and understanding.
4. Develop multimodal generative systems for text, image, and audio data.
5. Evaluate generative models using qualitative and quantitative metrics.
6. Analyze ethical, legal, and societal implications of Generative AI.

UNIT-I ***(10 hours)***

Foundations of Generative Modeling: Discriminative vs Generative Models; Probability Distributions and Likelihood-Based Learning; Maximum Likelihood Estimation; Latent Variable Models; Representation Learning; Autoencoders: Architecture, Training, and Limitations; Introduction to Variational Inference.

UNIT-II ***(13 hours)***

Variational Autoencoders and GAN: Variational Autoencoders (VAEs): Evidence Lower Bound (ELBO), Reparameterization Trick; Conditional VAEs; Generative Adversarial Networks (GANs): Minimax Objective, Training Dynamics, Mode Collapse; Variants: DCGAN, WGAN, CycleGAN; Evaluation Metrics: Inception Score, FID.

UNIT-III ***(12 hours)***

Diffusion Models and Transformers: Denoising Diffusion Probabilistic Models (DDPMs); Score-Based Generative Models; Sampling and Acceleration Techniques; Transformer Architecture: Self-Attention, Positional Encoding; Large Language Models (LLMs): Pretraining, Fine-Tuning, Instruction Tuning; Prompt Engineering and In-Context Learning.

UNIT-IV ***(10 hours)***

Multimodal Generative AI and Applications: Text-to-Image Models (e.g., DALL·E-style architectures); Image-to-Image Translation; Speech and Audio Generation; Multimodal Transformers; Retrieval-Augmented Generation (RAG); Applications in Content Creation, Healthcare, Education, and Scientific Research; Ethical, Legal, and Social Issues: Bias, Hallucination, Copyright, and AI Safety.

Practical Component: ***(30 hours)***

1. Implementing autoencoders and VAEs for image generation.
2. Training GANs for image synthesis tasks.
3. Implementing diffusion models for denoising and generation.
4. Fine-tuning transformer-based language models for text generation tasks.
5. Designing multimodal generative pipelines (text-to-image or image captioning).
6. Mini-project on a real-world Generative AI application.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
2. D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. (draft), Pearson, 2023.
3. I. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," arXiv:1701.00160, 2016.

Suggested Readings:

1. D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *International Conference on Learning Representations (ICLR)*, 2014.
2. A. Vaswani et al., "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
3. J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

SCIENTIFIC MACHINE LEARNING (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Scientific Machine Learning	4	3	0	1	NIL

Course Objectives:

1. Introduce students to the principles of Scientific Machine Learning, combining mechanistic models with machine learning.
2. Enable students to solve ODEs and PDEs using neural networks without traditional discretization methods.
3. Familiarize students with Physics-Informed Neural Networks (PINNs) and constrained learning frameworks.
4. Provide hands-on experience with operator learning and surrogate modeling.
5. Train students to handle uncertainty, interpretability, and robustness in scientific data-driven models.
6. Encourage interdisciplinary thinking through case studies and mini-projects.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the foundations of Scientific Machine Learning and its role in integrating physical laws with data-driven models.
2. Apply machine learning techniques to scientific and engineering problems involving noisy, sparse, and high-dimensional data.
3. Develop Physics-Informed Neural Networks (PINNs) for solving ordinary and partial differential equations.
4. Implement operator learning approaches such as DeepONets and Fourier Neural Operators for learning solution mappings.
5. Use SciML techniques for real-world applications in physics, healthcare, climate science, and engineering systems.
6. Critically evaluate model interpretability, generalization, and uncertainty in scientific ML applications.

UNIT-I ***(10 hours)***

Introduction to Scientific Machine Learning: Scientific Computing vs Data-Driven Modeling; Motivation for Scientific Machine Learning; Limitations of Purely Data-Driven ML; Role of Physical Laws, Constraints, and Conservation Principles; Overview of Differential Equations in Science and Engineering; Introduction to Hybrid Modeling; Examples from Fluid Dynamics, Biomedical Engineering, Climate Modeling, and Materials Science.

UNIT-II ***(13 hours)***

Machine Learning for Scientific Data: Scientific Data Characteristics: Noise, Sparsity, and Multiscale Nature; Regression and Classification for Scientific Applications; Gaussian Processes for Function Approximation; Kernel Methods; Dimensionality Reduction Techniques: PCA, Autoencoders; Model Generalization and Extrapolation in Scientific Domains; Bias–Variance Tradeoff in Physics-Based Systems.

UNIT-III ***(10 hours)***

Physics-Informed Neural Networks (PINNs): Neural Networks as Function Approximators; Automatic Differentiation; Formulation of Physics-Informed Loss Functions; Solving Ordinary Differential Equations (ODEs) using PINNs; Solving Partial Differential Equations (PDEs); Boundary and Initial Conditions Handling; Training Challenges and Optimization Strategies; Case Studies: Burgers' Equation, Heat Equation, and Navier–Stokes Equations.

UNIT-IV ***(12 hours)***

Advanced SciML Topics and Applications: Operator Learning: Deep Operator Networks (DeepONets), Fourier Neural Operators (FNOs); Reduced-Order Modeling and Surrogate Models; Uncertainty Quantification and Bayesian PINNs; Scientific Discovery using ML (Symbolic Regression and Equation Discovery); Ethical and Reliability Considerations in Scientific AI; Case Studies: Digital Twins, Medical Imaging, Climate Prediction, and Blood Flow Modeling.

Practical Component: ***(30 hours)***

1. Implementing regression and Gaussian Process models for scientific datasets.
2. Solving ODEs using neural networks and automatic differentiation.
3. Implementing Physics-Informed Neural Networks (PINNs) for benchmark PDEs.
4. Training PINNs for boundary value and initial value problems.
5. Implementing operator learning models such as DeepONets or Fourier Neural Operators.
6. Developing a mini-project applying SciML to a real-world scientific problem.
7. Developing a natural language processing pipeline for text classification, sentiment analysis, or spam detection.
8. Building and testing generative models (e.g., GANs) for image generation tasks.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Machine Learning," *Nature Reviews Physics*, vol. 1, pp. 1–19, 2021.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

Suggested Readings:

1. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.
2. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018

VISUAL COMPUTING (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Visual Computing	4	3	0	1	Probability and Statistics for Computer Science

Course Objectives:

1. Introduce fundamental concepts of visual computing and image formation.
2. Understand computer vision and graphics techniques for visual data processing.
3. Apply algorithms for image analysis, feature extraction, and visualization.
4. Explore real-world applications of visual computing systems.

Course Outcomes:

1. Explain core principles of visual computing and image representation.
2. Implement basic image processing and computer vision algorithms.
3. Analyze visual data for recognition and interpretation tasks.
4. Apply visual computing techniques to real-world applications.

UNIT-I *(12 hours)*

Introduction to Visual Computing: Overview of Visual Computing, Image Formation, Image Representation, Color Models, Human Visual System, Digital Image Fundamentals.

UNIT-II *(12 hours)*

Image Processing Techniques: Image enhancement, filtering, edge detection, segmentation, morphological operations, frequency domain analysis.

UNIT-III *(12 hours)*

Computer Vision and Feature Analysis: Feature detection and description, object detection, motion analysis, camera models, stereo vision, Overview of 3D reconstruction

UNIT-IV *(9 hours)*

Advanced Topics and Applications: Visual learning and deep vision (overview), graphics pipelines, visualization techniques, applications in AR/VR, medical imaging, autonomous systems and smart surveillance.

Practical Component: *(30 hours)*

1. Implementation of programs for image representation and geometric transformations such as scaling, rotation, and translation.
2. Development of image filtering and enhancement techniques using spatial and frequency domain methods.

3. Implementation of feature extraction and feature matching algorithms.
4. Development of basic object detection and tracking applications.
5. Mini-project involving the design and implementation of a visual computing application.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed., Springer, 2022.
2. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed., Pearson, 2018.
3. D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, 2nd ed., Pearson, 2011.

Suggested Readings:

1. A. Watt, *3D Computer Graphics*, 3rd ed., Addison-Wesley, 2000.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016

‘

APPLIED AI (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Applied AI	4	3	0	1	Fundamentals of Data Analytics

Course Objectives:

1. Provide a structured understanding of AI techniques tailored to specific data types: numeric, text, image, audio, video, and time-series.
2. Develop skills in preprocessing, feature engineering, and modeling for different data modalities.
3. Enable students to select, design, and implement appropriate AI/ML pipelines based on data characteristics.
4. Foster the ability to build end-to-end data analytics solutions using real-world datasets.
5. Prepare students for industry roles in AI engineering, data science, and analytics.

Course Outcomes:

On completion of this course, students will be able to:

1. To understand how AI techniques are applied to different types of data in real-world analytics.
2. To build competency in selecting appropriate AI models based on data characteristics.
3. To apply AI techniques for numeric, text, multimedia, and time-series data.
4. To develop analytical, modeling, and decision-making skills using AI-driven approaches.

UNIT-I ***(12 hours)***

AI for Numeric and Structured Data: Introduction to Data Types in AI (Numeric, Text, Audio/Video, Images, Time series), Characteristics of Numeric and Structured Data, Data Collection and Quality issues, Data Preprocessing, Handling missing values, Data Normalization and Standardization, Outlier Detection and Treatment, Exploratory Data Analysis (EDA), Data Visualization Techniques, ML for Numeric Data: Linear and Multiple Regression, Logistic Regression, k-Nearest Neighbour, Decision Trees and Ensemble Methods (Random Forest), Evaluation Metrics (Accuracy, Precision, Recall, RMSE).

UNIT-II ***(10 hours)***

AI for Text and Natural Language Data: Introduction to Unstructured and Textual data: Challenges and Opportunities, Text Preprocessing Techniques (Tokenization, Stop-word removal, Stemming and Lemmatization), Text Representation Methods: Bag-of-Words, TF-IDF, Word Embeddings (Word2Vec, GloVe), Introduction to Natural Language Processing: Tasks and Applications, Sentiment Analysis, Topic Modeling, Text Classification, Named Entity Recognition, Transformers and BERT Architecture, Fine-tuning Pre-Trained Language Models for Downstream Tasks.

