Appendix-62
AC dated 12.07.2024

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

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

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

Course Structure of B.Tech. (CSE) Second Year
Semester – III

S.No.	Course	Course Title	Credits*			Total Credits
			L	T	P	
1.	DSC-7	Analysis and Design of Algorithms	3	0	1	4
2.	DSC-8	Digital System Design	3	0	1	4
3.	DSC-9	Database Management Systems	3	0	1	4
4.	DSE-1 or GE-3	Select a course from the specified list of DSE-1 or Select a course from the specified list of GE-3				4
5.	AEC	Select a course from the specified list of AECs				2
6.	SEC or IAPC	Choose one SEC or Internship/Apprenticeship/Project/Community Outreach (IAPC)				2
7.	VAC	Select a course from the specified list of VACs				2
Total Credits						22

Semester – IV

S.No.	Course	Course Title	Credits*			Total Credits
			L	T	P	
1.	DSC-10	Operating System	3	0	1	4
2.	DSC-11	Software Engineering	3	0	1	4
3.	DSC-12	Computer System Architecture	3	0	1	4
4.	DSE-2 or GE-4	Select a course from the specified list of DSE-2 or Select a course from the specified list of GE-4				4
5.	AEC	Select a course from the specified list of AECs				2
6.	SEC or IAPC	Choose one SEC or Internship/Apprenticeship/Project/Community Outreach (IAPC)				2
7.	VAC	Select a course from the specified list of VACs				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 Second Year

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

List of SECs offered by the department in Second Year

S.No.	Semester	Paper Title
1.	III	Backend Development
2.	IV	App Development using Flutter

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

Specializations and Minors offered by the department

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**
III	GE-3	Fundamentals of DBMS	Probability and Statistics for Computer Science	Probability and Statistics for Computer Science	Software Requirement Engineering	Fundamentals of Cybersecurity	Foundations of Augmented and Virtual Reality
IV	GE-4	Fundamentals of Operating System	Fundamentals of Data Analytics	Fundamentals of Data Analytics	Object Oriented Software Engineering	Cryptography Essentials	3D Graphics and Rendering

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

Detailed Syllabus of Discipline Specific Core (DSC) courses for B.Tech. (CSE) - Semester 3

ANALYSIS AND DESIGN OF ALGORITHMS (DSC-7)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Analysis and Design of Algorithms	4	3L	0T	1P	Data Structures

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

Course Outcomes

At the end of this course, students will:

1. Develop the ability to analyze the running time and prove the correctness of basic algorithms.
2. Be able to design efficient algorithms for moderately difficult computational problems, using various algorithm design techniques taught in the course.
3. Be able to prove the hardness of NP-Hard problems using simple reductions.
4. Be able to do performance analysis of simple approximation algorithms.

Course Objectives

1. Familiarize the student with some basic algorithms and their efficiency analysis.
2. Provide a detailed introduction to different algorithm design paradigms with illustrative problems.
3. Learn and implement dynamic programming and greedy algorithms.
4. Familiarize the student with graphs, computationally hard problems and tackling them using approximation algorithms.

Unit 1: Fundamentals of Algorithmic Problem Solving

Introduction to Algorithms and their Importance, Understanding the Role of Algorithms in Computing, Algorithmic Paradigms: Overview and Classification, Basic Analysis of Algorithms: Time and Space Complexity, Asymptotic Notations: Big O, Big Theta, Big Omega

Unit 2: Divide and Conquer Algorithms

Principles of Divide and Conquer, Classic Examples: Binary Search, Merge Sort, Quick Sort, Analysis of Divide and Conquer Algorithms, Application in Multiplication of Large Integers and Matrix Multiplication, Master Theorem for Divide and Conquer Recurrences

Unit 3: Dynamic Programming and Greedy Algorithms

Introduction to Dynamic Programming, Key Principles and Comparison with Divide and Conquer, Examples: Fibonacci Series, Longest Common Subsequence, Knapsack Problem etc, Greedy Algorithm Fundamentals, Greedy Algorithm Examples: Activity Selection, Kruskal's and Prim's Algorithms for MST etc

Unit 4: Graph Algorithms and Advanced Techniques

Graph Representation and Traversal: BFS and DFS, Shortest Path Algorithms: Dijkstra's, Bellman-Ford, Network Flow Problems: Ford-Fulkerson Method, Introduction to NP-Completeness and Intractable Problems, Approximation Algorithms for NP-Hard Problems

Suggested Readings

1. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
2. "Algorithm Design" by Jon Kleinberg and Éva Tardos
3. "The Algorithm Design Manual" by Steven S. Skiena
4. "Grokking Algorithms: An illustrated guide for programmers and other curious people" by Aditya Bhargava

