A.C. 27.12.2024
Appendix-39

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

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

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

Course Structure of the B.Tech. (CSE) Third Year

Semester – V

S.No.	Course Domain	Course Title	Credits*			Total Credits
			L	T	P	
1.	DSC-13	Theory of Computation	3	0	1	4
2.	DSC-14	Artificial Intelligence and Machine Learning	3	0	1	4
3.	DSC-15	Computer Networks	3	0	1	4
4.	DSE-3	Select a course from the specified list of DSE-3				4
5.	GE-5	Select a course from the specified list of GE-5				4
6.	SEC or IAPC	Choose one SEC or Internship/Apprenticeship/Project/Community Outreach (IAPC)				2
Total Credits						22

Semester – VI

S.No.	Course Domain	Course Title	Credits*			Total Credits
			L	T	P	
1.	DSC-16	Cybersecurity	3	0	1	4
2.	DSC-17	Cloud Computing	3	0	1	4
3.	DSC-18	Software Project Management	3	0	1	4
4.	DSE-4	Select a course from the specified list of DSE-4				4
5.	GE-6	Select a course from the specified list of GE-6				4
6.	SEC or IAPC	Choose one SEC or Internship/Apprenticeship/Project/Community Outreach (IAPC)				2
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 in Third Year

S.No.	Semester	DSE	Paper Title
1.	V	DSE - 3	Object Oriented Programming
2.			Computational Statistics and Probability
3.			Front-end Web Design and Development
4.			Discrete Structures
5.	VI	DSE - 4	Foundations of Data Analysis
6.			Computer Graphics
7.			Introduction to IoT
8.			Optimization Techniques
9.			Compiler Design

List of SEC/IAPC offered by the Department of Computer Science and Engineering in Third Year

S. No	Semester	SEC / IAPC	Course Title
1.	V	IAPC	Internship
2.	VI	SEC	Full Stack Application Development

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

Specializations/Minors offered by the Department of Computer Science and Engineering

Semester	GE	Minor (for ECE / EE)	Specializations / Minors (Open to CSE / ECE / EE)				
			Artificial Intelligence & Machine Learning	**Data Science**	**Software Engineering**	**Blockchain and Cybersecurity**	**Augmented Reality / Virtual Reality**
V	GE-5	Foundations of Computer Networks	Data Mining & Warehousing	Data Mining & Warehousing	Predictive Modeling	Blockchain Essentials	Human Computer Interaction
			Neural Networks	Advanced Data Analytics	Software Testing	Ethical Hacking and Advanced Cybersecurity	Game Development
VI	GE-6	Fundamentals of Software Engineering	AI for Image Processing	Health Data Analytics	Agile Software Development	Blockchain Applications	Advanced Game Development
			NLP: Techniques and Applications	Fundamentals of Time Series Analysis	Software Reliability	Cybersecurity with Blockchain	Augmented and Virtual Reality Systems Design

Detailed Syllabus of Discipline Specific Core (DSC) Courses for B.Tech. (CSE) - Semester V

THEORY OF COMPUTATION (DSC-13)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Theory of Computation	4	3L	1T	0P	-	-

Course Hours: L: 03 T: 01 P: 00

Course Outcomes

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

1. Understand fundamental models of computation and their capabilities (e.g., finite automata, pushdown automata, and Turing machines).
2. Analyze formal languages and classify them into regular, context-free, and recursively enumerable languages.
3. Construct formal proofs to show properties of languages and automata, and prove language equivalences or non-equivalences.
4. Reason about computational limits, including undecidability and complexity classes, to understand the boundaries of algorithmic solvability.

Course Objectives

1. Introduce students to the theoretical underpinnings of computation and formal language theory.
2. Enable students to model and analyze computational problems using mathematical frameworks such as automata and grammars.
3. Impart knowledge on the relationships between different classes of languages and the machines that recognize them.
4. Familiarize students with Turing machines and concepts of decidability, reducibility, and complexity.

UNIT I

Automata Theory: Introduction to automata theory: definitions and concepts - symbols, alphabets, strings, and languages, Finite automata - DFA and NFA, Minimization of DFA, Regular Expressions, Properties of regular languages: pumping lemma, closure properties.

UNIT II

Context-Free Languages and Pushdown Automata: Context-Free Grammars (CFGs): Definition, derivations, parse trees, ambiguity. Normal forms: Chomsky Normal Form (CNF) and Greibach Normal Form (GNF). Pushdown Automata (PDA): Definition and acceptance by empty stack and final state. Equivalence of PDAs and CFGs. Properties of Context-Free Languages: Closure properties, Pumping Lemma for CFLs. Applications of CFLs in compiler design and syntactic parsing.

UNIT III

Computability Theory: The Church-Turing thesis and models of computation, Turing machines: definition, design, and examples, Recursive and Recursively Enumerable Languages, Decidability: recognizable and decidable languages, The Halting problem and reductions, Non-computable functions and undecidable problems in computer science

UNIT IV

Complexity Theory: Introduction to complexity classes: P, NP, NP-complete, NP-hard, Polynomial time reductions and examples of NP-complete problems, Space complexity: PSPACE, NPSPACE, L, NL, and the Savitch's theorem, Time-space trade-offs and the class hierarchy, Complexity classes beyond NP: EXPSPACE, NEXP

Tutorial Component

1. Designing and simulating finite automata for simple problems.
2. Designing and simulating PDA.
3. Designing and simulating Turing machines.
4. Implementing algorithms for parsing regular expressions.
5. Implementing algorithms for parsing context-free grammars.
6. Solving decision problems and proving decidability.
7. Reductions among computational problems to prove NP-completeness.

List of tutorial tasks

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

Suggested Readings

1. "Introduction to the Theory of Computation" by Michael Sipser
2. "An Introduction to Formal Languages and Automata" by Peter Linz
3. "Introduction to Automata Theory, Languages, and Computation" by John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman
4. "Computational Complexity: A Modern Approach" by Sanjeev Arora and Boaz Barak

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (DSC-14)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Artificial Intelligence and Machine Learning	4	3L	0T	1P	-	Data Structures

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

Course Outcomes

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

1. Understand the basic concepts of Artificial Intelligence, including problem-solving, search strategies, and heuristic methods.
2. Implement machine learning algorithms such as regression, decision trees, and evaluation metrics, and understand their performance.
3. Apply deep learning techniques using neural networks, CNNs, and RNNs to solve problems in image and sequence data.
4. Implement and evaluate reinforcement learning algorithms, including Q-learning and policy gradients, in real-world applications.
5. Understand and apply natural language processing techniques for text analysis and classification.
6. Utilize generative models like autoencoders and GANs for tasks involving data generation and image synthesis.

Course Objectives

1. Provide students with foundational knowledge of AI, focusing on search algorithms, problem-solving techniques, and applications.
2. Introduce machine learning algorithms, with an emphasis on both supervised and unsupervised learning methods.
3. Enable students to explore deep learning and neural networks for handling complex data, including images and text.
4. Equip students with practical skills in reinforcement learning and its applications in dynamic, real-world environments.
5. Introduce students to NLP concepts, such as text preprocessing, classification, and sentiment analysis.
6. Provide an understanding of generative models and their applications in image generation and anomaly detection.

UNIT I

Introduction to Artificial Intelligence (AI)*:* Definition and Scope of AI; History and Evolution of AI; AI Applications: Natural Language Processing, Robotics, Expert Systems, and Vision Systems; Search Strategies: Uninformed Search (BFS, DFS), Informed Search (Greedy, A*), and Heuristics; Problem-Solving Techniques: Constraint Satisfaction Problems, Game Playing, and Adversarial Search.

UNIT II

Machine Learning Fundamentals*:* Introduction to Machine Learning (ML): Types of Learning – Supervised, Unsupervised, and Reinforcement Learning; Linear Regression, Logistic Regression, and Decision Trees; Model

Evaluation Metrics: Confusion Matrix, Precision, Recall, F1 Score, and ROC-AUC; Overfitting and Underfitting; Regularization Techniques: Lasso and Ridge Regression.

UNIT III

Deep Learning and Neural Networks*:* Introduction to Neural Networks; Perceptrons and Multilayer Perceptrons (MLP); Backpropagation and Gradient Descent Optimization; Convolutional Neural Networks (CNN): Architecture and Applications in Image Processing; Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM): Applications in Sequential Data and Text Processing.

UNIT IV

Advanced Topics and AI Applications: Reinforcement Learning: Markov Decision Processes, Q-Learning, and Policy Gradients; Natural Language Processing (NLP): Tokenization, Stemming, Lemmatization, and Text Classification; Generative Models: Autoencoders and Generative Adversarial Networks (GANs); Ethical Implications of AI: Bias, Fairness, and Accountability; Case Studies: AI in Healthcare, Autonomous Vehicles, and Financial Systems.

Practical Component

1. Implementing a basic AI agent for problem-solving using search algorithms like BFS and DFS.
2. Developing machine learning models for regression and classification using popular algorithms like Decision Trees, k-NN, and SVM.
3. Implementing and training a simple neural network using Python and TensorFlow or PyTorch.
4. Designing and testing deep learning models, including CNNs, for image classification tasks.
5. Implementing reinforcement learning algorithms like Q-learning and testing them in a simulated environment.
6. Developing a natural language processing pipeline for text classification, sentiment analysis, or spam detection.
7. 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.

Suggested Readings

1. “Artificial Intelligence: A Modern Approach” by Stuart Russell and Peter Norvig (3rd Edition)
2. “Pattern Recognition and Machine Learning” by Christopher Bishop
3. “Deep Learning” by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
4. “Machine Learning: A Probabilistic Perspective” by Kevin P. Murphy
5. “Python Machine Learning” by Sebastian Raschka
6. “Reinforcement Learning: An Introduction” by Richard S. Sutton and Andrew G. Barto
7. “Natural Language Processing with Python” by Steven Bird, Ewan Klein, and Edward Loper

COMPUTER NETWORKS (DSC-15)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Computer Networks	4	3L	0T	1P	-	Operating System

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

Course Outcomes

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

1. Understand and explain the various layers of computer networks and the protocols used in each layer.
2. Analyze network performance using key metrics and troubleshoot networking issues.
3. Demonstrate the ability to design and implement network solutions.
4. Identify and describe the latest trends in computer networking, including software-defined networking (SDN), Internet of Things (IoT), and 5G technologies.
5. Apply networking concepts through practical experiments in simulated environments**.**

Course Objectives

1. To introduce the fundamental concepts of computer networks and communication systems.
2. To provide an understanding of the network architecture, protocols, and layers.
3. To examine various types of networks and their components, including physical media, protocols, and network devices.
4. To familiarize students with current trends and technologies in networking such as SDN, IoT, and 5G.
5. To enhance problem-solving and analytical skills through hands-on experiments in network simulation and implementation.

UNIT I

Introduction to Computer Networks: Overview of computer networks and their importance, Network topologies, types, and models, OSI Reference Model, TCP/IP reference model, Addressing, Network Devices, Introduction to software-defined networking (SDN)

Physical layer: transmission media, signaling, modulation and demodulation, switching

UNIT II

Data link layer: design issues, framing, error detection and correction, Elementary data link protocols, Sliding window protocols, MAC Protocols

Network Layer: Network layer design issues, Shortest path routing, Link-State Routing algorithm, Distance-Vector Routing algorithm; Logical Addressing and Internet Protocols: IPv4 and IPv6, Address Mapping, Error Reporting ICMP, Network devices: routers, switches, firewalls

UNIT III

Transport Layer: Process to process Delivery, Elements of Transport Protocol, Congestion control, Internet Transport Protocol: UDP and TCP

Application layer protocols: HTTP, DNS, FTP, SMTP, Electronic Mail

UNIT IV

Network Security: Security Services, Communication security, email security, web security

Emerging Technologies: SDN architecture, and applications, Wireless and mobile networks: WiFi, LTE, 5G, Networking IoT devices and protocols like MQTT and CoAP, Network virtualization and cloud computing, Blockchain in networking

Practical Component

1. Setting up a small network and configuring network devices.
2. Analyzing network traffic using packet sniffing tools.
3. Simulating routing protocols using network simulation software (Simulate and visualize the working of RIP, OSPF, and BGP in a network environment).
4. Setting up and configuring a simple LAN using switches and routers
5. Simulate the three-way handshake process of TCP and implement error control and flow control.
6. Implementing a basic secure network communication protocol.
7. Overview and configuration of 5G network components in a simulator.

List of Experiments

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

Suggested Readings

1. "Data Communications and Networking" by Behrouz A. Forouzan
2. "Computer Networks" by Andrew S. Tanenbaum and David J. Wetherall (5th Edition, Pearson)
3. "Computer Networking: A Top-Down Approach" by James Kurose and Keith Ross (6th Edition, Pearson)
4. "Data and Computer Communications" by William Stallings (10th Edition, Pearson)
5. "Network Security Essentials: Applications and Standards" by William Stallings

Detailed Syllabus of Discipline Specific Elective (DSE) courses for B.Tech. (CSE) – Semester V (DSE-3)

OBJECT-ORIENTED PROGRAMMING (DSE-3)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Object-Oriented Programming	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand and apply basic OOP concepts.
2. Implement advanced OOP concepts such as abstract classes, interfaces, exception handling, generic programming, inner classes, lambda expressions, reflection, and annotations in software development projects.
3. Apply creational, structural, and behavioral design patterns in real-world software development scenarios.
4. Use OOP principles to design and develop small to medium-scale software projects.

Course Objectives

1. Provide students with a thorough understanding of the basic principles of object-oriented programming, including classes, objects, inheritance, polymorphism, encapsulation, and the use of constructors and destructors.
2. Equip students with advanced OOP concepts such as abstract classes, interfaces, exception handling, generic programming, inner classes, lambda expressions, reflection, and annotations.
3. Introduce students to various design patterns, their real-world applications, and the impact of anti-patterns, emphasizing creational, structural, and behavioral patterns.
4. Enable students to apply OOP principles in software engineering using languages such as Java and C++, exploring advanced features and best practices, and analyzing case studies of large-scale OOP-based software systems.

UNIT I

OOP Concepts: Fundamentals of OOP: Classes, Objects, Methods, Inheritance: Single, Multiple, Multilevel, Hierarchical, Polymorphism: Compile-time and Runtime, Encapsulation and Data Hiding, Constructors and Destructors.

UNIT II

Advanced OOP Concepts: Abstract Classes and Interfaces, Exception Handling and Assertions, Generic Programming, Inner Classes and Lambda Expressions, Reflection and Annotations.

UNIT III

Design Patterns: Creational Patterns: Singleton, Factory, Builder, Prototype, Structural Patterns: Adapter, Composite, Proxy, Flyweight, Behavioral Patterns: Strategy, Command, Observer, Iterator, Real-world applications of design patterns, Anti-patterns and their impact.

UNIT IV

OOP in Software Development: OOP principles in software engineering, OOP in Java: Advanced features and libraries, OOP in C++: STL, Templates, Operator Overloading, Case studies: OOP in large-scale software systems, Best practices in OOP

Practical Component

1. Development of small to medium scale OOP projects
2. Implementation of various design patterns in projects
3. Object-oriented analysis and design exercises
4. Case studies: Analyzing OOP-based software

Suggested Readings

1. "Object-Oriented Analysis and Design with Applications" by Grady Booch, Robert A. Maksimchuk, Michael W. Engle
2. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

List of Experiments

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

COMPUTATIONAL STATISTICS AND PROBABILITY (DSE-3)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Computational Statistics and Probability	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand and apply probability theory.
2. Analyze random variables and distributions.
3. Execute point and interval estimation, perform linear and multiple regression analysis, conduct ANOVA and non-parametric tests, and analyze time series data for forecasting purposes.
4. Apply Monte Carlo methods and simulations, understand and use queueing theory (M/M/1, M/M/c), model decision processes using Markov chains, analyze system reliability and survival, and utilize statistical learning and data mining techniques.