UNIT-III ***(11 hours)***

AI for Image, Audio, and Video Analytics: Introduction to Multimedia Data, Digital Image Representation, Image Processing and Enhancement, Feature Extraction, Fundamentals of Computer Vision, Convolutional Neural Networks (CNN) Architecture, CNN Applications in Image Analytics,

Object Detection and Image Classification, Audio Data Analytics, Audio Signal Representation, Feature Extraction from Audio Data, Audio Classification and Speech Recognition Basics, Introduction to Video Analytics (Frame Extraction, Motion Analysis, Video Classification).

UNIT-IV ***(10 hours)***

AI for Time-series and Sequential Data Analytics: Introduction to Time-Series and Sequential Data, Characteristics and Challenges of Temporal Data, Time-series Data Processing and Visualization, Trend, Seasonality, and Noise Analysis, Feature Engineering for Time-dependent Data, Traditional Time-series Models, Moving Average Models, Exponential Smoothing, ARIMA, ML techniques for Time-series Forecasting (Feature-based Regression, XGBoost), Deep Learning Models for Sequential Data, RNNs, LSTM Networks, Anomaly Detection in Time-series Data.

Practical Component: ***(30 hours)***

1. Data preprocessing and EDA on structured datasets, including handling missing values, normalization, and visualization.
2. Implementation of machine learning models for numeric data (Linear Regression, Logistic Regression, k-NN, Decision Trees, Random Forest) with performance evaluation.
3. Text preprocessing and representation using Bag-of-Words, TF-IDF, and word embeddings; implementation of text classification or sentiment analysis.
4. Fine-tuning of a pre-trained Transformer-based model (e.g., BERT) for a downstream NLP task.
5. Image classification using CNNs and basic object detection using deep learning frameworks.
6. Audio data preprocessing and feature extraction (e.g., MFCCs) with implementation of a simple audio classification model.
7. Time-series visualization and forecasting using ARIMA and machine learning models (e.g., regression/XGBoost).
8. Implementation of LSTM-based sequence models for time-series prediction or anomaly detection.
9. Mini-project involving an end-to-end AI pipeline on a real-world dataset (numeric, text, multimedia, or time-series), including preprocessing, modeling, evaluation, and reporting.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed., O'Reilly Media, 2022.
2. J. D. Kelleher, *Deep Learning*, MIT Press, 2019.
3. S. Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed., Packt Publishing, 2019.
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

Suggested Readings:

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
2. D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed., Pearson, 2014.
3. A. C. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists*, O'Reilly Media, 2016.

PROGRAM ANALYSIS AND VERIFICATION (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Program Analysis and Verification	4	3	0	1	Data Structures

Course Objectives:

1. To introduce the mathematical and logical foundations required for program analysis and verification.
2. To study classical dataflow analysis techniques and pointer analysis for imperative programs.
3. To understand abstract interpretation as a principled framework for sound program analysis.
4. To introduce formal reasoning, automated verification techniques, and systematic program testing methods.
5. To develop the ability to reason about program correctness, safety, and behavior using formal and semi-formal methods.

Course Outcomes:

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

1. Apply lattice theory and logical formalisms to reason about program properties.
2. Analyze programs using dataflow analysis techniques such as constant propagation, reaching definitions
3. Explain and apply abstract interpretation concepts, including abstract domains, Galois connections, and widening/narrowing, to approximate program semantics.
4. Reason about program correctness using Hoare logic, weakest preconditions, and model checking techniques.
5. Use automated verification approaches, including SMT/SAT solving.

UNIT-I ***(8 hours)***

Foundational Topics: Partial orders and lattices: Reflexivity, antisymmetry, transitivity, Least upper bound (join), greatest lower bound (meet). Propositional Logic.

UNIT-II ***(10 hours)***

Dataflow Analysis: Program Representation. Dataflow Analysis: Constant Propagation, Reaching Definitions, Available Expressions, Kildall's algorithm

UNIT-III ***(12 hours)***

Abstract Interpretation: Abstract domains and lattices, Abstract join-over-all-paths analysis, Galois connections and abstraction/concretization functions, Abstract interpretation as approximation of concrete semantics. Numerical abstract domains: intervals, affine equalities. Widening and narrowing.

UNIT-IV ***(15 hours)***

Program Correctness and Verification: Floyd-Hoare Logic, Weakest Precondition, Software Model Checking: symbolic execution, state-space reduction, state-less model checking, counter-example guided abstraction refinement (CEGAR), Introduction to SMT and Solvers: SAT, SMT, DPLL.

Essential Readings:

1. A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, Compilers: Principles, Techniques, and Tools, 2nd ed., Pearson, 2006.
2. Jeannet, B. and A. Miné, Apron: (A) Library of Numerical Abstract Domains for Static Analysis. Lecture Notes in Computer Science (LNCS). Springer. 2009.
3. E. M. Clarke, O. Grumberg, and D. A. Peled. Model Checking, MIT Press, 1999.

Suggested Readings:

1. F. Nielson, H. R. Nielson, and C. Hankin, Principles of Program Analysis, Springer, 1999.
2. Cousot, R. and Cousot, P., Static Analysis Frameworks. Lecture Notes in Computer Science (LNCS), Springer, 1997.
3. D. Kroening and O. Strichman, *Decision Procedures:* An Algorithmic Point of View, 2nd ed., Springer, 2016.
4. B. C. Pierce et al., Software Foundations: Programming Language Foundations, Electronic textbook, 2018

Practical Component: ***(30 hours)***

1. Program representation and control-flow analysis exercises using static analysis tools (e.g., construction of Control Flow Graphs and intermediate representations).
2. Implementation and experimentation with classical dataflow analyses such as Constant Propagation, Reaching Definitions, and Available Expressions using compiler or analysis frameworks.
3. Development and evaluation of abstract interpretation–based analyses using numerical abstract domains (e.g., interval analysis) with widening and narrowing strategies.
4. Formal specification and verification of simple programs using Floyd–Hoare logic and weakest precondition reasoning.
5. Program verification using model checking tools, including symbolic execution and counterexample analysis.
6. Hands-on exercises with SAT/SMT solvers for constraint solving and automated reasoning tasks.
7. Mini-project involving static analysis or formal verification of a small software system using appropriate tools and frameworks.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

EMPIRICAL SOFTWARE ENGINEERING (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Empirical Software Engineering	4	3	0	1	Software Engineering, Object Oriented Software Engineering

Course Objectives:

1. To understand fundamental concepts and importance of empirical software engineering in practice.
2. To study software metrics, basic experimental design, and data collection techniques
3. To analyze software repositories and datasets for deriving useful insights.
4. To apply basic statistical and machine learning techniques for software data analysis.
5. To develop skills for conducting structured literature reviews and presenting empirical findings.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain basic concepts, processes, and ethical aspects of empirical software engineering.
2. Design simple empirical studies including formulation of research questions and data collection.
3. Analyze software metrics and repository data using appropriate tools and techniques.
4. Apply statistical and basic machine learning methods for data analysis and interpretation.
5. Prepare structured reports based on empirical studies and literature reviews.

UNIT-I ***(11 hours)***

Foundations of Empirical Software Engineering & Systematic Reviews: Introduction to empirical software engineering: concepts, motivation, and types of empirical studies; Empirical research process, study design, and , ethical considerations; Systematic Literature Reviews (SLR): planning, conducting, data extraction, synthesis, and reporting.

UNIT-II ***(11 hours)***

Software Metrics and Experimental Design*:*Fundamentals of software measurement and metrics (size, quality, object-oriented, and evolution metrics); Validation and industrial relevance of software metrics; Experimental design in software engineering: research questions, variables, hypothesis formulation, data collection, and selection of analysis techniques.

UNIT-III ***(10 hours)***

Mining Software Repositories and Data Sources*:* Introduction to mining software repositories (MSR); Version control systems, bug tracking systems, and configuration management data; Data extraction, static code analysis, and historical software analysis; Use of open research datasets for empirical studies.

UNIT-IV ***(13 hours)***

Data Analysis, Model Development, and Research Reporting: Statistical analysis of software engineering data and hypothesis testing; Model development using statistical and machine learning

techniques; Model evaluation, cross-validation, and performance metrics; Text mining applications in software engineering; Threats to validity, countermeasures, ethical reporting, and research documentation.

Practical Component: ***(30 hours)***

1. Systematic literature review exercises including research question formulation and structured reporting
2. Software metrics computation and validation using statistical analysis techniques.
3. Mining and analysis of data from software repositories such as Git and bug tracking systems
4. Statistical hypothesis testing and empirical data analysis using appropriate tools.
5. Development and evaluation of fault prediction models using statistical and machine learning techniques.
6. Model comparison and interpretation using performance evaluation metrics and cross-validation.
7. Text mining and classification of defect reports using feature extraction and ML methods.
8. Mini-project involving end-to-end empirical study including data collection, analysis, modeling, validation, and research reporting.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. R. Malhotra, *Empirical Research in Software Engineering: Concepts, Analysis, and Applications*, CRC Press / Taylor & Francis, 2016.
2. C. Wohlin et al., *Experimentation in Software Engineering*, 2nd ed., Springer, 2012.
3. T. Menzies, L. Williams, and T. Zimmermann (eds.), *Perspectives on Data Science for Software Engineering*, Morgan Kaufmann, 2016.

Suggested Readings:

1. F. Shull, J. Singer, and D. I. K. Sjøberg (eds.), *Guide to Advanced Empirical Software Engineering*, Springer, 2008.
2. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed., Morgan Kaufmann, 2016.

CYBER LAW AND DIGITAL FORENSICS (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Cyber Law and Digital Forensics	4	3	0	1	Cryptography Essentials

Course Objectives:

1. To understand security issues in the network and application layers.
2. To learn concepts related to cybercrime, cyber law, and digital investigations.
3. To acquire knowledge of forensic tools and techniques for evidence analysis.
4. To develop skills for the investigation and reporting of cybercrime incidents.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand various cybercrimes and cyber laws.
2. Analyze digital evidence and perform forensic investigations.
3. Apply forensic tools and techniques to identify and trace cyber attackers.
4. Interpret digital forensic findings for legal and investigative purposes

UNIT-I ***(11 hours)***

Introduction to Cyber Law and Cyber Crimes*:* Internet and cyberspace; cybercrime and its classification; hacking, cracking, viruses, malware, and cyber-attacks; pornography and intellectual property crimes; legal framework of cyber laws; Information Technology Act; social engineering; mail bombing; bug exploits; ethical and legal issues in cyber investigations.