Practical Component

1. Implementation of various algorithms in a high-level language
2. Case Studies: Analyzing real-world problems and designing algorithms
3. Comparative analysis of different algorithmic approaches for the same problem
4. Developing a mini-project using a combination of algorithmic techniques

List of Experiments

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

DIGITAL SYSTEM DESIGN (DSC-8)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Digital System Design	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. Develop a solid understanding of the fundamental principles and concepts of digital logic design, including Boolean algebra, logic gates, combinational and sequential circuits.
2. Gain the ability to design, analyze, and implement combinational and sequential logic circuits using both theoretical methods and hardware description languages (HDLs) such as VHDL and Verilog.
3. Achieve proficiency in using hardware description languages (HDLs) for modeling, simulating, and synthesizing digital systems.

Course Objectives

1. Understand the principles of digital logic design.
2. Learn to design, analyze, and implement combinational and sequential logic circuits.
3. Gain proficiency in using hardware description languages for digital system design.
4. Develop skills in designing and testing digital systems.

Unit 1: Fundamentals of Digital Logic

Introduction to Digital Systems: Digital vs. Analog systems, Binary numbers and arithmetic, Logic gates and their characteristics, Boolean Algebra and Logic Simplification: Boolean algebra principles, DeMorgan's Theorems, Simplification of Boolean expressions using algebraic methods, Combinational Logic Design: Design and analysis of combinational circuits, Karnaugh maps for simplification, Implementation using logic gates, Standard Combinational Modules: Multiplexers and demultiplexers, Encoders and decoders, Comparators and adders.

Unit 2: Sequential Logic Design

Differences between combinational and sequential circuits, Flip-flops: SR, D, JK, and T flip-flops, Design and Analysis of Sequential Circuits: State diagrams and state tables, Design of counters and registers, Analysis and design of synchronous sequential circuits.

Unit 3: Hardware Description Languages

Overview of VHDL and Verilog, Syntax and semantics of HDLs, Writing simple HDL programs, Combinational Logic Design using HDLs, Sequential Logic Design using HDLs.

Unit 4: Advanced Digital System Design

Design of Arithmetic Circuits: Binary arithmetic: addition, subtraction, multiplication, and division, Design of ALUs (Arithmetic Logic Units), Floating-point arithmetic circuits, Data Path and Control Path Design: Design of data paths, Control path design methodologies, Integration of data path and control path, Digital System Design Flow: Overview

of the design flow from specification to implementation, Design verification and validation, Case studies of digital system design projects

Suggested Readings

1. "Digital Design" by M. Morris Mano and Michael D. Ciletti
2. "Fundamentals of Logic Design" by Charles H. Roth Jr. and Larry L. Kinney
3. "Modern Digital Electronics" by R. P. Jain (TMH)
4. "Digital Principles and Application" by Malvino & Leach (TMH)

Practical Component

1. Develop proficiency in using VHDL/Verilog for digital system design.
2. Design, simulate, and implement basic logic gates (AND, OR, NOT, NAND, NOR, XOR) using hardware description languages (VHDL/Verilog)
3. Design and implement combinational circuits such as adders, multiplexers, and decoders using HDLs.
4. Design and implement sequential circuits such as flip-flops, counters, and shift registers using HDLs.

List of Experiments

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

DATABASE MANAGEMENT SYSTEMS (DSC-9)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Database Management Systems	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. Define program-data independence, data models for database systems, database schema and database instances.
2. Recall Relational Algebra concepts, and use it to translate queries to Relational Algebra.
3. Identify Structure Query Language statements used in creation and manipulation of database.
4. Identify the methodology of conceptual modeling through the Entity Relationship model.
5. Identify the methodology of the logical model and also identify the methodology of the physical model.
6. Analyze and design a real database application and develop and evaluate a real database application using a database management system.

Course Objectives

1. Provide an introduction to relational database systems.
2. Familiarize the student with the relational model, SQL, transactions, database design, and advanced concepts such as NoSQL, data warehousing, data mining, and distributed databases.