Course Objectives

1. Provide students with a thorough understanding of the basic concepts of probability, including conditional probability, random variables, and key probability distributions, along with expectations and moment generating functions.
2. Equip students with knowledge of joint, marginal, and conditional distributions, functions of random variables, the Central Limit Theorem, sampling distributions, and hypothesis testing methods.
3. Enable students to perform point and interval estimation, regression analysis, ANOVA, non-parametric tests, and time series analysis, and to understand their applications in data analysis.
4. Familiarize students with practical applications of computational statistics in computer science, including Monte Carlo simulations, queueing theory, Markov chains, reliability theory, and statistical learning.

UNIT I

Probability Theory: Basic concepts of probability, Conditional probability and Bayes' theorem, Discrete and continuous random variables, Probability distributions: Binomial, Poisson, Normal, Expectation, Variance, and Moment generating functions.

UNIT II

Random Variables and Distributions: Joint, Marginal, and Conditional distributions, Functions of random variables, Central Limit Theorem and its implications, Sampling distributions and estimators, Hypothesis testing: Z-test, T-test, Chi-square test.

UNIT III

Statistical Methods: Point and Interval estimation, Regression analysis: Linear and Multiple regression, Analysis of Variance (ANOVA), Non-parametric tests, Time series analysis and forecasting.

UNIT IV

Applications in Computer Science: Monte Carlo methods and simulations, Queueing theory: M/M/1, M/M/c queues, Markov chains and decision processes, Reliability theory and survival analysis, Statistical learning and data mining.

Practical Component

1. Statistical analysis using R or Python
2. Data visualization techniques
3. Simulations for probabilistic models
4. Case studies: Application of statistics in computer science

Suggested Readings

1. "Probability and Statistics for Engineering and the Sciences" by Jay L. Devore
2. "Introduction to Probability and Statistics for Engineers and Scientists" by Sheldon M. Ross

List of Experiments

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

FRONT-END WEB DESIGN AND DEVELOPMENT (DSE-3)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Front-end Web Design and Development	4	0L	0T	4P	-	-

Course Hours: L: 00 T: 00 P: 04

Course Outcomes

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

1. Create and style web pages using HTML and CSS.
2. Develop interactive web pages using JavaScript.
3. Apply responsive design principles using media queries and responsive frameworks. Develop mobile-first designs, ensure accessibility and adherence to web standards, and achieve cross-browser compatibility.
4. Build applications using front-end frameworks and libraries.

Course Objectives

1. Provide students with a comprehensive understanding of HTML5 and CSS3, covering essential elements, attributes, semantic HTML, selectors, the box model, and layout techniques including Flexbox and Grid Layout. Additionally, introduce responsive web design principles and CSS preprocessors like SASS and LESS.
2. Equip students with the basics of JavaScript, including syntax, variables, and data types, as well as advanced concepts like ES6 features. Enable students to manipulate the DOM, handle events, and make asynchronous web requests using AJAX and the Fetch API.
3. Teach students the principles of responsive web design, including media queries and responsive frameworks, the mobile-first design approach, accessibility standards, and cross-browser compatibility.
4. Familiarize students with popular front-end frameworks and libraries, including React.js, Vue.js, and Angular. Teach the basics of building single-page applications (SPAs) and utilizing UI design frameworks like Bootstrap.

UNIT I

HTML and CSS: HTML5: Elements, Attributes, Semantic HTML, CSS3: Selectors, Box Model, Flexbox, Grid Layout, Responsive Web Design Principles, CSS Preprocessors: SASS, LESS.

UNIT II

JavaScript and DOM Manipulation: JavaScript Basics: Syntax, Variables, Data Types, DOM Manipulation: Selectors, Events, Event Listeners, ES6 Features: Arrow Functions, Promises, Modules, AJAX and Fetch API for Asynchronous Web Requests.

UNIT III

Responsive Design: Media Queries and Responsive Frameworks, Mobile-First Design Approach, Accessibility and Web Standards, Cross-Browser Compatibility.

UNIT IV

Frameworks and Libraries: Introduction to React.js: Components, State, Props, Working with Bootstrap for UI Design, Introduction to Vue.js and Angular, Building Single Page Applications (SPAs).

Practical Component

1. Designing and developing a responsive website
2. Implementing interactive features using JavaScript and frameworks
3. Project: Develop a complete front-end for a web application

Suggested Readings

1. "HTML and CSS: Design and Build Websites" by Jon Duckett
2. "Eloquent JavaScript: A Modern Introduction to Programming" by Marijn Haverbeke
3. "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" by Kirupa Chinnathambi

List of Experiments

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

DISCRETE STRUCTURES (DSE-3)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Discrete Structures	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand and apply concepts of sets, relations, and functions.
2. Apply basic counting techniques in combinatorial problems, understand and work with different types of graphs and their representations, perform graph traversals, and analyze properties of trees and planar graphs. Utilize graph coloring techniques and network models in practical scenarios.
3. Understand and apply the properties of groups, subgroups, rings, and fields. Work with polynomials and understand their applications. Analyze and utilize lattices and Boolean algebra in various contexts.
4. Analyze and construct propositional and predicate logic statements, use logical connectives and truth tables to determine tautologies and contradictions, understand logical equivalence and normal forms. Simplify Boolean functions and apply them in digital logic design.
5. Participate in hands-on exercises and projects that involve set theory, combinatorics, graph theory, algebraic structures, and logic.

Course Objectives

1. Provide students with a comprehensive understanding of set theory, relations, and functions, including their definitions, properties, and operations.
2. Equip students with basic counting techniques and the fundamentals of graph theory, including graph types, representations, traversals, trees, planar graphs, graph coloring, and network models.
3. Introduce students to algebraic structures such as groups, rings, and fields, along with their properties and applications.
4. Familiarize students with propositional and predicate logic, logical connectives, truth tables, tautologies, contradictions, logical equivalence, and normal forms.

UNIT I

Sets, Relations, and Functions: Set Theory: Definitions, Operations, Venn Diagrams, Relations: Types, Properties, Closure, Functions: Types, Composition, Inverse, Counting: Pigeonhole Principle, Permutations and Combinations, Mathematical Induction and Recursion.

UNIT II

Combinatorics and Graph Theory: Basic Counting Techniques, Graphs: Types, Representations, Traversals, Trees: Properties, Binary Trees, Tree Traversals, Planar Graphs, Graph Coloring, Network Models and Algorithms.

UNIT III

Algebraic Structures: Groups: Definitions, Properties, Subgroups, Rings and Fields: Definitions, Properties, Polynomials and their Applications, Lattices and Boolean Algebra.

UNIT IV

Logic and Boolean Algebra: Propositional and Predicate Logic, Logical Connectives, Truth Tables, Tautologies, and Contradictions, Logical Equivalence, Normal Forms, Boolean Functions, Simplification of Boolean Functions, Applications in Digital Logic Design.

Practical Component

1. Problem-solving sessions using combinatorial techniques
2. Implementing graph algorithms
3. Boolean function simplification using software tools

Suggested Readings

1. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Concrete Mathematics: A Foundation for Computer Science" by Ronald L. Graham, Donald E. Knuth, and Oren Patashnik
3. "Discrete Mathematical Structures with Applications to Computer Science" by J. P. Trembly& P. Manohar
4. "Elements of Discrete Mathematics" by C. L. Liu

List of Experiments

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

Detailed Syllabus of Discipline Specific Core (DSC) Courses for B.Tech. (CSE) - Semester VI

CYBERSECURITY (DSC-16)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Cybersecurity	4	3L	0T	1P	-	Operating Systems, Computer Networks

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

Course Outcomes

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

1. Understand the principles of cybersecurity and identify various types of cyber threats.
2. Apply encryption techniques and cryptographic protocols to secure communication channels.
3. Design and implement secure system architectures to mitigate risks.
4. Utilize tools and techniques to perform vulnerability assessments and penetration testing.
5. Analyze and respond to security incidents effectively to protect digital assets.

Course Objectives

1. Provide a comprehensive understanding of fundamental cybersecurity principles, threats, and countermeasures.
2. Equip students with the skills to analyze and design secure systems and networks.
3. Familiarize students with encryption, authentication, and cryptographic protocols.
4. Develop critical thinking and problem-solving skills in securing digital environments against cyber threats.
5. Enable hands-on experience with security tools and frameworks for assessing and improving cybersecurity postures.

UNIT I

Introduction to Cybersecurity: Definition and importance of cybersecurity, core principles (confidentiality, integrity, availability), and frameworks like NIST and ISO 27001. Types of cyber threats: phishing, malware, ransomware, and vulnerabilities. Basic cryptographic techniques: encryption, hashing, and digital signatures. Overview of compliance and regulatory frameworks such as GDPR and HIPAA. Importance of security policies in organizational contexts.

UNIT II

Network Security: Network security mechanisms: firewalls, intrusion detection/prevention systems (IDS/IPS), and virtual private networks (VPNs). Secure network design principles: subnetting, VLANs, and demilitarized zones (DMZ). Vulnerabilities in network protocols (e.g., TCP/IP) and secure protocols like HTTPS and SSH. Wireless security principles, including WPA3 and defenses against common wireless attack vectors.

UNIT III

System and Application Security: Operating system security: user access control, privilege escalation, and securing Windows/Linux systems. Secure software development lifecycle (SDLC) practices to prevent vulnerabilities like SQL injection and buffer overflows. Web application security using OWASP Top 10 guidelines. Endpoint security management: anti-malware solutions, patch management, and best practices for safeguarding devices.

UNIT IV

Advanced Topics in Cybersecurity: Incident response lifecycle: detection, containment, eradication, recovery, and forensics. Security challenges in emerging technologies, including cloud computing, Internet of Things (IoT), and blockchain. Ethical hacking and penetration testing techniques using tools like Metasploit and Nmap. Future trends in cybersecurity: artificial intelligence in threat detection and the implications of quantum computing on cryptography.

Practical Component

1. Configuring and managing firewalls to enforce network security policies.
2. Performing network scanning and penetration testing using tools like Nmap and Metasploit.
3. Implementing Intrusion Detection Systems (IDS) to monitor network traffic for suspicious activities.
4. Encrypting and decrypting sensitive data using algorithms such as AES or RSA.
5. Conducting vulnerability assessments of web applications and fixing identified issues.
6. Simulating and responding to Distributed Denial of Service (DDoS) attacks.
7. Developing and analyzing secure authentication mechanisms using multi-factor authentication (MFA).

Suggested Readings

1. "Computer Security: Principles and Practice" by William Stallings and Lawrie Brown (Pearson)
2. "Cybersecurity Essentials" by Charles J. Brooks, Christopher Grow, Philip Craig, and Donald Short (Wiley)
3. "The Art of Deception" by Kevin Mitnick and William L. Simon (Wiley)
4. "Practical Malware Analysis" by Michael Sikorski and Andrew Honig (No Starch Press)

List of Experiments

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

CLOUD COMPUTING (DSC-17)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Cloud Computing	4	3L	0T	1P	-	Computer Networks

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

Course Outcomes

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

1. Understand fundamental concepts of cloud computing, including service and deployment models, and evaluate the benefits and limitations of cloud-based solutions.
2. Explore and manage virtualized resources, storage services, and networking components in the cloud, as well as identify and implement appropriate security measures.
3. Design, develop, and deploy cloud-native applications using modern tools, containerization, and serverless architectures.
4. Leverage advanced cloud services for big data analytics, machine learning, blockchain, and emerging cloud-based innovations.

Course Objectives

1. Introduce the fundamental principles, models, and services of cloud computing and its evolving ecosystem.
2. Provide hands-on knowledge of infrastructure management, virtualization, cloud networking, and security best practices.
3. Enable students to design, implement, and manage cloud-native applications through containers, orchestration, and serverless paradigms.
4. Familiarize students with advanced cloud services, including big data analytics, machine learning, and blockchain on the cloud.
5. Foster an understanding of cutting-edge and future cloud technologies, including quantum and edge computing, and their impact on modern IT landscapes.

UNIT I

Cloud Computing Fundamentals: Introduction to Cloud Computing: Definition, characteristics, evolution. Service Models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). Deployment Models: Public, Private, Hybrid, and Community Clouds. Economic and Organizational Considerations: Cost-benefit analysis, scalability, agility. Challenges and Risks: Vendor lock-in, data privacy and security, regulatory compliance, SLA issues.

UNIT II

Cloud Infrastructure and Management: Virtualization Concepts: Hypervisors, VMs, containers, resource abstraction and pooling. Cloud Storage Services and Databases: Object storage, block storage, file storage, NoSQL databases, and distributed storage considerations. Cloud Networking: Virtual Private Clouds (VPCs), Virtual Private Networks (VPNs), Content Delivery Networks (CDN), load balancing. Cloud Security and Compliance: Identity and

Access Management (IAM), data encryption, ISO / IEC security standards for cloud, compliance frameworks (GDPR, HIPAA). Resource Management: Autoscaling, monitoring, logging, cost optimization tools.

UNIT III

Developing and Deploying Cloud Applications: Cloud-Native Application Design: 12-factor apps, microservices architecture, cloud design patterns. Development and Deployment Models: CI/CD pipelines, DevOps and DevSecOps practices in the cloud. Containers and Orchestration: Docker fundamentals, Kubernetes architecture and resource management, Helm charts. Serverless Architectures: Functions as a Service (FaaS).

UNIT IV

Advanced Cloud Technologies and Trends: Big Data Analytics in the Cloud: Data lakes, data warehouses, scalable analytics platforms (e.g., AWS EMR, Google BigQuery). Machine Learning and AI Services: Pre-built ML APIs, custom model training and deployment, AutoML, edge inference.

Practical Component

1. Configuring and deploying a simple cloud-based application.
2. Implementing a containerized application with Docker and deploying it using Kubernetes.
3. Utilizing cloud services for storage, computing, and networking in a project.
4. Developing a serverless application that integrates with cloud-based AI services.

Suggested Readings

1. "Cloud Computing: Concepts, Technology & Architecture" by Thomas Erl, Ricardo Puttini, and Zaigham Mahmood
2. "AWS Documentation" - https://docs.aws.amazon.com/
3. "Kubernetes: Up and Running" by Kelsey Hightower, Brendan Burns, and Joe Beda
4. "Mastering Cloud Computing: Foundations and Applications Programming" by Rajkumar Buyya, Christian Vecchiola, S.Thamarai Selvi.
5. "Cloud Computing Design Patterns" by Thomas Erl, Robert Cope, Amin Naserpour.

List of Experiments

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

SOFTWARE PROJECT MANAGEMENT (DSC-18)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Software Project Management	4	3L	0T	1P	-	Software Engineering

Course Hours: L: 03 T: 01 P: 00

Course Outcomes

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

1. Understand the software project management process and lifecycle.
2. Apply project estimation techniques and create project plans.
3. Manage project schedules, resources, and risks.
4. Use modern tools and methodologies for software project management, including Agile and Scrum.
5. Communicate effectively and lead software development teams.