UNIT-II ***(11 hours)***

Forensic Science and Investigation Process: Principles of digital forensics; scientific approach to forensic investigation; identification and classification of digital evidence; evidence acquisition and preservation; chain of custody; evidence handling and storage; forensic investigation methodology; documentation and reporting.

UNIT-III ***(11 hours)***

Digital and Network Forensics: Hardware forensics; file systems and hidden data; anti-forensics techniques; network forensics; analysis of network traffic; virtual systems and virtualization forensics; mobile device forensics; watermarking techniques; secure computation protocols

UNIT-IV ***(12 hours)***

Application Forensics and Cybercrime Investigation: Application forensics; email forensics; social media and cloud forensics; investigation of cybercrimes; forensic tools and techniques; cybercrime case studies; preparation of forensic reports; presentation of digital evidence in legal proceedings.

Practical Component: ***(30 hours)***

1. Identification and analysis of cybercrime incidents using real-world case studies.
2. Acquisition and preservation of digital evidence following standard forensic procedures.
3. File system analysis, including recovery of deleted files and examination of metadata.
4. Disk imaging and verification using forensic tools.
5. Analysis of system logs, registry entries, and user activity artifacts.
6. Network traffic capture and analysis for identifying malicious activity.
7. Email, web, and social media forensic investigation.
8. Basic malware and suspicious file analysis in a controlled environment.
9. Preparation of forensic investigation reports with proper documentation and legal compliance.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. B. Nelson, A. Phillips, and C. Steuart, *Guide to Computer Forensics and Investigations*, 6th ed., Cengage Learning, 2020.
2. J. R. Vacca, *Computer Forensics: Computer Crime Scene Investigation*, 2nd ed., Charles River Media, 2005.
3. N. Jain and D. R. Kalbande, *Digital Forensic: The Fascinating World of Digital Evidence*, 1st ed., Wiley, 2016.

Suggested Readings:

1. C. Easttom, *Computer Security Fundamentals*, 4th ed., Pearson, 2019.
2. M. Britz, *Computer Forensics and Cyber Crime : An Introduction*, 4th ed., Pearson, 2020.

CRYPTANALYSIS & SECURITY MODELS (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Cryptanalysis & Security Models	4	3	0	1	Cryptography Essentials

Course Outcomes:

At the end of this course, students will be able to:

1. Understand classical and modern cryptanalysis techniques and their impact on cryptographic algorithms.
2. Analyze security models and adversarial assumptions.
3. Evaluate cryptographic protocol security using formal models.
4. Apply cryptanalysis methods to assess system security.
5. Identify security vulnerabilities and propose appropriate countermeasures.

Course Objectives:

1. Provide a thorough understanding of cryptanalysis techniques and security breaches.
2. Examine formal models of security and security proofs for cryptographic protocols.
3. Study security adversary models, attack strategies, and countermeasures.
4. Introduce security model frameworks used to analyze modern protocols.

UNIT-I ***(11 hours)***

Introduction to Cryptography & Cryptanalysis: Security threats, attacks, and models in network security; Security attacks, services and mechanisms; OSI security architecture and security policies; Cryptography basics; Classical encryption techniques - substitution and transposition ciphers; Introduction to cryptanalysis - ciphertext-only, known-plaintext, chosen-plaintext, chosen-ciphertext attacks; Goals of cryptanalysis.

UNIT-II ***(11 hours)***

Design and Cryptanalysis of Symmetric Ciphers: Block cipher design principles; Differential cryptanalysis; Linear cryptanalysis; Other modern attacks (integral, meet-in-the-middle); Modes of operation and their security implications; Security definitions - semantic security, IND-CPA, IND-CCA; Security of modern symmetric primitives and provable security concepts as applied to block ciphers and stream ciphers.

UNIT-III ***(11 hours)***

Public-Key Cryptanalysis & Security Models: Asymmetric cryptography overview: RSA, Diffie-Hellman; Adversarial models for public-key systems; Security definitions for public-key encryption schemes; Chosen-plaintext and chosen-ciphertext security models; Introduction to provable security and simulation-based security; Universal Composability (UC) security model basics.

UNIT-IV ***(12 hours)***

Security Protocols & Formal Security Analysis: Security protocol basics; Model of protocol execution; Authentication and key-exchange protocol analysis; Security proof techniques; Security protocols: SSL/TLS, IPSec, Kerberos, PGP; Security architectures and models; Countermeasures and modern cryptographic defenses such as elliptic curve cryptography.

Practical Component: ***(30 hours)***

1. Implementation of classical ciphers and basic cryptanalysis (e.g., substitution, transposition).
2. Differential and linear cryptanalysis simulations on test ciphers.
3. Evaluation of symmetric cipher security using IND-CPA and IND-CCA models.
4. Analysis of RSA and Diffie-Hellman under various attack models (chosen-plaintext / ciphertext).
5. Hands-on protocol security evaluation (SSL/TLS, IPSec).
6. Security tool experiments (e.g., hash collision tests, brute-force simulations).

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed., Pearson, 2023.
2. J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 3rd ed., CRC Press, 2020.
3. D. R. Stinson and M. Paterson, *Cryptography: Theory and Practice*, 4th ed., CRC Press, 2018.

Suggested Readings:

1. V. Nachef, J. Patarin, E. Volte, *Feistel Ciphers: Security Proofs and Cryptanalysis*, 1st ed., Springer, 2017.
2. M. Stamp and R. M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*, Wiley-IEEE Press, 2007.

AR/VR APPLICATION DEVELOPMENT (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
AR/VR Application Development	4	3	0	1	Foundations of Augmented and Virtual Reality

Course Objectives:

1. To understand the hardware and software architecture of AR/VR systems.
2. To design immersive user interfaces (UI) and user experiences (UX) tailored for spatial computing.
3. To develop interactive AR applications for mobile (ARCore/ARKit) and VR applications for headsets (Quest/Vive).
4. To explore the integration of spatial audio, haptics, and physics in virtual environments.

Course Outcomes:

By the end of this course, the student will be able to:

1. Identify and differentiate between various AR (Marker-based, Markerless) and VR (3DoF vs 6DoF) technologies.
2. Build functional 3D scenes with optimized assets and realistic lighting.
3. Implement complex interaction patterns such as teleportation, object manipulation, and gaze-based input.
4. Deploy cross-platform XR applications using OpenXR standards.

UNIT-I ***(12 hours)***

Introduction to XR Ecosystem: Taxonomy of Mixed Reality: The Milgram-Kishino Continuum; Difference between AR, VR, and MR;Hardware Foundations: Sensors (IMUs, Cameras, LIDAR), Displays (OLED, LCoS, Waveguides), and Tracking (Inside-out vs. Outside-in);Degrees of Freedom (DoF): 3DoF vs. 6DoF tracking; Human perception and the Vestibular system (reducing Motion Sickness);Development Tools: Overview of Unity, Unreal Engine, and WebXR.

UNIT-II ***(12 hours)***

Virtual Reality Development: Scene Construction: Importing 3D assets, Prefabs, and Scene hierarchy; VR Interaction Frameworks: XR Interaction Toolkit (XRI); Interactors, Interactables, and Sockets; Locomotion Systems: Teleportation, Continuous movement, and Snap-turn mechanics; Physics in VR: Rigidbodies, Colliders, and Trigger zones for immersive interaction.

UNIT-III ***(10 hours)***

Augmented Reality Development: AR Fundamentals: Environmental understanding (Plane detection), Motion tracking, and Light estimation.;Tracking Techniques: Image tracking (Marker-based) vs. Ground plane tracking (Markerless); Face tracking and Occlusion; AR Cloud & Persistence: Introduction to Spatial Anchors and Geo-spatial AR;UX for AR: Visual affordances, screen-space UI vs. world-space UI, and "The Hand" as an input device.

UNIT-IV ***(11 hours)***

Advanced XR Features and Optimization: Spatial Audio: 3D Sound spatialization, HRTF, and Doppler effect for immersion;Performance Optimization: Draw call batching, Occlusion culling, and LOD (Level of Detail) for mobile XR;Input Systems: Controller input, Hand tracking, Gaze-based selection, and Voice commands;Publishing: Build settings for Android/iOS (AR) and Standalone VR headsets (Meta Quest/PICO).

Practical Component: ***(30 hours)***

1. Environment Design: Create a 3D "Virtual Gallery" with baked lighting and optimized textures.
2. Basic Interaction: Script a "Physics Playground" where users can pick up, throw, and stack 3D objects using VR controllers.
3. Teleportation Logic: Implement a custom locomotion system with "Teleportation Areas" and "Teleportation Anchors."
4. Marker-based AR: Develop an AR mobile app that displays a 3D animated model when a specific "Image Target" (logo/business card) is scanned.
5. Furniture Placement (AR): Create a markerless AR app that allows users to place and rotate virtual furniture on a real-world detected floor.
6. Gaze-based Menu: Design a VR "Gaze UI" where looking at a button for 2 seconds triggers a scene change.
7. Hand Tracking Lab: Implement a "Virtual Piano" where users play notes using tracked finger movements (no controllers).

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. S. M. LaValle, *Virtual Reality*, Cambridge University Press, 2023.
2. J. Linowes, *Unity 2020 Virtual Reality Projects*, Packt Publishing, 2020.
3. D. Schmalstieg and T. Höllerer, *Augmented Reality: Principles and Practice*, Addison-Wesley / Pearson, 2016.