Unit 1: Introduction to DBMS

Overview of Database Systems, Database System Architecture, Data Models: Hierarchical, Network, Relational, Object-oriented, Data Independence, Database Languages and Interfaces

Unit 2: Data Modeling

Entity-Relationship Model: Entities, Attributes, Relationships, Enhanced E-R Model, Conversion of ER Model to Relational Model, Normalization: 1NF, 2NF, 3NF, BCNF, 4NF, 5NF, Denormalization Techniques

Unit 3: SQL and Database Design

SQL: DDL, DML, DCL, Advanced SQL: Joins, Subqueries, Views, Indexes, Stored Procedures and Triggers, Transaction Management: ACID Properties, Concurrency Control, Database Security and Authorization

Unit 4: Advanced Topics

NoSQL Databases: Key-Value, Document, Column-Family, Graph, Data Warehousing and Data Mining, Distributed Databases and Object-Oriented Databases, Big Data Technologies and Trends, Database Backup and Recovery

Suggested Readings

1. "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, S. Sudarshan

2. "Fundamentals of Database Systems" by Ramez Elmasri, Shamkant B. Navathe
3. "SQL: The Complete Reference" by James Groff , Paul Weinberg, Andy Oppel
4. "Expert MySQL" by Charles Bell

Practical Component

1. Design and implement a database for a real-world application
2. SQL query writing, optimization, and performance tuning
3. Implementing a NoSQL database solution

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 Elective (DSE) Courses for B.Tech. (CSE) - Semester 3

OBJECT-ORIENTED PROGRAMMING (DSE-1)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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 1: 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 2: Advanced OOP Concepts

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

Unit 3: 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 4: 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

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

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

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-1)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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 1: 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 2: 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 3: 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 4: 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.

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
3. "Probability, Random Variables, and Stochastic Processes" by S. U. Papoulis, A. Pillai

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

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-1)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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 1: HTML and CSS

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

Unit 2: 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 3: Responsive Design

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

Unit 4: 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).

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

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

List of Experiments

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

DISCRETE STRUCTURES (DSE-1)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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 1: 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 2: 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 3: Algebraic Structures

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

Unit 4: 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.

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

Practical Component

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

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 Core (DSC) courses for B.Tech. (CSE) - Semester 4

OPERATING SYSTEM (DSC-10)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

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

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

Course Outcomes

At the end of this course, students will:

1. Develop an understanding of how an operating system functions as a middle layer between the hardware of a computer and the user.
2. Appreciate the design issues and concepts underlying some of the well known operating systems.
3. Compare the pros and cons of various design options and various issues involved in the design of a large software.
4. Understand the concepts of process management, memory management, and file systems.

Course Objectives

1. Introduce the student to the basic concepts involved in the design and implementation of an operating system.
2. Familiarize the student with important modules of operating systems like process management, memory management, file systems, synchronization primitives and exception handling.

Unit 1: Introduction to Operating Systems

Overview of Operating Systems, OS structure: Monolithic, Microkernel, Layered, Modular, System calls and System programs, Bootstrapping and BIOS, OS services and User Interface.

Unit 2: Process Management

Process Concept, Process Scheduling, Inter-process communication (IPC), Threads and Concurrency, Deadlocks: Detection, Prevention, Avoidance, Synchronization mechanisms: Semaphores, Monitors.

Unit 3: Memory Management

Memory hierarchy, RAM vs Cache, Paging, Segmentation, and their combination, Virtual Memory: Demand Paging, Page Replacement Algorithms, Memory Allocation strategies, Memory Protection and Sharing.

Unit 4: File Systems and Security

File concepts, Access methods, Directory structure, File system mounting, File sharing, Protection, Disk scheduling algorithms, Security: Authentication, Encryption, Firewalls, Case studies: UNIX/Linux, Windows OS

Suggested Readings

1. "Operating System Concepts" by Abraham Silberschatz, Peter B. Galvin, Greg Gagne
2. "Modern Operating Systems" by Andrew S. Tanenbaum, Herbert Bos

Practical Component

1. Simulation of OS components (Scheduler, Memory Management)
2. Shell scripting and System programming
3. Development of simple synchronization problems
4. Case studies: Comparative OS analysis

List of Experiments

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

SOFTWARE ENGINEERING (DSC-11)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Software Engineering	4	2L	0T	2P	-

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

Course Outcomes

1. Students will be working in teams and by the end of the semester they will gain experience in handling various phases of software development. During the project, students will assume various participant roles like developer, project manager, architect etc, in addition to thinking and deciding on the design of various modules that constitute the software.
2. Students will learn to follow standard practices throughout the project and to stick to the specification of interfaces between different software modules so that a big software being developed by multiple persons can work together as required.

Course Objectives

1. Give students hands-on experience in designing and developing a reasonably big software - big enough that it warrants multiple phases (including architecture, design, and implementation) and multiple developers.
2. Promote collaboration and teamwork in students, with each team building a complete software product.
3. Expose the student to various phases of software development which includes requirements gathering and specification, systematic design using commonly seen design patterns, coding, component-wise testing, integration testing, version control, and software cost estimation.