Course Objectives

1. To introduce students to the concepts, processes, and tools of software project management.
2. To develop an understanding of how to plan, monitor, and control software projects.
3. To learn about risk management, quality management, and resource allocation in a software project.
4. To explore modern software project management practices, including agile methodologies.

UNIT I

Introduction to Software Project Management: Fundamentals of project management, Software Project life cycle and phases, Role of a project manager, Factors leading to project success, Project failure reasons, Key performance indicators (KPIs), Introduction to project management methodologies (Agile, Waterfall, PRINCE2)

UNIT II

Software Project Planning and Estimation: Scope management and work breakdown structure (WBS); Estimation Techniques: Function Point Analysis (FPA), COCOMO model, Use Case Point; Project Scheduling , Gantt charts, PERT charts, Critical Path Method (CPM), Resource allocation and budgeting, Risk management strategies

UNIT III

Project Monitoring and Control: Leadership and team management skills, Communication strategies within a project team, Monitoring project progress and performance metrics, Quality assurance and control in project management, Project closure and post-mortem analysis

UNIT IV

Advanced Topics in Software Project Management: Agile Methodologies, Scrum Framework, Software Project Management Tools like JIRA, Trello, Microsoft Project, OpenProject, Taiga, ProjectLibre, OrangeScrum,Redmine, Remote project management, Virtual teams, Collaborative tools, Impact of AI and Machine Learning in project management

Tutorial Component

1. Create a project charter for a hypothetical software project using *OpenProject* or similar software.
2. Perform stakeholder analysis and create a work breakdown structure (WBS) using *Taiga* or similar software.
3. Use *OrangeScrum* (open-source agile project management software) or similar software to estimate project costs and resources.
4. Create a project schedule using Gantt charts for a software project using *ProjectLibre* or similar software.
5. Perform risk analysis for a software project and develop a risk management plan using *OpenProject.*
6. Use project management software *Taiga* or similar software to monitor project progress and track milestones for an Agile project, integrating KPIs and Earned Value Analysis (EVA).
7. Simulate a project change request process and perform configuration management using version control tools (e.g., Git).
8. Create a project control plan for managing changes using the *Redmine* tool, including a change control board setup.
9. Set up an Agile Scrum project using *Taiga*, define roles (Product Owner, Scrum Master, Development Team), create product backlog, and manage sprints.
10. Use *Redmine* to manage an Agile project, set up sprint backlogs, track tasks, and manage communication with stakeholders.

List of tutorial tasks

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

Suggested Readings

1. Software Project Management by Bob Hughes, Mike Cotterell (7th Edition, McGraw-Hill)
2. Software Engineering Project Management by Richard H. Thayer (Wiley)
3. Agile Project Management with Scrum by Ken Schwaber (Microsoft Press)
4. The Art of Project Management by Scott Berkun (O'Reilly Media)
5. Modern Software Engineering by David L. Parnas (Springer)

Detailed Syllabus of Discipline Specific Elective (DSE) courses for B.Tech. (CSE) – Semester VI (DSE-4)

FOUNDATIONS OF DATA ANALYSIS (DSE-4)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Foundations of Data Analysis	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the significance of Data Analytics.
2. Apply descriptive and inferential statistics.
3. Perform regression analysis (linear and logistic) and apply classification techniques.
4. Conduct time series analysis and forecasting, and understand the fundamentals of model validation and selection.
5. Participate in hands-on projects that involve data collection, cleaning, analysis, and interpretation.

Course Objectives

1. Provide students with a comprehensive understanding of data analytics, its importance across various domains, and the different types of data.
2. Equip students with knowledge of descriptive and inferential statistics, exploratory data analysis (EDA), and data visualization techniques.
3. Introduce students to the basics of predictive analytics, including regression analysis (linear and logistic), classification techniques, time series analysis, forecasting, and model validation and selection.
4. Provide hands-on experience with data analytics tools such as R and Python, and their libraries.

Unit 1: Introduction to Data Analytics

Overview of data analytics and its significance in various domains, Types of data: structured, unstructured, semi-structured, Data analytics lifecycle: data collection, cleaning, analysis, interpretation, Case studies demonstrating the impact of data analytics.

Unit 2: Data Analysis Techniques

Descriptive statistics: measures of central tendency and variability, Inferential statistics: hypothesis testing, confidence intervals, Introduction to exploratory data analysis (EDA), Data visualization techniques and tools.

Unit 3: Introduction to Predictive Analytics

Basics of regression analysis: linear and logistic regression, Overview of classification techniques, Time series analysis and forecasting basics, Introduction to model validation and selection.

Unit 4: Data Analytics Tools and Applications

Introduction to data analytics software: R, Python and its libraries, Data analytics in business decision-making, Ethical considerations in data analytics, Emerging trends in data analytics.

Practical Component

1. Performing exploratory data analysis using a dataset and presenting findings.
2. Implementing linear regression and logistic regression models on real-world data.
3. Creating data visualizations to interpret and communicate data insights.
4. Conducting a basic time series analysis project.

Suggested Readings

1. "Data Science for Business" by Foster Provost and Tom Fawcett
2. "Python for Data Analysis" by Wes McKinney
3. "Naked Statistics: Stripping the Dread from the Data" by Charles Wheelan

List of Experiments

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

COMPUTER GRAPHICS (DSE-4)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Computer Graphics	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand and apply foundational computer graphics concepts.
2. Create and manipulate 2D and basic 3D graphics.
3. Develop and render complex 3D models.
4. Conceptualize and develop a complex 3D scene, integrating advanced graphics techniques and user interactions.

Course Objectives

1. Provide students with a comprehensive understanding of the basic principles and applications of computer graphics, including graphics systems, hardware, coordinate systems, and transformations.
2. Equip students with knowledge and skills in basic drawing algorithms, 2D transformations, clipping and windowing techniques, color models, filling algorithms, and an introduction to 3D graphics including transformations and projections.
3. Enable students to create complex 3D models, apply texture mapping and material design, utilize lighting and shading models, implement hidden surface removal algorithms, and understand real-time rendering techniques including shader programming and GPU utilization.
4. Familiarize students with keyframe animation, motion capture, rigging, skeletal animation, facial animation, lip syncing, physics-based animation, and procedural animation techniques.

Unit 1: Foundations of Computer Graphics

Introduction to computer graphics and applications, Graphics systems and hardware, Coordinate systems and transformations, Introduction to OpenGL and graphics libraries.

Unit 2: 2D and Basic 3D Graphics

Basic drawing algorithms: lines, circles, and polygons, 2D transformations and animations, Clipping and windowing techniques, Color models and filling algorithms, Introduction to 3D graphics, 3D transformations and projections.

Unit 3: Advanced 3D Modeling and Rendering

Complex 3D Modeling Techniques, Texture Mapping and Material Design, Lighting and Shading Models, Hidden surface removal algorithms, Real-Time Rendering: Algorithms, Shader Programming, and GPU Utilization.

Unit 4: Animation, Rigging, and Advanced Techniques

Keyframe Animation and Motion Capture, Rigging and Skeletal Animation, Facial Animation and Lip Syncing, Physics-Based Animation and Procedural Animation Techniques, Advanced Graphics Project: Conceptualizing a Complex 3D Scene, Integrating Techniques, Implementing Interactions, and Project Presentation.

Practical Component

1. Implementing 2D and 3D transformations and animations.
2. Developing a simple 3D model and applying texture mapping.
3. Creating basic and advanced shaders for real-time rendering.
4. Conducting a project on complex 3D scene development.
5. Applying animation and rigging techniques to a 3D character.

Suggested Readings

1. "Computer Graphics: Principles and Practice" by John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, and Kurt Akeley
2. "Fundamentals of Computer Graphics" by Peter Shirley, Steve Marschner
3. "Real-Time Rendering" by Tomas Akenine-Möller, Eric Haines, Naty Hoffman

List of Experiments

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

INTRODUCTION TO IOT (DSE-4)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Introduction to IoT	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Demonstrate a thorough understanding of the basic components of IoT, its communication models, and networking protocols.
2. Set up and configure IoT development boards like Raspberry Pi, Arduino, and ESP8266.
3. Apply various IoT connectivity technologies such as Wi-Fi, Bluetooth, Zigbee, and LoRa.
4. Understand the application of IoT in different domains including smart homes, industrial IoT, healthcare, and smart cities.

Course Objectives

1. Provide students with a comprehensive understanding of the basics of IoT.
2. Equip students with practical knowledge of IoT development boards, sensors, actuators, and programming languages.
3. Enable students to understand and apply various IoT connectivity technologies, data transmission methods, data protocols, and secure communication techniques.
4. Familiarize students with different applications of IoT in various domains such as smart homes, industrial IoT, healthcare, and smart cities.

Unit 1: IoT Fundamentals

Overview of IoT: Definition, History, and Applications, Basic Components of IoT: Sensors, Actuators, and Controllers, IoT Communication Models: Device-to-Device, Device-to-Cloud, and Device-to-Gateway, IoT Networking Protocols: MQTT, CoAP, HTTP, and WebSocket.

Unit 2: IoT Hardware and Software

Introduction to IoT Development Boards: Raspberry Pi, Arduino, ESP8266, etc., Setting up a basic IoT environment using Raspberry Pi/Arduino, Sensors and Actuators: Types and Use Cases, IoT Programming: Basics of Python and C/C++ for IoT, Interfacing Sensors with IoT Boards, Building IoT applications to collect and display sensor data.

Unit 3: IoT Communication and Networking

IoT Connectivity Technologies: Wi-Fi, Bluetooth, Zigbee, and LoRa, Data Transmission in IoT: Cloud Services and Data Storage, IoT Data Protocols: JSON, XML Secure Communication in IoT: Encryption and Authentication, Interfacing IoT systems with cloud.

Unit 4: IoT Applications and Case Studies

Smart Home and Building Automation, Industrial IoT (IIoT) and Smart Manufacturing, Healthcare and Wearable IoT Devices, Smart Cities and Infrastructure, Case studies of end-to-end IoT applications (e.g., smart home system, health monitoring device).

Practical Component

1. Weekly Lab Sessions: Hands-on experience with IoT development boards and sensors.
2. Mini Projects: Regular small projects to implement different IoT functionalities.
3. Final Project: A comprehensive project to design and develop a complete IoT system.
4. Case Study Analysis: Reviewing real-world IoT applications and their implementations.

Suggested Readings

1. "Internet of Things: A Hands-On Approach" by Arshdeep Bahga and Vijay Madisetti
2. "The Internet of Things: Enabling Technologies, Platforms, and Use Cases" by Pethuru Raj and Anupama C. Raman
3. "Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry" by Maciej Kranz

List of Experiments

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

OPTIMIZATION TECHNIQUES (DSE-4)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Optimization Techniques	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Demonstrate a thorough understanding of optimization principles, types, and mathematical foundations.
2. Apply linear and non-linear programming techniques.
3. Solve discrete optimization problems using heuristic methods.
4. Utilize advanced optimization techniques in real-world applications.

Course Objectives

1. Provide students with a comprehensive understanding of the basics of optimization, including its definition, types, importance, and mathematical foundations.
2. Equip students with knowledge of linear and non-linear programming methods, including the Simplex method, duality, sensitivity analysis, gradient descent, Newton's method, and constrained optimization techniques.
3. Introduce students to discrete optimization techniques such as integer programming, branch and bound, cutting planes, and combinatorial optimization problems.

Unit 1: Introduction to Optimization

Basics of Optimization: Definition, Types, and Importance, Mathematical Foundations: Linear Algebra and Calculus Review, Formulating Optimization Problems: Objective Function, Constraints, Types of Optimization Problems: Linear, Non-linear, Integer, and Combinatorial, Solving basic optimization problems using Python (SciPy).

Unit 2: Linear and Non-Linear Programming

Linear Programming: Simplex Method, Duality, Sensitivity Analysis, Non-Linear Programming: Gradient Descent, Newton's Method, Constrained Optimization: Lagrange Multipliers, KKT Conditions, Implementation of linear and non-linear optimization algorithms using Python (SciPy, CVXPY).

Unit 3: Discrete Optimization and Heuristics

Integer Programming: Branch and Bound, Cutting Planes, Combinatorial Optimization: Traveling Salesman Problem, Knapsack Problem, Heuristic Methods: Genetic Algorithms, Simulated Annealing, Tabu Search, Solving discrete optimization problems using Python (PuLP, DEAP).

Unit 4: Advanced Optimization Techniques and Applications

Multi-Objective Optimization: Pareto Optimality, Weighted Sum Method, Metaheuristics: Particle Swarm Optimization, Ant Colony Optimization, Real-World Applications: Scheduling, Network Optimization, Machine Learning, Developing optimization solutions for real-world problems using Python and specialized libraries (e.g., PyGMO).

Practical Component

1. Hands-on practice with optimization algorithms and tools.
2. A small project to apply optimization techniques to various problems.
3. A comprehensive project to formulate and solve a complex optimization problem.
4. Reviewing and analyzing optimization problems in real-world scenarios.

Suggested Readings

1. "Introduction to Operations Research" by Frederick S. Hillier and Gerald J. Lieberman
2. "Linear and Nonlinear Programming" by David G. Luenberger and Yinyu Ye
3. "Optimization in Operations Research" by Ronald L. Rardin
4. "Practical Optimization: Algorithms and Engineering Applications" by Andreas Antoniou and Wu-Sheng Lu

List of Experiments

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

COMPILER DESIGN (DSE-4)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Compiler Design	4	3L	0T	1P	-	-

Course Outcomes

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

1. Understand the phases and components of a compiler, including lexical analysis, syntax analysis, and code generation.
2. Design and implement lexical analyzers using finite state machines and regular expressions.
3. Apply different parsing techniques such as top-down parsing, bottom-up parsing, and LR parsing to construct efficient parsers.
4. Develop and implement syntax-directed translation schemes to generate intermediate code from high-level languages.
5. Use appropriate data structures, such as symbol tables, for managing information during the compilation process.
6. Implement error detection and recovery techniques to handle common syntax and semantic errors during compilation.
7. Optimize code through various techniques, such as loop optimization and DAG representation.

Course Objectives

1. Provide students with a comprehensive understanding of compiler components and their functionalities.
2. Introduce students to the design and implementation of lexical analyzers using finite state machines and regular expressions.
3. Teach various parsing techniques and help students develop efficient parsers for different programming languages.
4. Introduce syntax-directed translation schemes and help students generate intermediate code for high-level programming languages.
5. Familiarize students with the implementation of symbol tables and memory management during the compilation process.
6. Equip students with the skills to detect and recover from errors during the compilation process.
7. Teach optimization techniques to improve the efficiency of the generated code.

Unit 1: Introduction to Compilers

Definition and Role of a Compiler; Phases of a Compiler: Lexical Analysis, Syntax Analysis, Semantic Analysis, Intermediate Code Generation, Optimization, and Code Generation; Compiler Passes: Single-pass and Multi-pass Compilers; Overview of Finite State Machines (FSM) and Regular Expressions in Lexical Analysis; Implementation of Lexical Analyzers and Lexical Analyzer Generators; Introduction to Lex – A Lexical Analyzer Generator.