Suggested Readings:

1. J. Jerald, *The VR Book: Human-Centered Design for Virtual Reality*, ACM Books / Morgan & Claypool, 2015.

2. Unity Technologies, *Unity Multiplayer Networking Documentation* (Netcode for GameObjects / Fusion), Unity Manual, latest edition.
3. M. Ball, *The Metaverse: And How It Will Revolutionize Everything*, Liveright Pub Corp, 2022

COMPUTER VISION FOR AR/VR (DSE-9/GE-9)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Computer Vision For AR/VR	4	3	0	1	Foundations of Augmented and Virtual Reality

Course Objectives:

1. Provide students with foundational knowledge of computer vision concepts and basic image processing techniques for AR/VR systems.
2. Introduce feature detection, description, and camera geometry techniques essential for tracking, pose estimation, and spatial understanding in AR/VR applications.
3. Enable students to analyze motion, tracking, and depth perception methods for immersive environments.
4. Introduce students to practical computer vision applications in AR/VR, including hand tracking, gesture recognition, and interactive scene understanding.
5. Provide students with the ability to understand vision-based AR/VR prototypes while considering real-time performance, latency, and optimization constraints.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand core computer vision principles essential to AR/VR systems.
2. Apply real-time vision algorithms for tracking, pose estimation, and environment mapping.
3. Integrate vision algorithms into AR/VR applications and evaluate system performance.
4. Analyze the limitations and challenges of vision-based perception

UNIT-I

Introduction to Computer Vision for AR/VR ***(12 hours)***

Definition and Scope of Computer Vision, Role of Vision in AR/VR Systems, Human Visual Perception and Its Relevance to AR/VR, Image Formation and Representation, Camera Models: Pinhole Camera Model, Perspective Projection, Basic Image Processing Operations: Filtering, Edge Detection, and Image Enhancement.

UNIT-II

Feature Detection and Camera Geometry ***(12 hours)***

Interest Point and Feature Detection Techniques, Feature Descriptors and Matching, Camera Calibration, Homogeneous Coordinates and 3D Transformations, Multiple View Geometry: Epipolar Geometry, Fundamental and Essential Matrices, Pose Estimation Techniques for AR Object Placement.

UNIT-III

Tracking, Motion Analysis, and Depth Perception ***(11 hours)***

Object Tracking Techniques - Marker-Based and Markerless Tracking, Optical Flow Methods - Lucas–Kanade and Horn–Schunck Algorithms, Motion Estimation and Visual Odometry, Depth Estimation Techniques: Stereo Vision, Disparity Maps, and Triangulation, Structure from Motion, Applications of Depth and Motion in AR/VR Interaction and Scene Understanding.

UNIT-IV

SLAM and AR/VR Applications ***(10 hours)***

Simultaneous Localization and Mapping, Visual SLAM and Visual-Inertial SLAM for AR/VR Devices, Sensor Fusion: Integration of Cameras, IMU, and Depth Sensors. Computer Vision in VR: Hand Tracking, Gesture Recognition, and Eye Tracking. Performance Considerations: Real-Time Constraints, Latency, and Optimization.

Practical Component: ***(30 hours)***

1. Study image representation and color space conversions using OpenCV.
2. Perform image preprocessing and enhancement using filtering and edge detection techniques.
3. Implement feature detection algorithms.
4. Perform feature matching and homography-based object recognition.
5. Develop a marker-based augmented reality system for virtual object overlay.
6. Implement optical flow algorithms for motion estimation in video sequences.
7. Design a real-time object tracking system using computer vision techniques.
8. Estimate depth using stereo vision and generate disparity maps.
9. Reconstruct 3D scenes using structure from motion techniques.
10. Implement a basic visual SLAM pipeline for localization and mapping.
11. Develop hand or gesture tracking for interactive AR/VR applications.
12. Integrate computer vision modules with Unity or Unreal Engine for AR/VR systems.
13. Analyze real-time performance, latency, and optimization in vision-based AR/VR applications.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. Richard Szeliski, Computer Vision: Algorithms and Applications.
2. Learning OpenCV 4 by Gary Bradski & Adrian Kaehler.
3. Programming Computer Vision with Python.

Suggested Readings:

1. Robotics Vision and Control by Peter Corke.
2. Dieter Schmalstieg and Tobias Höllerer, Augmented Reality: Principles and Practice.

FUNDAMENTALS OF MACHINE LEARNING (GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Fundamentals of Machine Learning	4	3	0	1	Fundamentals of Computer Programming, Fundamentals of Algorithms

Course Objectives:

1. To introduce the fundamental concepts and terminology of machine learning.
2. To understand different types of learning: supervised, unsupervised, and reinforcement learning.
3. To develop basic skills in data preprocessing, model building, and evaluation.
4. To apply machine learning techniques to real-world problems from diverse domains.
5. To provide hands-on experience using simple ML tools and Python-based libraries.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain basic machine learning concepts and learning paradigms.
2. Prepare and preprocess datasets for analysis.
3. Build simple regression and classification models.
4. Evaluate model performance using standard metrics.
5. Apply machine learning techniques to domain-specific problems.

UNIT-I ***(10 hours)***

Introduction to Machine Learning and Data: Introduction to Machine Learning: definition, applications, and real-world use cases; Types of learning: supervised, unsupervised, and reinforcement learning;Overview of ML workflow; Types of data and data preprocessing: handling missing values, normalization, feature scaling; Introduction to Python and ML libraries (NumPy, Pandas, Scikit-learn).

UNIT-II ***(14 hours)***

Supervised Learning: Regression: Linear Regression, Multiple Regression; Classification: Logistic Regression, k-Nearest Neighbour (k-NN), Decision Trees; Model training and testing; Evaluation metrics: Accuracy, Precision, Recall, F1-score, Confusion Matrix, RMSE; Overfitting and underfitting.

UNIT-III ***(09 hours)***

Unsupervised Learning: Clustering: k-Means clustering, Hierarchical clustering; Dimensionality reduction: Introduction to PCA; Basic visualization techniques for high-dimensional data; Applications in business, healthcare, and social sciences.

UNIT-IV ***(12 hours)***

Introduction to Advanced Topics and Applications: Basics of Neural Networks, Introduction to model validation and cross-validation, Feature selection and importance, Ethical considerations in machine learning, Case studies from real world.

Practical Component: ***(30 hours)***

1. Data preprocessing and visualization using Python libraries.
2. Implementation of Linear Regression for real-world datasets.
3. Implementation of classification models (Logistic Regression, k-NN, Decision Tree).
4. Clustering using k-Means and visualization of clusters.
5. Dimensionality reduction using PCA.
6. Model evaluation using performance metrics and cross-validation.
7. Mini-project applying machine learning to a domain-specific dataset.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed., O'Reilly Media, 2022.
2. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann, 2011.
3. S. Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed., Packt Publishing, 2019.

Suggested Readings:

1. T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
2. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed., Morgan Kaufmann, 2016.
3. J. D. Kelleher, *Deep Learning*, MIT Press, 2019.

FUNDAMENTALS OF CLOUD COMPUTING (GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Fundamentals of Cloud Computing	4	3	0	1	Foundations of Computer Networks

Course Objectives:

1. To introduce fundamental concepts and architecture of cloud computing.
2. To understand the service and deployment models of cloud platforms.
3. To explain virtualization, storage, and networking in cloud environments.
4. To provide basic hands-on experience in deploying applications in the cloud.
5. To develop an understanding of cloud security principles and governance.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain cloud computing concepts, characteristics, and architecture.
2. Differentiate between cloud services and deployment models.
3. Describe virtualization, storage, and networking mechanisms in cloud systems.
4. Deploy and manage basic cloud-based applications.
5. Identify cloud security risks and apply fundamental protection mechanisms.

UNIT-I ***(11 hours)***

Introduction to Cloud Computing (Foundations): Evolution of Computing: Distributed Systems, Grid Computing, Cloud Definition and Essential Characteristics, Cloud Architecture: Front-end, Back-end, Middleware, Data Center Service Models: IaaS, PaaS, SaaS, Deployment Models: Public Cloud, Private Cloud, Hybrid Cloud, Community Cloud, Cloud Economics: Pay-as-you-go model, CAPEX vs OPEX, Cost-benefit considerations, Challenges: Vendor lock-in, Downtime, Compliance issues

UNIT-II ***(11 hours)***

Cloud Infrastructure and Virtualization: Virtualization Concepts: Hypervisors (Type I & Type II), Virtual Machines Containers (Basic Introduction), Resource Pooling and Multi-tenancy, Cloud Storage: Object Storage, Block Storage, File Storage, Introduction to Cloud Databases: Relational vs NoSQL (Conceptual level), Cloud Networking Basics: Virtual Private Cloud (VPC), Load Balancing, Content Delivery Network (CDN), Introduction to Cloud Service Providers

UNIT-III ***(11 hours)***

Cloud Application Deployment and Management: Cloud Application Architecture, Basic Cloud Design Principles, Introduction to DevOps in Cloud, CI/CD Overview (Conceptual), Basic Containerization using Docker, Introduction to Kubernetes, Serverless Computing (FaaS – Basic Concepts), Monitoring and Auto-scaling (Introduction).

UNIT-IV ***(12 hours)***

Cloud Security and Governance: Security Challenges in Cloud, Shared Responsibility Model, Identity and Access Management (IAM), Data Encryption (At Rest & In Transit), Backup and Disaster Recovery, Service Level Agreements (SLA), Compliance and Governance Basics, Ethical and Legal Issues in Cloud

Practical Component: ***(30 hours)***

1. Creating and configuring a Virtual Machine in a public cloud.
2. Deploying a static website using a cloud storage service.
3. Implementing basic Docker containerization.
4. Creating a simple serverless function.
5. Configuring IAM roles and policies.
6. Monitoring cloud resource usage and cost estimation.
7. Mini Project: Deploy a small web application on the cloud.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. T. Erl, R. Puttini, and Z. Mahmood, *Cloud Computing: Concepts, Technology & Architecture*, 1st ed., Prentice Hall, 2013.
2. Amazon Web Services, *AWS Documentation*, Available: https://docs.aws.amazon.com/
3. K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*, 2nd ed., O'Reilly Media, 2019.

Suggested Readings:

1. R. Buyya, C. Vecchiola, and S. Thamarai Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*, 1st ed., Morgan Kaufmann, 2013.
2. T. Erl, R. Cope, and A. Naserpour, *Cloud Computing Design Patterns*, 1st ed., Pearson, 2015.