Unit 1: Software Development Life Cycle

Overview of Software Engineering, Software Development Models: Waterfall, V-Model, Spiral, Agile methodologies: Scrum, Kanban, XP, Requirement Engineering: Elicitation, Analysis, Specification.

Unit 2: Software Design and Modeling

Software Design and Architecture, UML Diagrams: Use Case, Class, Sequence, State, Design Patterns: MVC, Singleton, Factory, Software Architecture: Layered, Client-Server, Microservices, API Design and Development.

Unit 3: Software Testing and Quality Assurance

Testing levels: Unit, Integration, System, Acceptance, Test Design Techniques: Black-box, White-box, Grey-box, Test Automation and Tools, Software Maintenance and Evolution, Software Metrics and Quality Attributes, Quality Assurance: Standards, ISO/IEC 9126.

Unit 4: Project Management

Project Planning and Estimation, Risk Management, Configuration Management, Software Project Management tools, Case studies: Successful and Failed Software Projects.

Suggested Readings

1. "Software Engineering" by K.K. Aggarwal, Yogesh Singh
2. "Software Engineering: A Practitioner's Approach" by Roger S. Pressman, Bruce Maxim
3. "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development" by Craig Larman

Practical Component

1. Software requirement and design exercises
2. Implementation of a small-scale software project
3. Software testing and debugging practices
4. Project management simulation

List of Experiments

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

COMPUTER SYSTEM ARCHITECTURE (DSC-12)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Computer System Architecture	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. State the history and development of modern computer systems.
2. Interpret data representation and computer arithmetic.
3. Determine the key aspects of microarchitecture and instruction set architecture.
4. Estimate performance of computer systems and suggest methods for performance enhancement.
5. Specify the importance of memory hierarchy for efficient memory design and virtual memory to overcome memory wall.
6. Predict performance of IO subsystems.

Course Objectives

1. Familiarize the student with the basics of organizational and architectural issues of a digital computer.
2. Familiarize the student with performance issues in processor and memory design of a digital computer.
3. Make the student understand various data transfer techniques in digital computers.
4. Familiarize the student with processor performance improvement using instruction level parallelism.

Unit 1: Basics of Computer Architecture

Overview of computer architecture and organization, Von Neumann architecture and its limitations, Instruction set architecture, RISC vs CISC, Data representation: Binary, Hexadecimal, ASCII, Computer arithmetic: Integer and floating-point operations.

Unit 2: Data Representation and Processor Architecture

CPU architecture: ALU, Control Unit, Registers, Instruction cycle, Pipelining, Superscalar processors, Microprogramming and Control Unit design, Cache memory: Mapping techniques, Replacement policies, Virtual memory: Paging, Segmentation, TLB.

Unit 3: Memory Systems

Primary and Secondary memory, RAM technologies: SRAM, DRAM, SDRAM, ROM, PROM, EPROM, EEPROM, Memory hierarchy and optimization, RAID levels and their applications.

Unit 4: I/O Systems and Data Communication

I/O devices and their interfaces, Direct Memory Access (DMA), Buses: Types, Protocols, Bus arbitration, Serial and Parallel communication, Basics of computer networks and data transmission.

Suggested Readings

1. "Computer Organization and Design: The Hardware/Software Interface" by David A. Patterson and John L. Hennessy
2. "Computer Systems: A Programmer's Perspective" by Randal E. Bryant and David R. O'Hallaron

Practical Component

1. Simulation of CPU operations
2. Assembly language programming exercises
3. Design and simulation of simple processor circuits
4. Case studies on memory and I/O management

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 Elective (DSE) Courses for B.Tech. (CSE) - Semester 4

FOUNDATIONS OF DATA ANALYSIS (DSE-2)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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.

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

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.

List of Experiments

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

COMPUTER GRAPHICS (DSE-2)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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.

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

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.

List of Experiments

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

INTRODUCTION TO IOT (DSE-2)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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).

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

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.

List of Experiments

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

OPTIMIZATION TECHNIQUES (DSE-2)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
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).

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

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.