Unit 2: Syntax Analysis

Role of Syntax Analysis in Compilers; Context-Free Grammars (CFG) and Backus-Naur Form (BNF); Parse Trees and Abstract Syntax Trees; Parsing Techniques: Top-Down Parsing (Recursive Descent, Predictive Parsing) and

Bottom-Up Parsing (Shift-Reduce Parsing, Operator Precedence Parsing); LR Parsers: SLR, Canonical LR, and LALR Parsing Techniques; Constructing Parsing Tables; Introduction to YACC – A Parser Generator.

Unit 3: Syntax-Directed Translation and Intermediate Code Generation

Syntax-Directed Definitions and Translation Schemes; Implementation of Syntax-Directed Translators; Intermediate Representations: Postfix Notation, Syntax Trees, Three-Address Code, Quadruples, and Triples; Translation of Programming Constructs: Assignment Statements, Boolean Expressions, Control Flow, Array References, Procedure Calls, and Case Statements; Runtime Environments and Storage Allocation Strategies.

Unit 4: Code Optimization and Code Generation

Need for Optimization; Basic Blocks and Flow Graphs; Loop Optimization Techniques; Data Flow Analysis for Optimization; DAG Representation of Basic Blocks; Peephole Optimization; Code Generation Techniques: Instruction Selection, Register Allocation, and Instruction Scheduling; Challenges in Code Generation: Target Machine Architecture and Runtime Considerations.

Practical Component

1. Implementing a lexical analyzer using regular expressions and finite state machines (FSM).
2. Developing a simple parser using recursive descent parsing for basic arithmetic expressions.
3. Constructing and testing an LR parser using tools like YACC or Bison.
4. Implementing syntax-directed translation schemes to convert expressions into intermediate code.
5. Designing a symbol table management system for tracking variables, functions, and scopes.
6. Developing a code generator that produces three-address code from a simple high-level programming language.
7. Implementing an optimization pass to perform dead code elimination or constant folding on intermediate code.

Suggested Readings

1. "Compiler Design: Principles, Techniques and Tools", 2nd Ed., Prentice-Hall, 2006 by AV Aho, MS Lam, R Sethi, JD Ullman.
2. "Modern Compiler Implementation in Java", Cambridge University Press, 2002 by AW Appel, J Palsberg.
3. "Modern Compiler Implementation in C", Cambridge University Press, 2004 by AW Appel, M Ginsburg.
4. "Engineering a Compiler", 2nd Ed., Morgan Kaufmann, 2011 by K Cooper, L Torczon.
5. "Compiler Construction: Principles and Practice", Cengage Learning, 1997 by KC Louden.
6. "Modern Compiler Design", Wiley, 2000 by D Grune, H Bal, C Jacobs, K. Langendoen.
7. "Programming Language Pragmatics", 3rd Ed., Morgan Kaufmann, 2009 by Michael L Scott.
8. "Advanced Compiler Design and Implementation", Morgan Kaufmann/Elsevier(India), 2003 by S Muchnick.

List of Experiments

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

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 in Third Year

1. **Minor in CSE (Offered to ECE and EE)**
    a. DSE-3 / GE-5: Foundations of Computer Networks
    b. DSE-4 / GE-6: Fundamentals of Software Engineering

2. **Minor/Specialization in Artificial Intelligence and Machine Learning (Offered to CSE, ECE and EE)**
    a. DSE-3 / GE-5: Data Mining & Warehousing
    b. DSE-3 / GE-5: Neural Networks
    c. DSE-4 / GE-6: AI for Image Processing
    d. DSE-4 / GE-6: NLP: Techniques and Applications

3. **Minor/Specialization in Data Science (Offered to CSE, ECE and EE)**
    a. DSE-3 / GE-5: Data Mining & Warehousing
    b. DSE-3 / GE-5: Advanced Data Analytics
    c. DSE-4 / GE-6: Health Data Analytics
    d. DSE-4 / GE-6: Fundamentals of Time Series Analysis

4. **Minor/Specialization in Software Engineering (Offered to CSE, ECE and EE)**
    a. DSE-3 / GE-5: Predictive Modeling
    b. DSE-3 / GE-5: Software Testing
    c. DSE-4 / GE-6: Agile Software Development
    d. DSE-4 / GE-6: Software Reliability

5. **Minor/Specialization in Blockchain and Cybersecurity (Offered to CSE, ECE and EE)**
    a. DSE-3 / GE-5: Blockchain Essentials
    b. DSE-3 / GE-5: Ethical Hacking and Advanced Cybersecurity
    c. DSE-4 / GE-6: Blockchain Applications
    d. DSE-4 / GE-6: Cybersecurity with Blockchain

6. **Minor/Specialization in Augmented Reality / Virtual Reality (Offered to CSE, ECE and EE)**
    a. DSE-3 / GE-5: Human Computer Interaction
    b. DSE-3 / GE-5: Game Development
    c. DSE-4 / GE-6: Advanced Game Development
    d. DSE-4 / GE-6: Augmented and Virtual Reality Systems Design

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

Detailed Syllabus of Discipline Specific Elective (DSE) / Generic Elective (GE) courses offered for Minors / Specializations by the department in Semester V

Foundations of Computer Networks (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Foundations of Computer Networks	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand and explain the various layers of computer networks and the protocols used in each layer.
2. Analyze network performance using key metrics and troubleshoot networking issues.
3. Demonstrate the ability to design and implement network solutions.
4. Apply networking concepts through practical experiments in simulated environments.

Course Objectives

1. To introduce the fundamental concepts of computer networks and communication systems.
2. To provide an understanding of the network architecture, protocols, and layers.
3. To examine various types of networks and their components, including physical media, protocols, and network devices.

UNIT I

Networking Fundamentals: Introduction to the concepts of computer networking, Types of networks: LAN, WAN, MAN, PAN, Networking topologies, protocols, and models, Introduction to the OSI and TCP/IP models, Basics of IP addressing and subnetting

UNIT II

Data Link Layer and Network Layer: Functions of the data link layer: framing, error control, flow control, MAC protocols: CSMA/CD, CSMA/CA, Network layer functionalities: routing, addressing, packet forwarding, Introduction to IPv4 and IPv6, Basic concepts of routing protocols (RIP, OSPF, BGP)

UNIT III

Transport Layer and Application Layer: Principles of transport layer protocols: TCP and UDP, Mechanisms of congestion control and error handling, Application layer protocols and services: HTTP, DNS, SMTP, FTP, Principles of network security: encryption, firewalls, VPNs

UNIT IV

Network Management and Emerging Technologies: Basics of network management: SNMP, network configuration, and troubleshooting, Wireless networking technologies: WiFi, Bluetooth, LTEIntroduction to network virtualization and SDN (Software Defined Networking), Future trends in networking: IoT, 5G, cloud networking

Practical Component

1. Configuring a small network with routers and switches.
2. Performing network traffic analysis using packet sniffing tools.
3. Setting up and configuring a VPN.
4. Implementing basic secure network communication protocol configuration on a firewall.

Suggested Readings

1. "Networking All-in-One For Dummies" by Doug Lowe
2. "CCNA Routing and Switching 200-125 Official Cert Guide Library" by Wendell Odom
3. "Data Communications and Networking" by Behrouz A. Forouzan
4. "Computer Networks" by Andrew S. Tanenbaum and David J. Wetherall (5th Edition, Pearson)
5. "Computer Networking: A Top-Down Approach" by James Kurose and Keith Ross (6th Edition, Pearson)
6. "Data and Computer Communications" by William Stallings (10th Edition, Pearson)
7. "Network Security Essentials: Applications and Standards" by William Stallings

List of Experiments

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

Data Mining & Warehousing (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Data Mining & Warehousing	4	3L	1T	0P	-	Database Management System

Course Hours: L: 03 T: 01 P: 00

Course Outcomes

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

1. Understand the core concepts, architectures, and data modeling techniques of data warehousing, and relate them to OLAP operations and data cube technology.
2. Apply data preprocessing methods, association rule mining, classification, and clustering algorithms to discover actionable insights from large, complex datasets.
3. Evaluate various clustering techniques, including partitioning, density-based, and hierarchical methods, and assess their effectiveness in segmenting data.

Course Objectives

1. Impart a thorough understanding of data warehousing concepts, architectures, and data modeling methods, including star and snowflake schemas, as well as the ETL processes and OLAP operations.
2. Familiarize students with fundamental data mining concepts and techniques, enabling them to handle and analyze large datasets effectively.
3. Highlight the importance of mining complex data types (text, web, time series) and familiarize students with modern tools and techniques required to extract meaningful patterns and knowledge.
4. Expose students to real-world business applications, case studies, and current trends in data warehousing and data mining.

UNIT I

Introduction to Data Warehousing: Concepts and architecture of a data warehouse, Data modeling for data warehouses: star schema, snowflake schema, ETL processes: extraction, transformation, loading, OLAP in data warehouse, Data Cube Technology, From Data Warehousing to Data Mining

UNIT II

Data Mining Concepts: Overview of data mining and knowledge discovery in databases, Data preprocessing: cleaning, normalization, transformation, Association rule mining: Apriori, FP-growth

Data Mining Techniques:Classification techniques: decision trees, Naive Bayes, KNN, SVM, random forests, Prediction, Classifier Performance measures, Clustering methods:

UNIT III

Cluster Analysis in Data Mining: Types of Data in Cluster Analysis. A Categorization of Major Clustering Methods, Partitioning Methods, Density Based Methods, Grid Based Methods; Model Based Clustering Methods, k-means clustering, hierarchical clustering, DBSCAN, Outlier Analysis

Mining Complex Data: Text mining and natural language processing, Mining Time Series and Sequence Data, Web mining: content mining, structure mining, usage mining, Ethical issues and privacy concerns in data mining

UNIT IV

Real-world Applications of Data Mining and Warehousing: Business intelligence and decision support systems, Data mining in retail, finance, healthcare, and social media, Recommender systems and market basket analysis, Case studies on successful data mining projects, Future trends in data mining and data warehousing

List of tutorial tasks

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

Suggested Readings

1. "Data Mining: Concepts and Techniques" by Jiawei Han, Micheline Kamber, and Jian Pei
2. "The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling" by Ralph Kimball and Margy Ross
3. "Data Mining Introductory & Advanced Topics" by M.H. Dunham, Pearson Education
4. "Data Warehousing Fundamentals" by P. Ponnian, John Wiley.
5. "Master in Data Mining" by M. Berry, G. Linoff, John Wiley
6. "Big Data: Principles and Best Practices of Scalable Realtime Data Systems" by Nathan Marz and James Warren

Neural Networks (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Neural Networks	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the architecture and functioning of neural networks, including perceptrons and multi-layer perceptrons.
2. Implement neural network models using backpropagation and gradient descent optimization techniques.
3. Apply Convolutional Neural Networks (CNNs) to solve problems related to image classification and object detection.
4. Design and implement Recurrent Neural Networks (RNNs) for sequence-based tasks like time-series forecasting and language modeling.
5. Develop deep learning models using frameworks like TensorFlow or PyTorch.
6. Evaluate the performance of neural network models through various metrics and techniques, such as cross-validation.
7. Understand the challenges associated with training deep learning models and techniques for handling them, including regularization and dropout.

Course Objectives

1. Introduce students to the fundamental concepts of neural networks and deep learning.
2. Teach students to design, implement, and train simple neural network architectures.
3. Equip students with practical experience in building CNNs and applying them to image processing tasks.
4. Introduce RNNs and demonstrate their use in sequence prediction tasks.
5. Provide hands-on experience in using deep learning frameworks and libraries for model implementation.
6. Teach students how to evaluate and improve the performance of neural network models.
7. Address the common challenges faced in training deep learning models, including overfitting and optimization.

Unit 1: Introduction to Neural Networks

Overview of neural networks and their role in artificial intelligence. Biological inspiration and comparison to artificial neurons. Basic structure of neural networks: perceptrons, activation functions (sigmoid, ReLU, tanh), and feedforward architectures. Fundamentals of supervised and unsupervised learning. Applications of neural networks in classification, regression, and clustering tasks.

Unit 2: Training Neural Networks

Backpropagation and gradient descent for weight optimization. Loss functions (mean squared error, cross-entropy) and performance metrics. Hyperparameter tuning, including learning rate, batch size, and epochs. Regularization techniques to prevent overfitting: L1/L2 regularization, dropout, and early stopping. Introduction to optimizers like SGD, Adam, and RMSprop. Challenges in training: vanishing and exploding gradients.

Unit 3: Advanced Architectures and Concepts

Deep neural networks (DNNs) and their layers: convolutional layers, pooling layers, and fully connected layers. Introduction to convolutional neural networks (CNNs) for image processing and recurrent neural networks (RNNs) for sequence prediction. Understanding long short-term memory (LSTM) and gated recurrent unit (GRU) networks. Transfer learning and pre-trained models. Generative adversarial networks (GANs) for data synthesis.

Unit 4: Applications and Future Directions

Practical applications of neural networks: computer vision, natural language processing, robotics, and healthcare. Neural networks in recommendation systems, autonomous vehicles, and financial forecasting. Introduction to reinforcement learning and neural networks' role in AI decision-making. Ethical considerations and challenges in deploying neural network models. Future trends: spiking neural networks and neuromorphic computing.

Practical Component

1. Implementing a simple neural network from scratch to solve a classification problem (e.g., XOR problem).
2. Developing and training a Convolutional Neural Network (CNN) for image classification using TensorFlow/Keras.
3. Implementing a Recurrent Neural Network (RNN) for sequence prediction tasks like time-series forecasting.
4. Fine-tuning a pre-trained neural network model for transfer learning tasks.
5. Developing an image generation model using Generative Adversarial Networks (GANs).
6. Evaluating and improving neural network performance using techniques such as dropout and data augmentation.
7. Experimenting with hyperparameter optimization and model evaluation to enhance the performance of deep learning models.

Suggested Readings

1. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
2. "Transformers for Natural Language Processing" by Denis Rothman
3. "Attention Is All You Need" by Vaswani et al., seminal paper introducing transformers
4. "GPT-3: Language Models are Few-Shot Learners" by Brown et al., detailing the architecture and capabilities of GPT-3

List of Experiments

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

Advanced Data Analytics (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Advanced Data Analytics	4	3L	0T	1P	-	Fundamentals of Data Analytics

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

Course Outcomes

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

1. Understand the fundamental concepts and techniques of data analytics, including data preprocessing, statistical modeling, and exploratory data analysis.
2. Develop predictive analytics models using various machine learning algorithms and evaluate their performance for informed decision-making.
3. Apply advanced analytics techniques such as clustering, dimensionality reduction, time series forecasting, and text mining to extract meaningful insights from complex datasets.
4. Understand and implement big data analytics principles and tools to handle large-scale and unstructured data efficiently.

Course Objectives

1. Equip students with a strong foundation in data preprocessing, statistical analysis, and exploratory data analytics.
2. Enable the application of advanced modeling, machine learning, and big data techniques for deriving actionable insights from diverse data sources.
3. Instill practical understanding, and future-oriented perspectives by examining real-world applications and emerging trends.

Unit 1: Data Analytics Fundamentals

Introduction to Data Analytics: Role, importance, and applications in various sectors (finance, healthcare, marketing, etc.). Data Types and Quality: Structured, semi-structured, and unstructured data; issues of missing values, noise, outliers. Data Preprocessing: Cleaning, normalization, dimensionality reduction (overview), and feature engineering. Statistical Foundations: Probability distributions, hypothesis testing, confidence intervals, correlation, and ANOVA. Exploratory Data Analysis (EDA) and Visualization: Summary statistics, histograms, box plots, scatter plots, heat maps, and dashboards using tools like Matplotlib, Seaborn, Plotly, Tableau.