AUTOMATED MACHINE LEARNING (GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Automated Machine Learning	4	3	0	1	Fundamentals of Data Analytics

Course Objectives:

1. Introduce the principles and evolution of AutoML systems.
2. Reduce manual effort in ML pipeline design through automation.
3. Enable hands-on experience with state-of-the-art AutoML frameworks.
4. Familiarize students with optimization techniques used in AutoML.
5. Develop understanding of interpretability and trust in automatically generated models.
6. Prepare students for industrial-scale ML deployment and research.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the motivation, scope, and challenges of Automated Machine Learning.
2. Explain the components of an end-to-end AutoML pipeline.
3. Apply automated techniques for feature engineering, model selection, and hyperparameter optimization.
4. Implement AutoML frameworks for classification, regression, and time-series tasks.
5. Evaluate AutoML systems in terms of performance, efficiency, and interpretability.
6. Analyze ethical, reproducibility, and deployment issues related to AutoML systems.

UNIT-I ***(10 hours)***

Introduction to Automated Machine Learning: Motivation for AutoML; Challenges in Traditional ML Pipelines; End-to-End ML Workflow; Search Spaces and Optimization Objectives; Taxonomy of AutoML Systems; Automation vs Human-in-the-Loop; Overview of Industrial AutoML Platforms and Research Trends.

UNIT-II ***(10 hours)***

Automated Data Processing and Feature Engineering: Data Cleaning and Preprocessing Automation; Handling Missing Values and Outliers; Feature Encoding and Scaling; Automated Feature Extraction and Construction; Feature Selection Methods; Dimensionality Reduction; AutoML for Tabular, Text, and Image Data.

UNIT-III ***(15 hours)***

Model Selection and Hyperparameter Optimization: Automated Model Selection; Hyperparameter Optimization Techniques: Grid Search, Random Search, Bayesian Optimization; Multi-Fidelity Optimization; Early Stopping and Resource Allocation; Neural Architecture Search (NAS): Search Spaces, Search Strategies, and Performance Estimation.

UNIT-IV ***(10 hours)***

Advanced AutoML Systems and Deployment: AutoML Frameworks: Auto-sklearn, TPOT, H2O AutoML, AutoGluon; AutoML for Deep Learning; Explainability and Interpretability in AutoML; AutoML in Production: Model Monitoring and Drift Detection; Ethical Issues, Bias, and Reproducibility; Case Studies in Healthcare, Finance, and Industry.

Practical Component: ***(30 hours)***

1. Building a baseline ML pipeline manually and comparing it with AutoML results.
2. Using AutoML frameworks for classification and regression tasks.
3. Automating feature engineering and hyperparameter tuning.
4. Implementing Bayesian optimization for hyperparameter search.
5. Applying AutoML to time-series or real-world datasets.
6. Mini-project: Designing an end-to-end AutoML pipeline for a domain-specific problem.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. F. Hutter, L. Kotthoff, and J. Vanschoren (eds.), *Automated Machine Learning: Methods, Systems, Challenges*, 1st ed., Springer, 2019.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed., MIT Press, 2016.
3. K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, 1st ed., MIT Press, 2012.

Suggested Readings:

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., Springer, 2006.
2. T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

FEDERATED LEARNING (DSE-6/GE-7)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Federated Learning	4	3	0	1	Fundamentals of Data Analytics

Course Outcomes:

At the end of this course, students will be able to:

1. Explain the FL setting and its relationship to distributed machine learning, including core actors, workflows, and design constraints.
2. Distinguish and model horizontal FL, vertical FL, and federated transfer learning scenarios and justify design choices for each.
3. Implement a horizontal FL (FedAvg-style) training workflow and evaluate convergence and performance under client heterogeneity.
4. Design and prototype vertical FL / federated transfer learning pipelines for feature-space and sample-space partitioned data.
5. Analyze incentive mechanism design goals in FL and implement a basic participation/reward scheme with measurable impact on training quality.
6. Build and report an FL application case-study (vision/language/recommendation or federated RL), including reproducible experimentation and a data-protection checklist.

Course Objectives:

1. Introduce federated learning (FL) as a privacy-aware collaborative learning paradigm and position it relative to distributed machine learning.
2. Develop the ability to classify FL problem settings as horizontal FL, vertical FL, or federated transfer learning and choose an appropriate workflow.
3. Enable students to implement core FL training loops and aggregation workflows for real datasets under practical constraints.
4. Train learners to reason about confidentiality, privacy, and data-protection drivers for FL, including legal considerations (e.g., GDPR-style requirements).
5. Develop an understanding of incentive mechanism design for federated learning participation and its impact on system performance.
6. Enable students to apply FL patterns to representative application families (vision, language, recommendation, and federated RL).

UNIT-I ***(10 hours)***

Introduction, Background, and Distributed Machine Learning Foundations for FL: motivation for federated learning and data confidentiality, privacy and data-protection drivers and constraints, background concepts and terminology for federated machine learning, distributed machine learning fundamentals relevant to FL (data locality, coordination, aggregation), high-level privacy-preserving

machine learning idea space and why FL differs from centralized training, end-to-end FL workflow overview and design constraints, overview of representative use-case motivations and responsible deployment considerations.

UNIT-II ***(12 hours)***

Horizontal Federated Learning: horizontal federated learning (HFL) setting and assumptions, client-server orchestration and round-based training, local training and global aggregation intuition, model update aggregation patterns and practical tuning levers, handling client participation variability and practical system considerations consistent with HFL, evaluation of federated models in HFL settings, summary of canonical HFL method motivations (including iterative model averaging as a representative approach).

UNIT-III ***(12 hours)***

Vertical Federated Learning and Federated Transfer Learning: vertical federated learning (VFL) setting with feature-space partitioned data and overlapping samples, coordination patterns for VFL workflows, role of privacy-preserving computation concepts at a high level as used to enable collaboration without raw data sharing, federated transfer learning (FTL) motivation for limited sample overlap and heterogeneous feature spaces, conceptual workflows for aligning knowledge across parties in FTL settings, comparative discussion of HFL vs VFL vs FTL assumptions and practical selection guidance.

UNIT-IV ***(11 hours)***

Incentive Mechanisms, Application Families, and Legal/Data-Protection Context: incentive mechanism design motivations in federated learning and how incentives affect participation and learning outcomes, high-level economic/game-theoretic framing for incentivizing collaboration, federated learning patterns for computer vision, language, and recommendation settings, introduction to federated reinforcement learning as an FL application family, selected application case patterns and deployment considerations, summary and outlook themes including open challenges, legal development on data protection with emphasis on GDPR-style principles that motivate privacy-aware ML workflows.

Practical Component: ***(30 hours)***

1. Setup and reproducibility: Python environment, experiment folder structure, logging, seeding, and a simple client–server simulation harness.
2. Distributed ML warm-up: implement centralized training vs distributed data-local training baseline to observe differences in data movement and orchestration costs.
3. Horizontal partitioning lab: create synthetic HFL client datasets (IID vs non-IID splits) and quantify heterogeneity (class distribution, sample counts).
4. Implement an HFL training loop: local training + server aggregation (FedAvg-style), run multi-round training, and plot performance across rounds.
5. Participation variability lab: simulate partial participation and dropped clients; study the impact on convergence and final accuracy.
6. Communication accounting: measure bytes/round and wall-clock time for HFL runs; compare at least two settings of local epochs and client fractions.
7. Vertical FL data modeling: construct a toy VFL dataset (two parties with different feature subsets and overlapping sample IDs) and define a collaborative objective.
8. Prototype a VFL workflow: implement a minimal secure “no-raw-feature-sharing” pipeline using feature partitioning and controlled intermediate exchanges (toy protocol sufficient for learning demonstration).
9. Federated transfer learning exercise: design a small FTL-inspired setup with limited overlap and heterogeneous feature spaces, implement a baseline transfer strategy, and document assumptions and limitations.

10. Incentive mechanism simulation: define a simple participation utility and a reward rule, simulate different incentive settings, and report the effect on participation rates and training quality.
11. Application case-study: choose one application family (vision or language or recommendation or federated RL) and implement a small-scale federated prototype with reproducible reporting.
12. Data-protection checklist: write a short compliance-oriented checklist for the mini-project (data minimization rationale, access controls, logging policy, retention, and GDPR-style considerations) and attach it to the final report.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, Federated Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool / Springer, 2019.
2. P. Kairouz, H. B. McMahan, et al., Advances and Open Problems in Federated Learning, Foundations and Trends in Machine Learning, Now Publishers / Emerald Publishing, 2021.

Suggested Readings:

1. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data: General Data Protection Regulation (GDPR), 2016.
2. H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.
3. K. Bonawitz, V. Ivanov, B. Kreuter, et al., "Practical Secure Aggregation for Federated Learning on User-Held Data," NIPS 2016 workshop on Private Multi-Party Machine Learning, 2016.

APPLIED GENERATIVE AI FOR DATA SCIENCE (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Applied Generative AI for Data Science	4	3	0	1	Fundamental of Data Analytics

Course Outcomes:

At the end of the course, students will be able to

1. Understand the fundamental concepts, architectures, and working principles of Generative Artificial Intelligence models.
2. Understand and apply generative AI techniques for the creation of images and digital paintings.
3. Understand the methods and models used for AI-based music and audio generation.
4. Understand the ethical, legal, and societal implications of Generative AI in creative and interactive applications.

Course Objectives:

- To develop a strong conceptual understanding of the fundamental principles, architectures, and learning paradigms underlying Generative Artificial Intelligence.
- To familiarize students with AI-driven techniques for the generation of images, artistic paintings, music, and interactive play environments.
- To provide practical exposure to state-of-the-art generative models and tools, enabling students to design, implement, and evaluate AI-driven content generation systems.

UNIT-I ***(11 hours)***

Introduction to Generative Artificial Intelligence (Gen AI): Introduction to Gen AI, Evolution and Historical Overview of Generative modeling, Difference between Gen AI and Discriminative Modeling, Capabilities of Generative AI and its Use Cases in Real World, Applications of Generative AI in Different Sectors: Healthcare, Finance, Marketing, Media, and Education, Role of Gen AI Models and Tools for Text, Code, Image, Audio, and Video Generation; Types of Generative Models- GANs, VAEs, Autoregressive Models, and Vector Quantized Diffusion Models.