Unit 2: Predictive Analytics and Modeling

Regression Techniques: Linear regression (simple, multiple), regularization methods (Ridge, Lasso), logistic regression for classification. Tree-based Models: Decision trees, ensemble methods (random forests, gradient boosting). Support Vector Machines (SVM) and Neural Networks: Introduction to kernel methods, perceptrons, feedforward networks. Model Evaluation: Train-test splits, cross-validation, confusion matrix, ROC curves, AUC, precision, recall, F1-score. Model Selection and Tuning: Hyperparameter optimization (grid search, random search, Bayesian optimization). Formulation of predictive tasks from business problems, Translation of business problems into analytical models.

Unit 3: Advanced Analytics Techniques

Unsupervised Learning: Clustering methods (k-means, hierarchical clustering, DBSCAN), principal component analysis (PCA) for dimensionality reduction. Time Series Analysis and Forecasting: ARIMA, seasonal decomposition, exponential smoothing, and introduction to LSTM-based forecasting. Text Mining and Sentiment Analysis: Text preprocessing (tokenization, stemming, lemmatization), topic modeling, sentiment classification. Hadoop ecosystem, Spark for large-scale analytics, NoSQL databases.

Unit 4: Data Analytics in Practice

Ethical Considerations: Privacy, bias, fairness, transparency, and interpretability in analytical models. Industry-Specific Applications, financial analytics, healthcare analytics. Case Studies: Examination of real-world projects showcasing the end-to-end data analytics lifecycle.

Practical Component

1. Conducting exploratory data analysis using a popular data analysis tool or programming language.
2. Building and evaluating predictive models for a given dataset.
3. Implementing clustering techniques on a dataset to uncover patterns.
4. Implementing time series forecasting models on a time-series dataset.
5. Extract insights from textual data by performing sentiment analysis and topic modeling.
6. Work with a domain-specific dataset to build an end-to-end analytics solution - data ingestion, preprocessing, modeling, and insights presentation.

Suggested Readings

1. "Data Mining: Concepts and Techniques" by Jiawei Han, Micheline Kamber, Jian Pei
2. "Applied Predictive Modeling" by Max Kuhn, Kjell Johnson
3. "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking" by Foster Provost and Tom Fawcett
4. "Practical Statistics for Data Scientists: 50 Essential Concepts" by Peter Bruce and Andrew Bruce
5. "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython" by Wes McKinney

List of Experiments

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

Predictive Modeling (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Predictive Modeling	4	3L	1T	0P	-	Software Engineering

Course Hours: L: 03 T: 01 P: 00

Course Outcomes

By the end of the course, students will be able to:

1. Understand the principles and significance of predictive modeling in software engineering.
2. Design and implement predictive models for software engineering problems using empirical methods.
3. Evaluate predictive models using statistical and performance metrics.
4. Apply machine learning and data-driven approaches to improve software quality and project outcomes.

Course Objectives

1. To understand the fundamentals of predictive modeling in the context of software engineering.
2. To explore the application of machine learning techniques to predict software quality attributes.
3. To evaluate the performance of predictive models using appropriate metrics.
4. To introduce recent trends and advancements in predictive modeling for software development and maintenance.

UNIT I

Fundamentals of Predictive Modelling in Software Engineering: Introduction to predictive modeling in software engineering, Overview of empirical methods in software engineering research, Data collection and preprocessing for predictive modeling, Software engineering metrics: product, process, and resource metrics, Applications of predictive modeling in defect prediction, effort estimation, and maintenance

UNIT II

Machine Learning Techniques for Predictive Modelling: Supervised learning techniques: Linear regression, logistic regression, and decision trees, Support Vector Machines (SVMs) and ensemble methods (e.g., Random Forests, Gradient Boosting), Unsupervised learning techniques: Clustering and dimensionality reduction, Neural networks and deep learning for software engineering data, Feature selection and engineering for improving model performance, Case studies: Defect prediction using classification models

UNIT III

Model Evaluation and Validation: Metrics for predictive model evaluation: Accuracy, precision, recall, F1-score, Mean Absolute Error (MAE), Mean Squared Error (MSE), and RMSE, Cross-validation techniques (e.g., k-fold, leave-one-out), Handling imbalanced datasets in software engineering (e.g., SMOTE), Overfitting and underfitting in predictive models, Tools and frameworks for predictive modeling in Python (e.g., scikit-learn, Weka)

UNIT IV

Recent Trends and Advanced Topics: Explainable AI (XAI) in software predictive modeling, Predictive modeling for agile and DevOps practices, Effort estimation using advanced ML models (e.g., Bayesian models, transformer

architectures), Applications in software evolution and maintenance (e.g., predicting change impact), Emerging trends: Transfer learning and meta-learning for software engineering data

Tutorial Component

1. Hands-on experiments with defect prediction datasets (e.g., PROMISE repository).
2. Feature selection and extraction from software metrics data.
3. Implementation of predictive models using Python or Weka.
4. Evaluation of model performance using cross-validation and statistical metrics.
5. Group activity: Developing a predictive model for a software engineering problem (e.g., effort estimation).

List of tutorial tasks

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

Suggested Readings

1. "Empirical Software Engineering" by Ruchika Malhotra
2. "Data Mining and Predictive Analytics" by Daniel T. Larose and Chantal D. Larose
3. Research papers and case studies on predictive modeling in software engineering

Software Testing (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Software Testing	4	3L	0T	1P	-	Software Engineering

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

Course Outcomes

By the end of the course, students will be able to:

1. Explain key software testing concepts, principles, and processes.
2. Design test plans, test cases, and scenarios for different types of software applications.
3. Use automated testing tools to improve efficiency and accuracy.
4. Analyze software quality using appropriate metrics and standards.
5. Collaborate in teams to conduct testing and quality assurance activities effectively.
6. Understand where key testing concepts apply within the context of unified processes.

Course Objectives

1. To understand fundamental concepts in software testing, including software testing objectives, process, criteria, strategies, and methods.
2. To Identify a number of test styles and techniques and assess their usefulness in a particular context.
3. To discuss various software testing issues and solutions in software.
4. To learn how to plan a test project, design test cases and data, conduct testing operations, manage software problems and defects.

UNIT I

Fundamentals of Software Testing: Introduction to software testing and its importance, Testing Process, Limitations of Testing, Testing activities, Testing levels: unit testing, integration testing, system testing, acceptance testing, debugging, Performance testing, load testing, stress testing, regression testing

UNIT II

Verification Testing: Verification Methods, SRS Verification, Software Design Document Verification, Code Reviews, User Documentation Verification, Software Project Audits**.** White-box testing, black-box testing, and gray-box testing

Functional Testing: Boundary Value Analysis, Equivalence Class Testing, Decision Table Based Testing, Cause Effect Graphing Technique**.**

UNIT III

Structural Testing: Path testing, DD-Paths, Cyclomatic Complexity, Graph Metrics, Data Flow Testing, Mutation testing. Object Oriented Testing: Class Testing, GUI Testing.

Software Testing Tools: Defect life cycle and defect tracking tools (e.g., JIRA, Bugzilla), Fundamentals of automation testing, Benefits and challenges of automation testing, Methodology to evaluate automated testing, Testing Tools, Java Testing Tools, JMetra, JUNIT Cactus and ot

UNIT IV

Advanced Topics in Software Testing: Security testing, Usability testing, Testing in Agile and DevOps environments, Emerging trends: AI/ML in testing, Mobile app testing, Cloud-based testing

Practical Component

1. Developing and executing test cases for a software application.
2. Executing black-box and white-box testing techniques
3. Implementing automated tests using a testing framework.
4. Conducting performance testing on a web application.
5. Participating in a team project to manage and document the testing process for a software release.

Suggested Readings

1. Yogesh Singh, “Software Testing”, 1st Ed., Cambridge University Press.
2. Paul C. Jorgenson, Software Testing A Craftsman’s approach, CRC Press.
3. "Software Testing: Principles and Practices" by Srinivasan Desikan and Gopalaswamy Ramesh
4. Louise Tamres, “Software Testing”, Pearson Education Asia.
5. "The Art of Software Testing" by Glenford J. Myers, Corey Sandler, and Tom Badgett
6. "Agile Testing: A Practical Guide for Testers and Agile Teams" by Lisa Crispin and Janet Gregory

List of Experiments

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

Blockchain Essentials (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Blockchain Essentials	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the fundamental principles and cryptographic underpinnings of blockchain technology, and differentiate between various types of blockchain architectures.
2. Evaluate and select appropriate blockchain platforms and frameworks, and develop, deploy, and test smart contracts and decentralized applications (DApps).
3. Analyze real-world use cases of blockchain across multiple industries, identify regulatory and ethical challenges, and propose secure and compliant blockchain solutions.

Course Objectives

1. Introduce the foundational concepts, architecture, and cryptographic components of blockchain technology.
2. Provide hands-on exposure to blockchain platforms, enabling students to design, code, and deploy smart contracts and decentralized applications.
3. Examine practical industry applications of blockchain in finance, supply chain, healthcare, and government, and understand associated regulatory and ethical considerations.

Unit 1: Blockchain Fundamentals

Introduction to blockchain technology and its significance, Blockchain architecture: blocks, chains, consensus mechanisms, Types of blockchains: public, private, consortium, Cryptography in blockchain: hash functions, public key cryptography

Unit 2: Blockchain Platforms and Development

Overview of popular blockchain platforms: Ethereum, Hyperledger, Bitcoin, Smart contracts: definition, development, and deployment, Development tools and environments for blockchain, Building decentralized applications (DApps)

Unit 3: Blockchain in Industry

Blockchain applications in finance: cryptocurrencies, ICOs, DeFi, Blockchain for supply chain management, healthcare, and government, Regulatory and ethical considerations of blockchain technology, Security challenges and solutions in blockchain systems

Unit 4: Emerging Trends in Blockchain

Advances in consensus mechanisms: Proof of Stake, Delegated Proof of Stake, Proof of Authority, Scalability solutions: sidechains, sharding, layer 2 protocols, Interoperability among blockchain systems, Future directions and potential impacts of blockchain technology

Practical Component

1. Creating a simple smart contract and deploying it on a blockchain platform.
2. Developing basic DApp(s) using Ethereum and Solidity.
3. Analyzing blockchain use cases and proposing a solution for a real-world problem.

Suggested Readings

1. "Mastering Blockchain: Unlocking the Power of Cryptocurrencies, Smart Contracts, and Decentralized Applications" by Lorne Lantz and Daniel Cawrey
2. "Blockchain Basics: A Non-Technical Introduction in 25 Steps" by Daniel Drescher
3. "Blockchain Applications: A Hands-On Approach" by Arshdeep Bahga and Vijay Madisetti

List of Experiments

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

Ethical Hacking and Advanced Cybersecurity (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Ethical Hacking and Advanced Cybersecurity	4	3L	0T	1P	-	Fundamentals of Cybersecurity

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

Course Outcomes

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

1. Understand and apply the principles of ethical hacking and penetration testing.
2. Detect and exploit vulnerabilities in a controlled and responsible manner.
3. Utilize advanced tools and frameworks for ethical hacking and network defense.
4. Conduct forensic investigations and malware analysis to identify the root causes of breaches.
5. Formulate and implement advanced cybersecurity strategies to protect critical infrastructure.

Course Objectives

1. Introduce ethical hacking techniques and practices to identify vulnerabilities in systems and networks.
2. Provide insights into advanced cybersecurity measures, including malware analysis and threat hunting.
3. Equip students with tools for ethical hacking, such as Kali Linux, Metasploit, and Burp Suite.
4. Develop ethical responsibility and compliance with legal frameworks in cybersecurity.
5. Enable hands-on experience in simulating and defending against sophisticated cyber-attacks.

Unit 1: Fundamentals of Ethical Hacking

Definition, Scope, and Importance of Ethical Hacking; Understanding Cyber Laws and Ethical Hacking Ethics; Types of Hackers (White Hat, Black Hat, Grey Hat); Phases of Ethical Hacking (Reconnaissance, Scanning, Gaining Access, Maintaining Access, Covering Tracks); Introduction to Ethical Hacking Tools such as Kali Linux, Metasploit, and Wireshark.

Unit 2: Network and System Security

Overview of Network Security: Firewalls, IDS/IPS, VPNs, Proxy Servers; Vulnerability Assessment and Penetration Testing (VAPT) Techniques and Tools; Network Hacking: Packet Sniffing, Spoofing, and Man-in-the-Middle (MITM) Attacks; System Security Concepts: Privilege Escalation, Backdoors, and Trojans; Introduction to Wireless Network Security and WPA/WPA2 Cracking.

Unit 3: Application and Web Security

Understanding Web Application Attacks: SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF); Securing Web Applications: OWASP Top 10 Framework; Exploring Secure Coding Practices; Hacking APIs and Web Services; Introduction to Mobile Application Security.

Unit 4: Advanced Cybersecurity Concepts

Incident Detection and Response: Monitoring and Analyzing Cyber Attacks; Cryptographic Techniques: Encryption Standards, Digital Signatures, and Certificates; Advanced Persistent Threats (APTs): Detection and Mitigation

Strategies; Exploring Blockchain Security and Zero-Trust Architecture; Overview of Emerging Threats in AI and IoT.

Practical Component

1. Conducting reconnaissance using tools like Maltego to gather intelligence on a target system.
2. Exploiting vulnerabilities in a controlled lab environment to understand the ethical hacking lifecycle.
3. Developing exploits using custom scripts and testing them in a sandboxed environment.
4. Performing secure code analysis to identify common vulnerabilities like SQL injection or XSS.
5. Building and deploying honeypots to monitor and log potential attack attempts.
6. Simulating ransomware attacks and crafting effective incident response strategies.
7. Conducting security audits of cloud platforms to ensure compliance with security standards.

Suggested Readings

1. "The Hacker Playbook 3: Practical Guide To Penetration Testing" by Peter Kim
2. "Hacking: The Art of Exploitation" by Jon Erickson
3. "Penetration Testing: A Hands-On Introduction to Hacking" by Georgia Weidman
4. Hacking: The Art of Exploitation by Jon Erickson (No Starch Press)
5. Penetration Testing: A Hands-On Introduction to Hacking by Georgia Weidman (No Starch Press)
6. The Web Application Hacker's Handbook by Dafydd Stuttard and Marcus Pinto (Wiley)
7. Metasploit: The Penetration Tester’s Guide by David Kennedy, Jim O’Gorman, Devon Kearns, and Mati Aharoni (No Starch Press)

List of Experiments

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

Human Computer Interaction (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Human Computer Interaction	4	3L	1T	0P	-	-

Course Hours: L: 03 T: 01 P: 00

Course Outcomes

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

1. Understand the fundamental theories, principles, and concepts of HCI, and apply user-centered design methodologies to create intuitive interfaces.
2. Employ interaction design principles, prototyping techniques, and accessibility guidelines to design effective interfaces for various platforms and devices.
3. Plan, conduct, and analyze usability evaluations to assess user experience, iteratively refining designs based on qualitative and quantitative feedback.

Course Objectives

1. Introduce students to the core principles, history, and cognitive frameworks underlying human-computer interaction.
2. Provide hands-on experience in interaction design, prototyping, and the implementation of user-centered and accessible interfaces.
3. Train students in usability evaluation methods, enabling them to measure and improve the quality of user experiences through iterative design.

Unit 1: Introduction to HCI

Fundamentals of HCI: Human capabilities, interaction paradigms, and interface metaphors. History and Evolution: From command-line interfaces to GUIs, multi-touch, and beyond. User-Centered Design (UCD): Key concepts, user involvement, requirement gathering, and iterative development. Cognitive Models and UI Design: Mental models, GOMS, cognitive load, and perception principles. Evaluating User Experience (UX): Metrics, qualitative and quantitative measures, satisfaction, efficiency, and effectiveness.

Unit 2: Designing Interactive Systems

Interaction Design Basics: Affordances, signifiers, feedback loops, constraints, and mapping. Design Thinking and Prototyping: Design research, ideation techniques, low-fi and hi-fi prototyping, wireframing, and storyboarding. User Interface Design Principles: Consistency, visibility, learnability, error prevention, standards. Designing for web, mobile, and desktop. Accessibility and Inclusive Design: WCAG guidelines, designing for diverse users, assistive technologies, and inclusive user scenarios. Tools and Technologies: UI/UX design software (Sketch, Figma, Adobe XD), prototyping tools, style guides, and design systems.

Unit 3: Usability Evaluation and Testing

Usability concepts and heuristics, Planning and conducting usability tests, Quantitative and qualitative methods for evaluating HCI, Analyzing and interpreting test results, Iterative design and usability engineering

Unit 4: Advanced Topics in HCI

Emotional design and user engagement, Voice user interfaces and conversational design, Augmented reality (AR) and virtual reality (VR) in HCI, Future trends in HCI: AI, IoT, and wearable technologies, Ethical considerations in HCI design and research

Tutorial Component

1. Conduct interviews or surveys to identify user needs, create user personas, and draft preliminary requirements.
2. Develop low-fidelity wireframes and interactive prototypes for a given application scenario using tools like Figma or Sketch.
3. Apply Nielsen's heuristics to evaluate a prototype or an existing interface, identify usability issues, and propose improvements.
4. Conduct a small-scale usability test with participants on a prototype, use think-aloud protocols, record observations, and summarize findings.
5. Experiment with a simple voice interface (using existing frameworks) or explore a basic AR scenario, observing user interaction.
6. Developing an accessible web or mobile application.

Suggested Readings

1. "Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability" by Steve Krug
2. "The Design of Everyday Things" by Don Norman
3. "About Face: The Essentials of Interaction Design" by Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel
4. "Interaction Design: Beyond Human-Computer Interaction" by Jenny Preece, Helen Sharp, Yvonne Rogers

Game Development (DSE-3/GE-5)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Game Development	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the fundamental concepts, history, and processes involved in developing video games, from initial design to final deployment.
2. Apply game design principles to create engaging game worlds, narratives, levels, and characters suitable for various platforms and genres.
3. Develop and implement core gameplay functionalities using modern game engines, programming languages, graphics, physics, and AI techniques.

Course Objectives

1. Introduce the foundational principles of game development, including game genres, platforms, and industry-standard tools.
2. Cultivate skills in game design, storytelling, character creation, and level design, emphasizing user experience and engagement.
3. Provide technical proficiency in programming, graphics rendering, physics simulation, and AI, enabling students to implement game features effectively.

Unit 1: Introduction to Game Development

Overview of Game Development Process: Conceptualization, prototyping, production, polishing, and publishing. History and Evolution of Video Games: Arcade classics, console generations, PC gaming evolution, mobile and VR innovations. Game Genres and Platforms: Action, adventure, RPG, simulation, strategy, mobile vs. console vs. PC, VR/AR platforms. Introduction to Game Design Principles: Player-centered design, core gameplay loops. Game Engines and Tools: Introduction to Unity, Unreal Engine; asset creation tools (Blender, Maya); version control systems (Git).

Unit 2: Game Design and Storytelling

Elements of game design: mechanics, dynamics, aesthetics, Narrative and storytelling in games, Character development and world-building, Level design principles, Designing for different platforms: PC, consoles, mobile, VR/AR

Unit 3: Game Programming

Introduction to programming languages used in game development (C#, C++ etc), Game engine architecture and components, Graphics programming: rendering, shaders, particle systems, Physics and collision detection, AI in games: pathfinding, decision-making, NPC behaviors

Unit 4: Game Production and Post-Production

Project management in game development: Agile and Scrum methodologies, Quality assurance, testing strategies, and debugging, Marketing and monetization strategies for games, Post-launch support and community management, Emerging trends and future directions in game development

Practical Component

1. Designing a game concept and creating a design document.
2. Developing a simple game using a chosen game engine.
3. Implementing key game mechanics and programming AI behaviors.
4. Testing and debugging the game, followed by a presentation of the project.

Suggested Readings

1. "The Art of Game Design: A Book of Lenses" by Jesse Schell
2. "Rules of Play: Game Design Fundamentals" by Katie Salen and Eric Zimmerman
3. "Game Programming Patterns" by Robert Nystrom
4. "Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
5. "Learning C++ by Creating Games with Unreal Engine 4" by Sharan Volin

List of Experiments

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

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

Detailed Syllabus of Discipline Specific Elective (DSE) / Generic Elective (GE) courses offered for Minors / Specializations by the department in Semester VI

Fundamentals of Software Engineering (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Fundamentals of Software Engineering	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the fundamental concepts, principles, and models of software engineering and apply them to plan, develop, and manage software projects.
2. Analyze and specify software requirements using structured techniques and UML, ensuring that both functional and non-functional requirements are met.
3. Employ software design principles, coding standards, and quality measures to produce maintainable and efficient software.
4. Estimate project costs, identify and manage risks, and apply appropriate testing methodologies to ensure the reliability and quality of software systems.
5. Comprehend the processes of software maintenance, configuration management, and stay informed about emerging trends to adapt to evolving industry practices.

Course Objectives

1. Introduce students to the fundamental concepts of software engineering, including SDLC models, software metrics, and project management fundamentals.
2. Enable students to elicit, analyze, and document software requirements, and to understand various software design paradigms and coding best practices.
3. Familiarize students with project planning, scheduling, cost estimation models, and risk management strategies to improve software development effectiveness.
4. Develop an understanding of software testing techniques and levels, ensuring software reliability, performance, and adherence to specifications.

UNIT I

Introduction to Software Engineering: Definition and importance of software engineering, Software characteristics, Software components, Software applications, Software Engineering Principles, Software metrics and measurement,

monitoring and control, software development life cycle (SDLC) models: Waterfall, Agile, Spiral, DevOps, Software project management fundamentals

UNIT II

Software Requirements and Design: Requirements elicitation techniques, requirements analysis, requirements specification, Functional and non-functional requirements, Introduction to Unified Modeling Language (UML)

Software Design and Development: Programming paradigms: procedural, object-oriented, functional, logic; Code quality, Cohesiveness and Coupling, coding standards, and code reviews

UNIT III

Software Project Planning: Project planning and Project scheduling. Software Metrics: Size Metrics like LOC, Token Count, Function Count. Cost estimation using models like COCOMO. Risk management activities.

Testing: Verification and validation, Level of testing: Unit, Integration Testing, Top down and bottom up integration testing, Alpha and Beta testing, functional testing, structural testing.

UNIT IV

Software Maintenance and Advanced Topics: Software maintenance and evolution, Software configuration management, Risk management in software projects, Emerging trends in software engineering

Practical Component

1. Participating in a team project to develop a software application following a chosen SDLC model.
2. Conducting requirements analysis and creating a UML diagram for a system.
3. Implementing a software module and performing unit testing.
4. Reviewing peers' code for quality and adherence to coding standards.

Suggested Readings

1. K. K. Aggarwal & Yogesh Singh, “Software Engineering”, 2 ndEd., New Age International.
2. Software Engineering" by Ian Sommerville
3. R. S. Pressman, “Software Engineering – A practitioner’s approach”, 3rd ed., McGraw Hill Int. Ed.
4. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
5. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin
6. "The Pragmatic Programmer: Your Journey to Mastery" by Andrew Hunt and David Thomas

List of Experiments

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

AI for Image Processing (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
AI for Image Processing	4	3L	0T	1P	-	-

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

Course Objectives

1. Provide a solid foundation in the concepts and techniques of image processing using AI.
2. Equip students with skills to develop and optimize image recognition and analysis algorithms.
3. Introduce state-of-the-art neural networks like CNNs for image classification, segmentation, and object detection.
4. Develop an understanding of deep learning frameworks and libraries like TensorFlow and PyTorch.
5. Encourage innovative applications of AI in fields such as healthcare, autonomous systems, and media.

Course Outcomes

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

1. Understand the principles of AI and its application in image processing tasks.
2. Design and implement neural networks for image classification, segmentation, and detection.
3. Utilize advanced techniques like transfer learning and data augmentation for improved model performance.
4. Analyze and interpret image datasets for various industrial and research applications.
5. Develop innovative AI solutions for real-world image processing challenges.

Unit 1: Fundamentals of Image Processing

Introduction to Digital Image Processing; Image Representation: Pixels, Resolution, and Color Models; Basics of Image Enhancement Techniques: Filtering, Histogram Equalization, and Contrast Adjustment; Image Transformations: Fourier Transform, Wavelets, and Principal Component Analysis (PCA); Introduction to OpenCV and Image Manipulation Libraries in Python.

Unit 2: Artificial Intelligence and Computer Vision

Overview of AI Techniques for Image Processing; Edge Detection and Object Localization; Image Segmentation Techniques: Thresholding, Clustering, and Region-based Methods; Feature Extraction: SIFT, SURF, and ORB; Deep Learning for Vision Tasks: Introduction to Convolutional Neural Networks (CNNs) and Transfer Learning.

Unit 3: Advanced Image Processing with Deep Learning

Understanding Image Classification and Object Detection Models; Pre-trained Models: VGG, ResNet, Inception, and YOLO; Semantic Segmentation using U-Net and Mask R-CNN; Generative Models for Images: GANs and Variational Autoencoders (VAEs); Image Style Transfer and Super-Resolution Techniques.

Unit 4: Applications of AI in Image Processing

Medical Image Processing: Detection and Diagnosis (CT, MRI, X-rays); Facial Recognition Systems and Emotion Detection; Autonomous Vehicles: Lane Detection, Obstacle Detection, and Tracking; Satellite Image Analysis:

Remote Sensing and Land Use Classification; Ethical Considerations in AI-based Image Processing: Bias, Privacy, and Security Concerns.

Practical Component

1. Implementing basic image preprocessing techniques like resizing, normalization, and augmentation.
2. Designing and training convolutional neural networks (CNNs) for image classification tasks.
3. Developing edge detection algorithms using Sobel, Prewitt, or Canny methods.
4. Implementing object detection models using YOLO, SSD, or Faster R-CNN frameworks.
5. Performing image segmentation using U-Net or Mask R-CNN for medical or satellite imagery.
6. Building style transfer applications to modify images using neural networks.
7. Creating and testing face recognition systems using feature extraction and deep learning models.

Suggested Readings

1. Digital Image Processing by Rafael C. Gonzalez and Richard E. Woods (Pearson)
2. Deep Learning for Computer Vision with Python by Adrian Rosebrock (PyImageSearch)
3. Programming Computer Vision with Python by Jan Erik Solem (O'Reilly Media)
4. Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville (MIT Press)

List of Experiments

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

NLP: Techniques and Applications (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
NLP: Techniques and Applications	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the foundational principles of Natural Language Processing (NLP), including language modeling and basic text processing techniques.
2. Apply advanced NLP methodologies, such as Transformer architectures and large-scale pretrained language models, to solve complex language tasks.
3. Evaluate and optimize NLP systems using state-of-the-art ML approaches, including few-shot and zero-shot learning, prompt engineering, transfer learning, and in-context learning.

Course Objectives

1. Introduce students to the fundamental concepts and linguistic foundations of NLP.
2. Equip students with knowledge and skills in modern NLP, including deep neural architectures (RNNs, LSTMs, Transformers) and pretrained large language models (BERT, GPT, T5).
3. Train students to select and apply advanced ML techniques such as few-shot learning, transfer learning, and prompt engineering, enabling them to rapidly adapt NLP models to diverse tasks and domains.
4. Expose students to recent developments, including instruction-tuned language models, multi-modal NLP, reinforcement learning for improved model alignment.
5. Encourage critical thinking about the ethical implications of advanced NLP, ensuring responsible development and deployment of AI-driven language systems.

Unit 1: Fundamentals of NLP

Introduction to Natural Language Processing: Scope, challenges, and real-world applications. Linguistic Essentials: Morphology, syntax, semantics, pragmatics, and their relevance to NLP tasks. Text Processing Techniques: Tokenization, stemming, lemmatization, POS tagging, chunking, and dependency parsing. Traditional Language Modeling: N-grams, smoothing techniques, basic probabilistic approaches. Vector Representations: Bag-of-Words, TF-IDF, distributional semantics, introduction to embeddings (Word2Vec, GloVe).

Unit 2: Deep Learning for NLP

Neural Architectures: RNNs, LSTMs, GRUs for sequence modeling, sequence-to-sequence frameworks. Transformers and Attention Mechanisms: Self-attention, encoder-decoder frameworks, and the Transformer architecture. Large Language Models (LLMs): BERT, GPT, T5, and other pretrained models; fine-tuning techniques and transfer learning. Evaluation Metrics and Model Improvement: BLEU, ROUGE, METEOR, perplexity, accuracy; model interpretability and error analysis. Introduction to Prompt Engineering: Harnessing pretrained models through prompts and in-context learning.

Unit 3: Advanced NLP Techniques

Information Extraction and Semantic Understanding: Advanced NER, relation extraction, event detection, coreference resolution, semantic role labeling. Document-Level NLP: Topic modeling (LDA), advanced text classification, sentiment analysis, and stance detection. Few-shot and Zero-shot Learning: Exploiting pretrained models to perform tasks with minimal task-specific data. Advanced Prompting and Instruction-Tuning: Instruction-based interfaces (e.g., InstructGPT), guiding model behavior through task descriptions, Chain-of-Thought prompting for improved reasoning. Optimization and Model Efficiency: Model compression, quantization, distillation, and adapting LLMs for on-device and real-time NLP tasks.

Unit 4: Multimodal NLP and Ethics in NLP

Multi-modal NLP: Integrating text with images, video, and audio (CLIP, PaLM-E etc), bridging language with visual and other sensory modalities. Reinforcement Learning for Alignment: Reinforcement Learning from Human Feedback (RLHF) to refine model behavior, reduce harmful outputs, and improve user alignment.Ethical and Responsible AI in NLP: Bias detection and mitigation, fairness in language models, handling misinformation, and differential privacy techniques.

Practical Component

1. Implement tokenization, POS tagging, and a simple N-gram language model.
2. Train and evaluate RNN/LSTM-based language models, compare perplexity against baseline N-gram models.
3. Fine-tune a pretrained Transformer (BERT or DistilBERT) on a sentiment analysis or QA task.
4. Use GPT or T5 variants with prompt engineering for zero-shot and few-shot text classification.
5. Experiment with instruction-tuned models, apply chain-of-thought prompting to improve reasoning in a QA scenario.
6. Implement a simple text-image retrieval or captioning task using a vision-language model (e.g., CLIP).
7. Apply model distillation techniques to reduce model size and latency on a chosen NLP task.