UNIT-II ***(10 hours)***

Generative Models for Text: Language Model Basics, Building Blocks of Language Models, Transformer Architecture, Encoder and Decoder, Attention Mechanisms, Generation of Text, Models like BERT and GPT Models, Generation of Text, Autoencoding, Regression Models, Exploring ChatGPT, Prompt Engineering, Design Prompts, Revising Prompts using Reinforcement Learning, Retrieval Augmentation Generation, Multimodal LLM, and Issues of LLM.

UNIT-III ***(11 hours)***

Generation of Images: Generative Adversarial Networks, Adversarial Training Process, Nash Equilibrium, Variational Autoencoders, Stable Diffusion Models, Introduction to Transformer based Image Generation, CLIP, Visual Transformers ViT, Dall-E2, and Dall-E3. Generation of Painting, Music

and Play- Variants of GANs, Types of GANs, Cyclic GAN, Using Cyclic GAN to Generate Painting, Music Generating RNN, MuseGAN, Autonomous Agents- Deep Q Network, Actor-Critic Network.

UNIT-IV ***(10 hours)***

Gen AI Tools and Applications: Overview of GPT-based and AI-driven tools for data querying and preparation. Application of Generative AI in data cleaning, feature engineering, and exploratory analysis. Real-world use cases of Gen AI in enhancing data science workflows. Integration of Generative AI techniques within machine learning model development and deployment.

Practical Component: ***(30 hours)***

1. Implementation of autoencoders and variational autoencoders for basic generative modeling.
2. Fine-tuning and experimentation with Transformer-based language models for text generation and prompt engineering.
3. Implementation of GAN and CycleGAN models for image synthesis and style transfer.
4. Experimentation with Stable Diffusion and multimodal models such as CLIP/ViT for text-to-image tasks.
5. Music or sequential data generation using RNN/LSTM-based architectures.
6. Application of Generative AI tools in data science workflows including automated querying, feature engineering, and report generation.
7. Mini-project involving design and implementation of a Generative AI application with experimental evaluation and ethical assessment.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. A. Vemula, *Practical Generative AI for Data Science: From Theory to Real-World Applications*, 2024.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
3. A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed., O'Reilly Media, 2022.
4. D. Foster, *Generative Deep Learning*, 2nd ed., O'Reilly Media, 2022.

Suggested Readings:

1. L. Raschka, *Build a Large Language Model (From Scratch)*, Manning Publications, 2024.
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language models are few-shot learners. Advances in neural information processing systems, 33, pp.1877-1901.
3. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. and Krueger, G., 2021, July. Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748-8763).

DATA PRIVACY, SECURITY & REGULATORY COMPLIANCE (DSE-10/GE-10)

CREDIT DISTRIBUTION, ELIGIBILITY, AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Data Privacy, Security & Regulatory Compliance	4	3	0	1	Fundamentals of Data Analytics

Course Objectives:

1. To understand the foundational concepts of data privacy and security.
2. To explore privacy-preserving techniques in data science and machine learning.
3. To examine regulatory frameworks such as GDPR, HIPAA, and CCPA.
4. To develop skills to design and implement privacy-compliant data systems.

Course Outcomes:

At the end of the course, students will be able to:

1. Understand core principles of data privacy, security, and privacy-preserving techniques.
2. Analyze privacy attacks and design effective security mitigations.
3. Apply privacy-aware techniques in machine learning and distributed data systems.
4. Interpret and implement regulatory compliance requirements in data science projects.

UNIT-I ***(12 hours)***

Foundations of Data Privacy and Security: Introduction to Data Privacy, Data Protection, and Information Security, Privacy vs. Security: Concepts, Scope and Differences, Role of Data Privacy in Data Science and Analytics, Threats to Data Privacy in Modern Data-Driven Systems, Basic Cryptographic Concepts for Data Protection, Access Control Mechanisms: Authentication and Authorization, Data Anonymization, K-Anonymity and its Limitations, Differential Privacy: Definition and Motivation, Noise-based Privacy Mechanisms: Gaussian Noise Mechanism and Laplace Mechanism, Design Appropriate Privacy Measures, Differential Privacy Libraries in Pipelines.

UNIT-II ***(12 hours)***

Privacy Attacks, Threats, and Security Mitigation: Privacy Attacks: Analyzing Common Attack Vectors, Netflix Price Attack, Linkage Attack, Singling Out Attack, Strava Heap Map Attack, Membership Inference Attack, Inferring Sensitive Attributes, Attacks Against Privacy Protocols, Data Security in Privacy Protection, Data Loss Prevention Techniques, Additional Security Controls for Sensitive Data, Threat Modeling for Privacy and Security, Incident Response and Breach Management, Data Security Mitigations: Web Security Basics for Data Applications, Protecting Training Data, ML Models, and Inference APIs.

UNIT-III ***(11 hours)***

Privacy-Aware Machine Learning and Data Science: Privacy Challenges in ML, Privacy-Preserving Techniques in Data Science or ML, Differentially Private Stochastic Gradient Descent, Open-source Libraries for Privacy-Preserving ML, Designing Privacy-by-design ML Systems, Distributed Data and Privacy Challenges, Motivation for Distributed and Decentralized Data Analytics, Differential Privacy in Distributed Data, Federated Learning (FL): A Brief History, Why, When, and How to use FL, Federated

Learning Architecture and Workflows, Security and Privacy Threats in FL, Deploying FL Frameworks, Open Source Federated Libraries: Flower.

UNIT-IV ***(10 hours)***

Navigating the Legal Side of Privacy: Introduction to Legal and Regulatory Aspects of Data Privacy, General Data Protection Regulation (GDPR): Scope and Objectives, Fundamental Rights Under GDPR, Privacy-Enhancing Technologies for GDPR, The GDPR's Data Protection Impact Assessment: Agile and Iterative Risk Assessment, GDPR Compliance Challenges in Data Science and AI Systems, California Consumer Privacy Act (CCPA), Other Global Regulations: HIPPA, LGPD, and PIPL, Reading Privacy Policies and Terms of Service, Reading Data Processing Agreements, Data Governance Principles and Frameworks, Data Governance 2.0: Federated Governance, Governance challenges in anonymized and privacy-aware ML systems.

Practical Component: ***(30 hours)***

1. Implementation of basic cryptographic techniques, access control mechanisms, and anonymization methods (k-anonymity) on structured datasets.
2. Implementation of Differential Privacy mechanisms (Laplace and Gaussian noise) using open-source libraries and integration into ML pipelines.
3. Simulation and analysis of privacy attacks (linkage attack, membership inference attack) and design of mitigation strategies.
4. Implementation of privacy-preserving machine learning techniques including DP-SGD and experimentation with federated learning using open-source frameworks (e.g., Flower).
5. Threat modeling, data loss prevention strategies, and security configuration for ML model deployment and inference APIs.
6. Case study based compliance assessment project involving GDPR/CCPA/HIPAA analysis, DPIA preparation, and privacy-by-design system documentation.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. K. Jarmul, *Enhancing Privacy and Security in Data*, Foreword by N. D. Stefflbauer, 1st ed., O'Reilly Media, 2023.
2. P. Voigt and A. von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed., Springer, 2017.
3. C. Dwork and A. Roth, *The Algorithmic Foundations of Differential Privacy*, Foundations and Trends® in Theoretical Computer Science, Now Publishers, 2014.

Suggested Readings:

1. C. Kuner, L. A. Bygrave, and C. Docksey (eds.), *The EU General Data Protection Regulation (GDPR): A Commentary*, 1st ed., Oxford University Press, 2020.
2. A. Heather, B. Betsy, B. Paul, and L. Piotr, O. Ana, and S. Adam, *Building Secure and Reliable Systems*, 1st ed., O'Reilly Media, 2020.

SECURE SOFTWARE ENGINEERING (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Secure Software Engineering	4	3	0	1	Software Engineering, Object Oriented Software Engineering

Course Objectives:

1. To introduce the fundamental principles of software security, secure software development lifecycle, and threat modeling techniques.
2. To explain secure design principles, trust boundary enforcement, and identity and access management mechanisms for building secure software systems.
3. To study common classes of software vulnerabilities and the techniques used for their detection, analysis, and mitigation.
4. To expose students to secure software testing, deployment, and operational security practices, including DevSecOps, container security, and incident response.
5. To develop the ability to analyze, design, and evaluate secure software systems using industry standards, tools, and real-world case studies.

Course Outcomes:

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

1. Explain fundamental security principles, threat models, and risk assessment techniques used in secure software development.
2. Apply secure design principles and trust boundary enforcement techniques to design resilient software architectures.
3. Identify, analyze, and mitigate common software vulnerabilities using secure coding, review, and testing practices.
4. Use industry-standard open-source security tools to perform static, dynamic, and dependency security analysis.
5. Integrate security practices into software deployment and operations, including CI/CD pipelines, container platforms, and incident response.

UNIT-I ***(10 hours)***

Foundations of Security & Threat Modeling: Security principles: confidentiality, integrity, availability, and accountability; Secure Software Development Lifecycle (SSDLC) and integration with Agile/Waterfall models; Threat modeling concepts: assets, adversaries, attack surfaces, STRIDE, and attack trees; Risk assessment using DREAD/CVSS; Overview of security standards and compliance (OWASP, NIST, ISO/IEC 27001).

UNIT-II ***(13 hours)***

Secure Design Principles: Saltzer and Schroeder's secure design principles; Trust boundaries and secure data flow using DFDs; Input validation, canonicalization, output encoding, and taint tracking; Positive security models and policy enforcement; Identity and Access Management (IAM): authentication (MFA, OAuth2, OIDC) and authorization models (RBAC, ABAC).

UNIT-III ***(12 hours)***

Software Vulnerabilities and Security Testing: Common vulnerabilities: buffer overflows, injection attacks, race conditions, and dependency risks; Memory safety and secure coding practices; Static (SAST), dynamic (DAST), and software composition analysis (SCA); Secure code review techniques and fuzz testing.

UNIT-IV ***(10 hours)***

Secure Deployment and Operations: Security integration in CI/CD pipelines (Shift Left approach) Secrets management and secure configuration practices; Container and cloud security (Docker, Kubernetes basics); Logging, monitoring, incident response, and forensic readiness.