Suggested Readings

1. "Speech and Language Processing" by Dan Jurafsky and James H. Martin, Prentice Hall.
2. "Natural Language Processing" by Jacob Eisenstein, MIT Press.
3. "Neural Network Methods for Natural Language Processing" by Yoav Goldberg, Morgan & Claypool.
4. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)" by Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu, Journal of Machine Learning Research (JMLR).
5. "OpenAI and DeepMind Research Papers (GPT, BERT, PaLM etc) and recent ACL/NAACL/EMNLP findings" by various authors.
6. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
7. "Transformers for Natural Language Processing" by Denis Rothman
8. "Attention Is All You Need" by Vaswani et al., seminal paper introducing transformers
9. "GPT-3: Language Models are Few-Shot Learners" by Brown et al., detailing the architecture and capabilities of GPT-3

Health Data Analytics (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Health Data Analytics	4	3L	0T	1P	-	-

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

Course Outcomes

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

1. Understand the fundamental concepts, data sources, and regulatory frameworks that define health data analytics.
2. Employ statistical and machine learning techniques to preprocess, analyze, and derive insights from diverse healthcare datasets.
3. Integrate advanced methods, such as deep learning, NLP, and genomic data analysis, to address complex clinical decision-making challenges.

Course Objectives

1. Introduce the healthcare data landscape, including EHRs, clinical, imaging, genomic, and sensor data, as well as relevant compliance standards (HIPAA, GDPR).
2. Develop proficiency in data preprocessing, exploratory data analysis, and the application of machine learning models for predicting clinical outcomes and improving care quality.
3. Provide hands-on experience with advanced analytic techniques and interpret model results for informed decision-making.
4. Highlight the importance of privacy, security, and ethical considerations in health analytics, ensuring responsible and patient-centered AI applications.
5. Explore cutting-edge trends, technologies, and frameworks for health analytics and their potential to shape the future of healthcare delivery.

Unit 1: Introduction to Health Data Analytics

Healthcare Data Landscape: Electronic Health Records (EHRs), claims data, medical imaging, wearable sensor data, and genomic data. Stakeholders and Systems: Providers, payers, patients, regulatory bodies, and healthcare IT infrastructure. Healthcare Regulations and Standards: HIPAA, GDPR, FHIR; data integration and interoperability challenges. Data Quality and Preprocessing: Handling missing data, normalization, de-identification, data cleaning techniques. Basic Healthcare Statistics: Epidemiological measures, clinical outcome metrics, and public health indicators.

Unit 2: Techniques for Health Data Preprocessing and Analysis

Exploratory Data Analysis (EDA): Visualizing patient cohorts, stratifying populations by disease burden and risk factors. Feature Engineering: Encoding clinical variables, temporal data handling, and deriving risk scores. Predictive Modeling: Applying linear/logistic regression, decision trees, random forests, and gradient boosting for disease prediction and outcome forecasting. Machine Learning (ML) in Healthcare: Introduction to deep learning architectures (CNNs for imaging analysis, RNNs for temporal EHR data), model explainability. Evaluation Metrics: Sensitivity, specificity, PPV, NPV, AUROC, and their clinical significance.

Unit 3: Advanced Topics in Health Analytics

Advanced ML Techniques: Transfer learning, few-shot and zero-shot learning for rare diseases or small datasets. Natural Language Processing: Extracting insights from clinical notes, entity recognition (conditions, medications), summarization of patient records. Genomic data analysis. Federated Learning and Privacy-Preserving Analytics: Training models across distributed datasets while maintaining patient confidentiality.

Unit 4: Population Health Analytics and Imaging Analytics

Population Health Management: Identifying high-risk populations, preventive care strategies, reducing readmissions, and cost containment. Imaging Analytics: Radiomics, computer-aided diagnosis, pathology image interpretation, AI-driven diagnostics. AI-driven clinical decision support systems (CDSS).

Practical Component

1. Use data preprocessing on a synthetic EHR dataset, handle missing values, and perform basic statistical analysis.
2. Build and evaluate a logistic regression or random forest model to predict a healthcare-related metric.
3. Extract medical conditions or medications from clinical notes.
4. Integrate genomic data with clinical variables (if datasets are available).
5. Model Deployment and Interpretability - Create a simple dashboard for clinicians, integrate model explanations.

Suggested Readings

1. "Healthcare Data Analytics" by Chandan K. Reddy, Charu C. Aggarwal.
2. "Health Analytics: Gaining the Insights to Transform Health Care" by Jason Burke, Wiley.
3. "Bioinformatics and Functional Genomics" by Jonathan Pevsner.
4. "Healthcare Analytics for Quality and Performance Improvement" by Trevor L. Strome
5. Research articles and review papers published in leading journals on recent AI-driven DSS implementations and genomic data-driven studies.

List of Experiments

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

Fundamentals of Time Series Analysis (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Fundamentals of Time Series Analysis	4	3L	0T	1P	-	-

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

Course Outcomes

By the end of the course, students will be able to:

1. Identify and describe the structure and components of time series data.
2. Apply statistical and computational techniques for time series modeling.
3. Perform forecasting and evaluate the performance of time series models.
4. Solve domain-specific problems using time series analysis tools and techniques.

Course Objectives

1. To understand the fundamental principles and methods of time series analysis.
2. To equip students with the skills to analyze, model, and forecast time series data.
3. To introduce tools and libraries used for implementing time series models in Python.
4. To explore applications of time series analysis in various domains.

UNIT I

Introduction to Time Series Analysis: Definition and characteristics of time series data, Types of time series (e.g., stationary, non-stationary, seasonal), Components of time series: trend, seasonality, cycle, and noise, Exploratory Data Analysis (EDA) for time series, Data preparation and pre-processing for time series (e.g., scaling, transformations)

UNIT II

Statistical Methods for Time Series: Stationarity and differencing, Autoregressive models (AR), Moving Average models (MA), and ARMA models, ARIMA and SARIMA models for seasonal time series, Model selection criteria: AIC, BIC, Diagnostic checking: Residual analysis and model adequacy tests

UNIT III

Advanced Time Series Modeling: Exponential Smoothing methods (e.g., Simple, Holt's, Holt-Winters), State-space models and Kalman filters, Vector Autoregressive (VAR) and Vector Error Correction Models (VECM), Machine Learning for time series: Random Forests, Gradient Boosting, and LSTMs (Long Short-Term Memory networks), Evaluating forecasting accuracy: MSE, RMSE, MAPE

UNIT IV

Applications and Case Studies: Applications in finance (e.g., stock price prediction), retail (demand forecasting), and healthcare (disease trend analysis), Time series analysis in climate data and energy forecasting, Case study: Forecasting using Python libraries (statsmodels, Prophet, etc.), Challenges in real-world time series data (e.g., missing values, irregular intervals), Emerging trends: Anomaly detection in time series etc

Practical Component

1. Exploratory Analysis and Visualization (Time series decomposition using Python libraries (e.g., statsmodels, Plotting ACF and PACF)
2. Modeling and Forecasting (Implementing ARIMA/SARIMA models on real datasets, Experimenting with exponential smoothing techniques (e.g., Holt-Winters)
3. Machine Learning for Time Series (Developing LSTM models for time series forecasting, Comparing ML-based approaches with traditional models)
4. Real-World Applications (Case studies in finance, retail, and healthcare, Using Prophet and other frameworks for quick forecasting solutions)
5. Handling missing data and irregular intervals and Conducting anomaly detection experiments in time series data

Suggested Readings

1. "Introduction to Time Series and Forecasting" by Peter J. Brockwell and Richard A. Davis
2. "Time Series Analysis and Its Applications" by Robert H. Shumway and David S. Stoffer
3. "Practical Time Series Analysis" by Aileen Nielsen
4. Python libraries: statsmodels, Prophet, scikit-learn, and TensorFlow

List of Experiments

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

Agile Software Development (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Agile Software Development	4	3L	0T	1P	-	-

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

Course Objectives

1. To understand the core principles, values, and practices of Agile Software Development.
2. To explore Agile methodologies such as Scrum, Kanban, and Extreme Programming (XP).
3. To develop skills for Agile planning, estimation, and iterative development.
4. To emphasize the role of testing and quality assurance in Agile frameworks.

Course Outcomes

By the end of the course, students will be able to:

1. Explain the Agile philosophy and its relevance to modern software development.
2. Apply Agile methodologies to plan, execute, and manage software projects.
3. Demonstrate skills in iterative development, sprint planning, and backlog management.
4. Integrate continuous testing and delivery practices within Agile frameworks.
5. Utilize Agile tools for collaboration, tracking, and reporting.

Unit I

Introduction to Agile Development: Agile Manifesto and principles, Comparison of Agile with traditional software development approaches (e.g., Waterfall), Benefits and challenges of Agile development, Overview of Agile methodologies: Scrum, Kanban, XP, Lean Software Development, Agile team roles and responsibilities (e.g., Product Owner, Scrum Master, Team Member)

Unit II

Agile Practices and Frameworks: Scrum framework:Sprint planning, daily stand-ups, sprint reviews, and retrospectives, Product backlog and sprint backlog management, Kanban: Visualizing workflows and limiting work-in-progress (WIP), Extreme Programming (XP): Pair programming, test-driven development (TDD), refactoring, Agile estimation and planning: Story points, velocity, and release planning

Unit III

Agile Testing and Quality Assurance: Principles of Agile testing: Continuous testing and early defect detection, Test-driven development (TDD) and behavior-driven development (BDD), Exploratory testing in Agile, Automated testing in Agile: Tools and strategies, Role of CI/CD (Continuous Integration/Continuous Deployment) in Agile testing, Managing technical debt and maintaining code quality

Unit IV

Tools, Trends, and Case Studies: Agile tools for project management and collaboration: JIRA, Trello, Azure DevOps, Agile metrics and reporting: Burn-down charts, burn-up charts, velocity tracking, DevOps and its integration with Agile, Case studies of successful Agile implementations (e.g., Spotify Model), Recent trends: Scaled Agile Framework (SAFe), Agile in distributed teams, AI in Agile

Practical Component

1. Create a mind map or discussion board reflecting how the Agile Manifesto applies to real-world scenarios.
2. Write user stories for a sample project. Prioritize and create a product backlog.
3. Simulate a Sprint Planning session. Divide a product backlog into sprints, estimating tasks using story points or hours
4. Role-play a Scrum team. Conduct a mock Daily Scrum, Sprint Review, and Sprint Retrospective.
5. Use an open source tool or physical sticky notes to manage a Kanban board. Practice limiting work in progress (WIP).
6. Practice effort estimation techniques for sample tasks.
7. Collaboratively solve a simple coding problem using pair programming techniques.
8. Work on a small project across multiple sprints. Include sprint planning, execution, review, and retrospective.
9. Use Agile project management tools to manage a project.
10. Simulate stakeholder meetings to gather feedback on deliverables after a sprint.

Suggested Readings

1. *Agile Software Development: The Cooperative Game* by Alistair Cockburn
2. *Essential Scrum: A Practical Guide to the Most Popular Agile Process* by Kenneth S. Rubin
3. *Test-Driven Development: By Example* by Kent Beck

List of Experiments

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

Software Reliability (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Software Reliability	4	3L	1T	0P	-	Software Engineering

Course Hours: L: 03 T: 01 P: 00

Course Objectives

1. To provide a thorough understanding of software reliability concepts and their significance.
2. To equip students with the ability to model, predict, and improve software reliability.
3. To apply Software Reliability Growth Models in Software Development
4. To emphasize the Application of Software Reliability Models
5. To introduce the use of tools and techniques for reliability analysis and testing.

Course Outcomes

1. Define and explain the principles and importance of software reliability in modern software systems.
2. Apply reliability metrics and models to evaluate and predict the performance of software systems..
3. Design strategies for improving software reliability through fault prevention, detection, and tolerance.
4. Utilize real-world data and case studies to develop a practical understanding of reliability challenges and solutions

Unit I: Fundamentals of Software Reliability

Definition and importance of software reliability, Software faults, errors, and failures, Reliability vs. availability, Software reliability in the software development lifecycle (SDLC), Factors affecting software reliability, Introduction to probabilistic approaches in reliability

Unit II:

Software Reliability Metrics & Models: Reliability metrics: Mean Time To Failure (MTTF), Mean Time Between Failures (MTBF), failure rate, Basic reliability models: Exponential and Weibull models, Musa, Jelinski-Moranda models, Fault injection and debugging concepts, Limitations of reliability models

Unit III:

Non Homogeneous Poisson Process Models: Musa models- Basic Execution time, Logarithmic Poisson Execution time models - Goel – Okumoto model, Yamada delayed S-shaped model, Imperfect debugging models

Reliability in Testing and Management: Testing Strategies for Reliability Assurance, Stress and Fault Tolerance Testing, Managing Reliability in Development and Deployment, Real-World Data Analysis: Using failure data to predict software performance. Tools: Overview of Software Failure and Reliability Assessment Tool (SFRAT)

Unit IV:

Recent Trends in Software Reliability: AI and machine learning in reliability prediction, Reliability in cloud-based and distributed systems, Reliability in IoT and embedded systems, DevOps practices for enhanced reliability, Emerging tools and platforms for reliability engineering.

Tutorial Component

1. Case Study Discussion: Analyze real-world examples of software failures (e.g., Therac-25, Ariane 5, or Toyota's recall due to software bugs), Identify the causes of faults, errors, and failures in the chosen case study. Discuss how reliability could have been improved.
2. SDLC and Reliability: Map reliability considerations to each phase of the SDLC. Create a checklist of reliability practices for design, development, and testing.
3. Fault and Failure Classification: Use sample software logs to classify errors as faults, failures, or bugs. Analyze the impact of these faults on system reliability.
4. Introduction to Probabilistic Approaches: Perform simple probabilistic calculations for failure rates using given data. Discuss the importance of probability in reliability prediction.

Suggested Readings

1. John D. Musa, Anthony Iannino, Kazuhira Okumoto, “Software Reliability – Measurement, Prediction, Application, Series in Software Engineering and Technology.
2. Michael Lyu, “Handbook of Software Reliability Engineering”, IEEE Computer Society Press
3. John D. Musa, “Software Reliability Engineering”, Tata McGraw Hill, 1999.
4. Patrick P. O'Connor, “Practical Reliability Engineering” , 4th Edition, John Wesley & sons, 2003.
5. Xie M, “Software Reliability Modelling”, World Scientific, Singapore, 1991.
6. Research papers published in relevant international journals

List of tutorial tasks

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

Blockchain Applications (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Blockchain Applications	4	3L	0T	1P	-	Blockchain Essentials

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

Course Outcomes

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

1. Explain the foundational concepts of blockchain technology and evaluate its applications across diverse sectors such as finance, supply chain, and healthcare.
2. Critically assess the design, functionality, and risks associated with decentralized finance (DeFi) platforms, stablecoins, and digital currencies.
3. Explore and implement blockchain-based solutions for identity management, governance, and digital assets (NFTs), considering regulatory, ethical, and environmental impacts.
4. Identify emerging trends in blockchain technology.
5. Propose, design, and analyze use-case-driven blockchain solutions to address real-world problems, ensuring security, transparency, and scalability.

Course Objectives

1. Introduce students to the fundamental principles of blockchain technology, its underlying cryptographic mechanisms, and consensus models.
2. Familiarize students with practical blockchain applications in finance (DeFi, CBDCs), supply chain management, healthcare, and digital identity systems.
3. Equip students with an understanding of smart contracts, NFTs, and DAOs, enabling them to design secure and reliable decentralized applications.

Unit 1: Introduction to Blockchain Applications

Foundational blockchain concepts, Blockchain Applications, Introduction to Decentralized Finance (DeFi), Smart contracts and their use in lending, borrowing, and staking, Stablecoins and Central Bank Digital Currencies (CBDCs), Risks and challenges of DeFi (scalability, security, fraud), Cryptocurrency ecosystems: Wallets, exchanges, and trading platforms., Case studies: Bitcoin as a payment system, Ethereum's smart contract use cases., Case studies of successful implementations (e.g., Bitcoin, Ethereum, Hyperledger).

Unit 2: Blockchain in Supply Chain and Healthcare

Supply Chain Applications: Ensuring transparency and traceability in supply chains, Use cases: Counterfeit prevention, product provenance, Integration with IoT and AI, Case studies: IBM Food Trust, VeChain.

Healthcare Applications: Secure data sharing and patient data privacy, Blockchain in drug traceability and clinical trials, Case studies: MediBloc, Medicalchain.

Unit 3: Digital Identity, NFTs, and Governance

Identity Management: Decentralized Identity (DID) and self-sovereign identity, Blockchain for Know Your Customer (KYC) processes, Case studies: Sovrin Network, Microsoft ION.

Non-Fungible Tokens (NFTs): Token standards (ERC-721, ERC-1155), Use cases: Art, gaming, and digital collectibles, Legal, ethical, and environmental concerns.
Governance Applications: Decentralized Autonomous Organizations (DAOs), Blockchain-based voting systems and transparency in elections, Case studies: Aragon DAOs, blockchain voting experiments.

Unit 4: Emerging Trends
Emerging Trends in Blockchain:Layer 2 solutions: Lightning Network, Optimistic Rollups., Interoperability solutions: Polkadot, Cosmos, Blockchain and AI integration, Quantum computing and blockchain security.

Practical Component
1. Create and deploy a simple smart contract for a financial application (e.g., lending/borrowing on Ethereum).
2. Prototype a product tracking system using blockchain (e.g., Hyperledger or Ethereum).
3. Build a secure medical record-sharing app using blockchain.
4. Create a DID system using tools like uPort or Sovrin.
5. Mint and trade an NFT on Ethereum or Polygon.
6. Simulate DAO governance and build a blockchain-based voting system.
7. Choose an industry-specific problem and develop a blockchain-based solution, such as decentralized crowdfunding platform, blockchain certificate verification system, A supply chain traceability application etc

Suggested Readings
1. Blockchain Basics: A Non-Technical Introduction in 25 Steps by Daniel Drescher
2. "Blockchain Applications: A Hands-On Approach" by Arshdeep Bahga and Vijay Madisetti
3. DeFi and the Future of Finance by Campbell R. Harvey, Ashwin Ramachandran, and Joey Santoro.
4. Blockchain and the Supply Chain: Concepts, Strategies, and Practical Implementation by Nick Vyas.
5. Blockchain in Healthcare: Innovations that Empower Patients, Connect Professionals, and Improve Care by Malaya Zeal.
6. The NFT Handbook: How to Create, Sell and Buy Non-Fungible Tokens by Matt Fortnow and QuHarrison Terry.
7. Mastering Blockchain: Unlocking the Power of Cryptocurrencies, Smart Contracts, and Decentralized Applications by Imran Bashir
8. Bitcoin and Cryptocurrency Technologies by Arvind Narayanan, Joseph Bonneau, et al

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

Cybersecurity with Blockchain (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Cybersecurity with Blockchain	4	3L	0T	1P	-	Fundamentals of Cybersecurity, Blockchain Essentials

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

Course Outcomes

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

1. Understand the integration of blockchain principles with traditional cybersecurity frameworks and evaluate how decentralization, consensus mechanisms, and cryptographic constructs enhance security.
2. Design and implement decentralized identity and access control solutions, ensuring secure key management and robust verification of credentials.
3. Identify and mitigate vulnerabilities in smart contracts and blockchain applications, utilizing auditing, testing, and formal verification tools to ensure reliability and correctness.
4. Employ advanced techniques for data privacy and confidentiality within blockchain ecosystems, leveraging zero-knowledge proofs, secure multi-party computation, and other privacy-preserving methods.
5. Analyze the security implications of various tokenized assets (e.g., stablecoins, security tokens) and integrate privacy-enhancing technologies to develop secure, scalable, and trustworthy blockchain-based solutions.

Course Objectives

1. Provide an in-depth understanding of blockchain security fundamentals and their interplay with cybersecurity principles.
2. Equip students with skills for implementing secure decentralized identity frameworks and managing cryptographic keys in blockchain environments.
3. Enable students to identify, analyze, and remediate vulnerabilities in smart contracts, applying secure coding standards and formal verification techniques.
4. Introduce data privacy challenges in blockchain applications and familiarize students with cutting-edge cryptographic tools for maintaining confidentiality and integrity.

Unit 1: Foundations of Blockchain-Enhanced Security

Revisiting Cybersecurity Fundamentals in a Blockchain Context: CIA triad, threat landscape, and how blockchain properties reinforce or challenge these concepts. Blockchain Security Architecture: Decentralization, consensus mechanisms (PoW, PoS), tamper-evident ledgers, and how these attributes counter traditional cyber threats. Threat Modeling and Risk Assessment in Blockchain Networks: Identifying attack surfaces (51% attacks, Sybil attacks), securing communication channels, and secure key management.

Unit 2: Identity and Access Control with Blockchain

Decentralized Identity Management (DID): Self-sovereign identity, verifiable credentials, DID frameworks (e.g., Sovrin), and blockchain authentication. Access Control Mechanisms: Permissioned vs. permissionless blockchains,

role-based and attribute-based access control models implemented on DLTs. Key Management and Security: Private key handling, hardware security modules (HSM), multi-signature schemes, and hierarchical deterministic wallets.

Unit 3: Secure Smart Contracts and Blockchain Applications

Smart Contract Vulnerabilities: Common exploits (reentrancy, arithmetic overflows, access control flaws), vulnerability scanning tools, and secure coding best practices. Security Auditing and Formal Verification: Automated auditing tools (Mythril, Slither) and formal verification methods (Solidity, Vyper etc) to ensure correctness.

Unit 4: Advanced Techniques, Data Privacy

Data Privacy in Blockchain Applications: Techniques for handling sensitive data on-chain, zero-knowledge proofs (ZKPs), state channels, and confidential transactions. Stablecoins, Security Tokens, and Digital Assets: Analyzing token standards and their security implications. Privacy-Preserving Frameworks: Homomorphic encryption, secure multi-party computation (MPC), and differential privacy methods integrated with blockchain.

Practical Component

1. Secure Key Management and DID Setup
2. Smart Contract Vulnerability Analysis
3. Use automated vulnerability scanning tools (Mythril, Slither) to identify potential issues and apply remediations.
4. Formal Verification of Smart Contracts:
5. Experiment with zero-knowledge proof frameworks to implement a confidential transaction.
6. Integrate a simple state channel or sidechain to enhance privacy and scalability.
7. Perform a threat model and risk assessment for a hypothetical blockchain application scenario. Identify attack surfaces and propose mitigation strategies, documenting the security design.

Suggested Readings

1. "Mastering Blockchain: Unlocking the Power of Cryptocurrencies, Smart Contracts, and Decentralized Applications" by Lorne Lantz and Daniel Cawrey
2. "Ultimate Blockchain Security Handbook: Advanced Cybersecurity Techniques and Strategies for Risk Management, Threat Modeling, Pen Testing, and Smart Contract Defense for Blockchain" by Taha Sajid
3. "Hands-On Blockchain for Python Developers" by Arjuna Sky Kok, for practical coding approaches
4. Research papers from reputed journals and conferences on blockchain security, smart contract verification, and privacy-preserving techniques
5. Online resources and documentation for decentralized identity frameworks (e.g., Sovrin, W3C DID), auditing tools (MythX, Slither), and ZKP libraries

List of Experiments

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

Advanced Game Development (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Advanced Game Development	4	3L	0T	1P	-	Game Development

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

Course Outcomes

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

1. Implement advanced rendering techniques, including PBR, ray tracing, and custom shader pipelines, to achieve high-fidelity visual experiences.
2. Integrate complex physics simulations into games and apply performance profiling and optimization techniques to ensure smooth, responsive gameplay.
3. Architect scalable multiplayer systems, employ secure networking protocols, and utilize CI/CD pipelines and data analytics tools to streamline production and enhance player retention.
4. Develop immersive XR applications, adapt games for multiple platforms and cloud infrastructures, and ensure high-quality user experiences across diverse devices and environments.

Course Objectives

1. Equip students with the knowledge to create visually sophisticated games using modern rendering technologies and custom shader development.
2. Enable students to design and implement realistic physics simulations while maintaining optimal performance through profiling and optimization strategies.
3. Provide students with a comprehensive understanding of multiplayer architectures, secure network communication, automated workflows, and data-driven decision-making in game production.
4. Familiarize students with advanced XR development techniques, cross-platform portability considerations.

Unit 1: Advanced Graphics Techniques

High-Fidelity Rendering Techniques: Physically Based Rendering (PBR), global illumination, ray tracing, and advanced post-processing effects. Shader Programming and Custom Pipelines: Writing custom shaders, and integrating special effects.

Unit 2: Physics Simulations and Performance Assessment

Complex Physics Simulations: Soft-body physics, realistic fluid and cloth simulations, advanced collision algorithms, advanced simulations. Performance Profiling and Optimization: GPU/CPU profiling, LOD strategies, batching, culling, and memory optimization.

Unit 3: Multiplayer, Networking, and Production Pipelines

Multiplayer Game Architectures: Client-server models, peer-to-peer, dedicated servers, load balancing, matchmaking, and latency compensation. Network Protocols and Security: TCP/UDP trade-offs, prediction and interpolation, cheat detection, secure data transmission. Continuous Integration and Deployment (CI/CD): Automated builds, testing pipelines, asset management, version control best practices for large teams. Data Analytics and Telemetry: Collecting in-game analytics, using A/B testing, telemetry-driven updates, player retention analysis.

Unit 4: XR Development and Cross-platform portability

Extended Reality (XR) Development: Advanced VR/AR techniques, haptic feedback, spatial audio, user comfort, and input methods. Cross-Platform and Cloud Gaming: Porting to multiple platforms (consoles, PC, mobile), leveraging cloud gaming infrastructures.

Practical Component

1. Implement ray tracing or advanced post-processing in a chosen game engine, profile and optimize for performance.
2. Build a small-scale multiplayer prototype, handle network synchronization issues, and add basic matchmaking.
3. Integrate a continuous integration pipeline with automated builds and tests, experiment with asset versioning and rollbacks.
4. Create a simple VR or AR experience emphasizing interaction fidelity, comfort, and immersion.
5. Add telemetry events, analyze player data logs,

Suggested Readings

1. "Game Engine Architecture" by Jason Gregory
2. "Real-Time Rendering" by Tomas Akenine-Moller et al.
3. Latest GDC (Game Developers Conference) and SIGGRAPH proceedings, relevant research papers, and industry white papers on advanced topics in rendering, AI, and XR.
4. Documentation of industry-standard physics engines.

List of Experiments

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

Augmented and Virtual Reality Systems Design (DSE-4/GE-6)

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Eligibility criteria	Prerequisite of the course (if any)
		Lecture	Tutorial	Practical		
Augmented and Virtual Reality Systems Design	4	3L	0T	1P	-	Foundations of Augmented and Virtual Reality

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

Course Outcomes

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

1. Understand the basic principles and technologies behind augmented and virtual reality systems.
2. Design and implement interactive AR/VR applications using modern hardware and software tools.
3. Apply knowledge of 3D modeling, rendering, and computer vision to create immersive virtual environments.
4. Develop AR applications that integrate the physical world with digital content through sensors and mobile devices.
5. Analyze the use of AR/VR in various industries, including gaming, education, healthcare, and retail.
6. Understand the challenges and limitations of AR/VR technologies and develop strategies to overcome them.
7. Evaluate the effectiveness of AR/VR in enhancing user experience and engagement.

Course Objectives

1. Provide students with a foundational understanding of augmented and virtual reality technologies and their applications.
2. Equip students with the skills to develop AR/VR applications using appropriate hardware platforms and software frameworks.
3. Introduce students to advanced topics such as 3D rendering, gesture recognition, and immersive experiences.
4. Help students design and create interactive AR experiences using real-world sensors and mobile applications.
5. Enable students to analyze the diverse applications of AR/VR in various fields and industries.
6. Discuss the ethical and social implications of AR/VR technologies and their influence on user behavior.
7. Prepare students to overcome technical challenges in AR/VR systems and deliver high-quality user experiences.

Unit 1: Introduction to AR and VR Systems

Overview of Augmented Reality (AR) and Virtual Reality (VR); Key Differences and Applications; Hardware and Software Components: Head-mounted Displays, Sensors, and Input Devices; Basics of 3D Coordinate Systems and Transformations; Introduction to AR/VR Development Tools: Unity, Unreal Engine, and ARKit/ARCore.

Unit 2: AR/VR Interaction and User Experience Design

Design Principles for Immersive Interfaces; Gesture Recognition, Eye-tracking, and Motion Capture; Techniques for Creating Intuitive AR/VR Experiences; Haptic Feedback Systems; Evaluation of User Experience (UX) in AR/VR Environments; Case Studies of Successful AR/VR Applications.

Unit 3: AR/VR Development Techniques

3D Modeling and Scene Design for AR/VR; Integration of Real and Virtual Content: Rendering and Registration; AR/VR Content Development Using Unity/Unreal Engine; Marker-based and Markerless AR Systems; Optimizing Performance for Real-time Interaction in AR/VR.

Unit 4: Advanced Topics and Applications in AR/VR

Mixed Reality and the Future of Immersive Technologies; Use of AI in AR/VR: Scene Understanding and Object Recognition; Applications in Healthcare, Education, and Entertainment; Challenges and Ethical Considerations in AR/VR Systems: Privacy, Security, and Accessibility; Emerging Trends: Spatial Computing and Metaverse Development.

Practical Component

1. Creating and deploying basic AR applications using mobile devices and AR frameworks like ARCore or ARKit.
2. Developing a virtual reality environment using Unity and Oculus Rift or HTC Vive.
3. Designing an interactive 3D virtual world and implementing object manipulation in VR.
4. Building an AR application that uses real-time tracking to place virtual objects in the physical world.
5. Implementing gesture recognition for interacting with AR/VR environments.
6. Conducting usability testing of AR/VR applications and optimizing them for improved user experiences.
7. Evaluating the impact of VR and AR applications in education, healthcare, or entertainment through small prototype projects.

Suggested Readings

1. Augmented Reality: Principles and Practice by Dieter Schmalstieg and Tobias Hollerer (Pearson)
2. Virtual Reality by Steven M. LaValle (Cambridge University Press)
3. Learning Virtual Reality: Developing Immersive Experiences and Applications for Desktop, Web, and Mobile by Tony Parisi (O'Reilly Media)
4. Augmented Reality: Where We Will All Live by Jon Peddie (Springer)
5. Programming 3D Applications with HTML5 and WebGL by Tony Parisi (O'Reilly Media)
6. 3D User Interfaces: Theory and Practice by Doug Bowman, Ernst Kruijff, Joseph LaViola, and Ivan Poupyrev (Addison-Wesley)
7. Unity Virtual Reality Projects by Jonathan Linowes (Packt Publishing)