Practical Component: ***(30 hours)***

1. Threat modeling and secure design exercises using standard security frameworks
2. Static and dynamic security testing using open-source SAST and DAST tools
3. Dependency and vulnerability analysis of third-party libraries
4. Secure code review and automated fuzz testing exercises
5. Security integration in CI/CD pipelines using automated security tools

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. M. Graff and K. R. van Wyk, *Secure Coding: Principles and Practices*, 1st ed., O'Reilly Media, 2003.
2. M. Dowd, J. McDonald, and J. Schuh, *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*, 1st ed., Addison-Wesley, 2006.
3. A. Shostack, *Threat Modeling: Designing for Security*, 1st ed., Wiley, 2014.
4. T. Janca, *Alice and Bob Learn Application Security*, 1st ed., Wiley, 2020.

Suggested Readings:

1. R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed., Wiley, 2008.
2. R. C. Seacord, *Secure Coding in C and C++*, 2nd ed., Addison-Wesley, 2013.
3. OWASP Foundation, *OWASP Top 10: The Ten Most Critical Web Application Security Risks*, Latest ed., OWASP Foundation.
4. OWASP Foundation, *Application Security Verification Standard (ASVS)*, Latest ed., OWASP Foundation.

5. International Organization for Standardization (ISO), *ISO/IEC 27001: Information Security Management Systems — Requirements*, Latest ed., ISO.
6. International Organization for Standardization (ISO), *ISO/IEC 27002: Information Security Controls*, Latest ed., ISO.

BUILD, RELEASE AND DEPLOYMENT ENGINEERING (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Build, Release and Deployment Engineering	4	3	0	1	Object Oriented Software Engineering

Course Objectives:

1. To introduce modern software delivery models and build–release–deployment engineering practices.
2. To develop understanding of configuration management, version control, semantic versioning, and continuous integration.
3. To enable design and implementation of automated build, test, and deployment pipelines.
4. To familiarize students with continuous delivery ecosystems including infrastructure, environments, and dependency management.
5. To expose students to DevOps and DevSecOps practices including containerization, orchestration, Infrastructure as Code, cloud deployment, monitoring, and security integration

Course Outcomes:

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

1. Explain modern software delivery models and build–release–deployment engineering practices.
2. Apply configuration management, version control, and semantic versioning in software projects.
3. Design and implement CI/CD pipelines with automated builds, testing, and quality assurance.
4. Develop and deploy containerized applications using orchestration platforms and cloud environments.
5. Apply Infrastructure as Code principles to provision and manage scalable infrastructure.
6. Integrate DevOps and DevSecOps practices including monitoring, security controls, and vulnerability management within automated pipelines.

UNIT-I ***(11 hours)***

Foundations and Continuous Integration: Evolution of software delivery. Role of Build, Release, and Deployment Engineering in Software Lifecycle. Configuration Management. Version control systems. Semantic versioning and release numbering schemes. Continuous Integration (CI): concepts and benefits. CI pipeline stages. CI principles. CI with distributed teams. Automated testing in CI pipelines. Implementing a Testing strategy.

UNIT-II ***(12 hours)***

Deployment Pipeline: Deployment practices. Preparing to release. Implementing a deployment pipeline. Build tools. Principles and Practices of Build and Deployment Scripting. The Commit Stage. Principles

and Practices. Automated Acceptance Testing. Testing Nonfunctional requirements. Deploying and Releasing applications

UNIT-III ***(8 hours)***

Delivery Ecosystem: Managing Infrastructure and Environments. Managing Data. Managing Components and Dependencies. Managing Continuous Delivery.

UNIT-IV ***(14 hours)***

Modern DevOps Practices and Scalable Deployment: Containerization and Orchestration. Introduction to Docker and containerization. Building and managing Docker containers. Introduction to Kubernetes. Kubernetes architecture and deployment strategies. Infrastructure as Code (IaC). Design Patterns for Infrastructure. Scaling IaC with Teams. DevOps in the cloud (AWS, Azure, GCP). DevSecOps and security best practices. Secrets management (Vault, AWS Secrets Manager). Vulnerability scanning (Trivy, SonarQube)

Practical Component: ***(30 hours)***

1. Implementation of version control workflows, branching strategies, and configuration management practices using distributed VCS.
2. Design and implementation of Continuous Integration pipelines with automated build, unit testing, and quality checks.
3. Construction of deployment pipelines including commit stage, automated acceptance testing, and release automation.
4. Management of infrastructure, environments, data, and dependencies in a continuous delivery ecosystem.
5. Development and deployment of containerized applications using Docker and orchestration with Kubernetes.
6. Implementation of Infrastructure as Code and DevSecOps practices including cloud provisioning, secrets management, and vulnerability scanning.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, 1st ed., Addison-Wesley, 2010.
2. K. Morris, *Infrastructure as Code: Managing Servers in the Cloud*, 1st ed., O'Reilly Media, 2016.
3. N. Turnbull, *The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise*, 1st ed., IT Revolution Press, 2019.
4. G. Kim, J. Humble, P. Debois, and J. Willis (eds.), *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, 2nd ed., IT Revolution Press, 2021.

Suggested Readings:

1. R. Wang, *Infrastructure as Code, Patterns and Practices: With Examples in Python and Terraform*, 1st ed., Manning Publications, 2022.
2. B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, *Kubernetes: Up and Running*, 3rd ed., O'Reilly Media, 2022.

AI IN CYBERSECURITY (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
AI in Cybersecurity	4	3	0	1	Cryptography Essentials

Course Outcomes:

At the end of this course, students will be able to:

1. Understand key AI and machine learning concepts relevant to cybersecurity applications.
2. Analyze cybersecurity threats using supervised and unsupervised learning techniques.
3. Design and implement AI-driven security solutions such as intrusion detection, malware classification, and behavioral profiling.
4. Evaluate the performance of AI models in real-world cybersecurity scenarios.
5. Identify ethical, legal, and privacy concerns when deploying AI in cybersecurity systems.

Course Objectives:

1. Introduce fundamental AI and machine learning concepts as applied to cybersecurity.
2. Understand how AI techniques enhance cybersecurity practices such as intrusion detection and malware detection.
3. Explore anomaly detection, behavioral analysis, and intelligent threat hunting using AI.
4. Familiarize students with practical tools and frameworks for implementing AI-based security solutions.
5. Analyze ethical, legal, and safety considerations in AI-enabled cybersecurity systems.

UNIT-I ***(11 hours)***

Introduction to AI and Cybersecurity: Introduction to cybersecurity and AI; Cyber threat landscape and attack vectors; Fundamentals of AI, machine learning (supervised and unsupervised), and their role in cybersecurity; Python programming environment setup for AI in security; Data collection and preprocessing for security datasets.

UNIT-II ***(11 hours)***

AI Techniques for Security Analytics: Supervised learning for intrusion detection and classification (decision trees, SVMs, neural networks); Unsupervised learning for anomaly and behavioral analysis; Feature extraction and selection for security data; Time-series analysis for network traffic behaviors; Evaluation metrics for classifier performance (precision, recall, ROC).

UNIT-III ***(11 hours)***

AI for Malware and Threat Detection: AI-driven malware analysis and classification; Deep learning approaches for malware detection; Natural language processing (NLP) in phishing and spam detection; Botnet detection and profiling; Adversarial attacks on ML models and defenses; Generative models and security applications.

UNIT-IV ***(12 hours)***

Practical AI Security Systems and Ethics: AI for Security Information and Event Management (SIEM); Automated incident response with AI; Explainable AI (XAI) for cybersecurity decision support; Ethical, legal, and privacy considerations in AI use for security; Emerging topics: autonomous security systems, reinforcement learning in cyber defense, future trends.

Practical Component: ***(30 hours)***

1. Setting up AI development environments (Python, TensorFlow/PyTorch, Scikit-Learn).
2. Data preprocessing and feature engineering for network security datasets.
3. Implementing supervised classifiers for intrusion detection.
4. Unsupervised anomaly detection for network traffic.
5. Deep learning-based malware classification.
6. Use of NLP for phishing and spam detection.
7. Security evaluation of AI models under adversarial conditions.
8. Case studies and mini-projects on building AI-enhanced cybersecurity applications.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. C. Chio and D. Freeman, *Machine Learning and Security: Protecting Systems with Data and Algorithms*, 1st ed., O'Reilly Media, 2018.
2. A. Parisi, *Hands-On Artificial Intelligence for Cybersecurity: Implement smart AI systems for preventing cyber attacks and detecting threats and network anomalies*, pPackt Publishing Limited, 2019.
3. S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*, CRC Press, 2011.

Suggested Readings:

1. O. Santos, S. Salam, and H. Dahi, *AI Revolution in Networking, Cybersecurity, and Emerging Technologies*, 1st ed.,Addison-Wesley Professional, 2024.
2. S. Mahajan, M. Khurana, and V. V. Estrela, *Applying Artificial Intelligence in Cybersecurity Analytics and Cyber Threat Detection*, 1st ed. Wiley, 2024.

QUANTUM CRYPTOGRAPHY (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Quantum Cryptography	4	3	0	1	Cryptography Essentials

Course Outcomes:

At the end of this course, students will be able to:

1. Understand quantum bits (qubits) and quantum state representations.
2. Explain the principles of quantum key distribution (QKD) protocols and security.
3. Analyze the role of entanglement and uncertainty in cryptographic settings.
4. Evaluate advanced quantum cryptographic primitives beyond QKD.
5. Understand challenges, device independence, and future directions in quantum cryptography.

Course Objectives:

1. To introduce the fundamentals of quantum mechanics as they apply to information processing.
2. To study quantum communication concepts and how they differ from classical cryptography.
3. To explore quantum key distribution and security proofs in quantum settings.
4. To understand advanced quantum cryptographic protocols and their security guarantees.

UNIT-I ***(11 hours)***

Introduction to Quantum Information and Cryptography: Fundamentals of quantum mechanics for information theory: qubits, superposition, measurement, density matrices, partial trace; Basics of quantum states and quantum measurements; Classical vs. quantum information; Overview of quantum cryptography and its significance; Comparison with classical cryptography.

UNIT-II ***(11 hours)***

Quantum Key Distribution (QKD): Security of classical key distribution vs. quantum key distribution; BB84 protocol; Information reconciliation and privacy amplification; Entanglement-based QKD protocols; Device-independent QKD and security from uncertainty relations; Practical considerations and imperfections in QKD systems.

UNIT-III ***(11 hours)***

Entanglement and Advanced Quantum Protocols: Quantum entanglement and its cryptographic uses; Purifications and monogamy of entanglement; Secret sharing via quantum states; Quantum bit commitment (limitations and impossibility results); Quantum coin flipping; Oblivious transfer and position verification

UNIT-IV ***(12 hours)***

Security Analysis and Emerging Topics: Security proofs in quantum cryptography: trace distance, fidelity, von Neumann and min-entropy, uncertainty principle as a guessing game; Quantum attacks on classical cryptography (Shor's and Grover's algorithms overview); Post-quantum cryptography and quantum-resistant primitives; Future directions: quantum internet and network standards for QKD.

Practical Component ***(30 hours)***

1. Simulation and implementation of BB84 QKD protocol using quantum computing simulators (e.g., Qiskit).
2. Experiments with quantum random number generation.
3. Quantum measurement and state preparation tasks.
4. Security analysis of simple QKD implementations under noise and eavesdropping models.
5. Simulation of basic entanglement and purification processes.
6. Comparison of classical vs. quantum key distribution performance

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary ed., Cambridge University Press, 2010.
2. G. Van Assche, *Quantum Cryptography and Secret-Key Distillation*, 1st ed., Cambridge University Press, 2012.
3. M. M. Wilde, *Quantum Information Theory*, 2nd ed., Cambridge University Press, 2017.

Suggested Readings:

1. P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*, Oxford University Press, 2006.
2. T. Vidick, S. Wehner, *Introduction to Quantum Cryptography*, Cambridge University Press, 2023.

AR/VR HARDWARE SYSTEMS: CONCEPTS AND ARCHITECTURES (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
AR/VR Hardware Systems: Concepts and Architectures	4	3	0	1	Foundations of Augmented and Virtual Reality

Course Objectives:

1. To introduce students to the fundamental hardware components that enable immersive AR/VR experiences.
2. To provide an understanding of display technologies, sensing modalities, and processing architectures used in modern AR/VR devices.
3. To familiarize students with system-level challenges such as latency, power consumption, heat dissipation, and ergonomics.
4. To enable students to analyze commercial AR/VR platforms from a hardware architecture perspective.
5. To equip students with the skills required to interface sensors, displays, and compute units in AR/VR prototypes.

Course Outcomes:

At the end of this course, students will be able to:

1. Understand the hardware building blocks of AR/VR systems, including display, sensing, and processing units.
2. Analyze AR/VR device architectures with respect to performance, latency, power, and user comfort.
3. Evaluate trade-offs in AR/VR hardware design for immersive and real-time applications.
4. Design and prototype basic AR/VR hardware subsystems considering real-world constraints.

UNIT-I

Introduction to AR/VR Hardware Systems ***(12 hours)***

Overview of Augmented Reality and Virtual Reality Hardware; Evolution of AR/VR Devices; System-Level Architecture of AR/VR Platforms; Hardware vs Software Responsibilities in AR/VR Systems; Key Performance Metrics: Latency, Frame Rate, Field of View (FoV), Resolution, and Refresh Rate; Hardware Challenges in Immersive Computing.

UNIT-II

Display Technologies for AR/VR ***(13 hours)***

Human Visual System and Display Requirements; Near-Eye Display Systems; Optical See-Through vs Video See-Through Displays; Head-Mounted Displays; Various Display Technologies; Optical

Components: Lenses, Mirrors, and Combiners; Vergence–Accommodation Conflict and Mitigation Techniques.

UNIT-III

Sensing, Tracking, and Input Hardware ***(10 hours)***

Tracking Requirements in AR/VR Systems; Positional and Rotational Tracking; Inside-Out vs Outside-In Tracking; Sensors for AR/VR; Hand and Gesture Tracking Hardware; Sensor Calibration and Synchronization; Sensor Fusion at Hardware Level.

UNIT-IV

Processing Architectures and System Integration ***(10 hours)***

Processing Units for AR/VR; Edge vs On-Device Processing; Memory Systems and Bandwidth Requirements; Power Management and Thermal Design; Hardware Latency Pipeline and Optimization; Future Trends in AR/VR Hardware.

Practical Component: ***(30 hours)***

1. Study of internal hardware architecture of commercial AR/VR headsets.
2. Interfacing IMU sensors with microcontrollers for orientation tracking.
3. Camera-based positional tracking using embedded platforms.
4. Analysis of display parameters such as refresh rate, resolution, and latency.
5. Sensor fusion implementation using IMU and camera data.
6. Power consumption analysis of AR/VR hardware components.
7. Hands-on experimentation with AR/VR development kits (e.g., Oculus, HoloLens, mobile-based AR).
8. Hardware latency measurement and performance evaluation.

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. Virtual Reality by Steven M. LaValle
2. Understanding Virtual Reality by William R. Sherman & Alan B. Craig
3. Augmented Reality: Principles and Practices by Dieter Schmalstieg & Tobias Hollerer

Suggested Readings:

1. S. Aukstakalnis, Practical Augmented Reality: A Guide to the Technologies, Applications, and Human Factors for AR and VR.
2. J. Jerald, The VR Book: Human-Centered Design for Virtual Reality, ACM.
3. B. Furht, Handbook of Augmented Reality, Springer, 2011.

ETHICS, PRIVACY, AND SAFETY IN XR SYSTEMS (DSE-10/GE-10)

CREDIT DISTRIBUTION AND PREREQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Ethics, Privacy, and Safety in XR Systems	4	3	0	1	Foundations of Augmented and Virtual Reality

Course Objectives:

1. To introduce ethical challenges in XR (AR/VR/MR) systems.
2. To understand privacy risks arising from immersive technologies.
3. To study psychological, social, and physical safety concerns in XR environments.
4. To analyze regulatory, legal, and governance frameworks for XR technologies.
5. To develop responsible XR design practices aligned with industry standards.

Course Outcomes:

At the end of this course, students will be able to:

1. Analyze ethical implications of immersive XR systems.
2. Evaluate privacy risks related to biometric, behavioral, and spatial data.
3. Identify psychological, physical, and social safety threats in XR platforms.
4. Apply privacy-by-design and safety-by-design principles in XR applications.
5. Design XR systems aligned with regulatory and responsible innovation frameworks.

UNIT-I ***(11 hours)***

Foundations of Ethics in XR Systems: Introduction to immersive technologies and societal impact; Ethical frameworks: consequentialism, deontology, virtue ethics in technology design; Digital embodiment, identity, and behavioral manipulation in XR; Algorithmic bias, accessibility, and inclusivity in immersive systems; Case studies: persuasive design, dark patterns, and behavioral influence in VR.

UNIT-II ***(11 hours)***

Privacy in XR Systems: Data types in XR: biometric data, gaze tracking, motion data, spatial mapping; Privacy risks in immersive environments: inference attacks, behavioral profiling; Surveillance, bystander privacy, and spatial data governance; Privacy-by-design principles in AR/VR development; Legal and regulatory frameworks: GDPR principles, data minimization, consent in immersive systems.

UNIT-III ***(11 hours)***

Safety and Well-being in Immersive Environments: Physical safety: motion sickness, spatial disorientation, and ergonomic risks; Psychological effects: addiction, desensitization, identity dissociation, harassment in virtual spaces; Social safety: toxicity, cyberbullying, and avatar-based abuse; Content moderation and governance in multi-user XR platforms; Ethical challenges in AI-driven immersive agents and virtual influencers.

UNIT-IV ***(12 hours)***

Responsible XR Design and Governance: Safety-by-design and ethical design methodologies; Risk assessment frameworks for XR systems; Transparency, explainability, and accountability in immersive AI systems; Industry standards and policy guidelines for XR platforms; Future challenges: metaverse governance, digital twins, biometric security, and neuro-rights.

Practical Component: ***(30 hours)***

1. Ethical analysis of a popular AR/VR application using a simple ethical framework template.
2. Identification of privacy risks in XR applications using data flow diagrams.
3. Basic design of a privacy-by-design feature (e.g., consent mechanism, anonymization).
4. Safety evaluation of XR platforms using a checklist-based approach.
5. Case study discussion on real-world XR incidents (privacy/safety issues).
6. Group activity: design guidelines for responsible XR systems.
7. Mini-project: Design a conceptual XR application focusing on Ethical considerations, Privacy safeguards and User safety features

List of Experiments

Note: The course instructor will design experiments/mini-projects to complete the practical component of the course.

Essential Readings:

1. E. Langran and M. Slater, *Virtual Reality and Ethics: Ethical Issues in Immersive Technologies*, Routledge, 2020.
2. E. J. Ramirez, *The Ethics of Virtual and Augmented Reality: Building Worlds*, 1st Ed., Routledge, 2024.
3. M. Cotton, *Virtual Reality, Empathy and Ethics*, Springer (Palgrave Macmillan), 2021.
4. B. Friedman and D. G. Hendry, *Value Sensitive Design: Shaping Technology with Moral Imagination*, MIT Press, 2019.
5. EU GDPR Regulation (General Data Protection Regulation) – Selected Articles relevant to immersive systems.

Suggested Readings:

1. M. J. Grabowski, *Ethics of Virtual Reality: Technology and Experience*, Lexington Books / Bloomsbury, 2024.
2. S. K. Gupta et al. (eds.), *Exploring the Impact of Extended Reality (XR) Technologies on Promoting Environmental Sustainability*, Springer, 2025.
3. N. Diakopoulos, *Automating the News: How Algorithms Are Rewriting the Media*, Harvard University Press, 2019.
4. S. Madary and T. K. Metzinger, *Real Virtuality: A Code of Ethical Conduct for Virtual and Augmented Reality*, Frontiers in Robotics and AI
5. R. Calo, *Artificial Intelligence Policy: A Primer and Roadmap*, UC Davis Law Review (selected readings).