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 department in Second Year

1. **Minor in CSE (Offered to ECE and EE)**
    a. DSE-1 / GE-3: Fundamentals of DBMS
    b. DSE-2 / GE-4: Fundamentals of Operating System

2. **Minor/Specialization in Artificial Intelligence & Machine Learning (Offered to CSE, ECE, and EE)**
    a. DSE-1 / GE-3: Probability and Statistics for Computer Science
    b. DSE-2 / GE-4: Fundamentals of Data Analytics

3. **Minor/Specialization in Data Science (Offered to CSE, ECE, and EE)**
    a. DSE-1 / GE-3: Probability and Statistics for Computer Science
    b. DSE-2 / GE-4: Fundamentals of Data Analytics

4. **Minor/Specialization in Software Engineering (Offered to CSE, ECE, and EE)**
    a. DSE-1 / GE-3: Software Requirement Engineering
    b. DSE-2 / GE-4: Object Oriented Software Engineering

5. **Minor/Specialization in Blockchain and Cybersecurity (Offered to CSE, ECE, and EE)**
    a. DSE-1 / GE-3: Fundamentals of Cybersecurity
    b. DSE-2 / GE-4: Cryptography Essentials

6. **Minor/Specialization in Augmented Reality / Virtual Reality (Offered to CSE, ECE, and EE)**
    a. DSE-1 / GE-3: Foundations of Augmented and Virtual Reality
    b. DSE-2 / GE-4: 3D Graphics and Rendering

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 3

FUNDAMENTALS OF DBMS (DSE-1 / GE-3)

CREDIT DISTRIBUTION 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 DBMS	4	2L	0T	2P	-	-

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

Unit 1: Introduction to Database Systems

Evolution of database systems, Types of database systems: relational, NoSQL, NewSQL, Database system architecture and components, Data models: ER model, relational model

Unit 2: SQL and Database Design

Basics of SQL: DDL, DML, DCL, Normalization and denormalization: concepts and techniques, Database design process: from ER diagrams to relational schemas, Indexing, constraints, and triggers

Unit 3: Advanced Database Features

Stored procedures and views, Introduction to transaction management and concurrency control, Database security: authorization, authentication, encryption, Introduction to data warehousing and OLAP

Unit 4: NoSQL Databases and Big Data

Introduction to NoSQL databases: key-value, document, column-family, graph, Use cases for NoSQL databases and how they complement relational databases, Basics of big data technologies: Hadoop, Spark, Trends in database technologies

Suggested Readings

1. "Database System Concepts" by Abraham Silberschatz, Henry Korth, and S. Sudarshan
2. "SQL in 10 Minutes, Sams Teach Yourself" by Ben Forta
3. "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" by Pramod J. Sadalage and Martin Fowler

Practical Component

1. Designing and implementing a relational database schema based on a case study.
2. Writing SQL queries to manipulate and retrieve data.

3. Implementing a simple application using a NoSQL database.
4. Conducting a small-scale data warehousing project.

List of Experiments

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

PROBABILITY AND STATISTICS FOR COMPUTER SCIENCE (DSE-1 / GE-3)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Probability and Statistics for Computer Science	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 solid understanding of basic probability concepts, including conditional probability, Bayes' theorem, and various probability distributions (Binomial, Poisson, Normal), and calculate expectation and variance for different random variables.
2. Analyze and interpret statistical data using point and interval estimation, perform hypothesis testing (Z-test, T-test, Chi-square test), and understand the implications of the Central Limit Theorem and sampling distributions.
3. Apply regression analysis (linear and multiple), conduct ANOVA, and use non-parametric tests to draw meaningful inferences from data, along with performing time series analysis and forecasting.
4. Utilize statistical software tools (R or Python) for data analysis and visualization, perform simulations for probabilistic models, and critically evaluate case studies to understand the application of statistical methods in computer science.

Course Objectives

1. Impart a comprehensive understanding of the fundamental concepts in probability theory and statistics, including the ability to handle discrete and continuous random variables, and probability distributions.
2. Enable students to apply probability and statistical methods to analyze and solve problems in computer science, utilizing tools such as R or Python for statistical analysis and data visualization.
3. Train students in using statistical methods for hypothesis testing, regression analysis, and non-parametric tests, and to apply these techniques to real-world computer science applications such as queueing theory, Markov chains, and reliability analysis.
4. Expose students to advanced statistical applications in computer science, including Monte Carlo methods, time series analysis, statistical learning, and data mining, and to understand their practical implications through case studies and simulations.

Unit 1: 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 2: 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 3: 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 4: 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

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

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

List of Experiments

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

SOFTWARE REQUIREMENT ENGINEERING (DSE-1 / GE-3)

CREDIT DISTRIBUTION 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 Requirement 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. Demonstrate a comprehensive understanding of the requirements engineering process, including the elicitation, analysis, specification, and validation of functional, non-functional, system, and user requirements.
2. Conduct effective requirement elicitation using techniques such as interviews, surveys, use case scenarios, workshops, and prototyping, ensuring the accurate and complete gathering of requirements from stakeholders.
3. Utilize various modeling techniques (use case diagrams, activity diagrams, state diagrams) to analyze and specify software requirements, and prepare detailed and clear software requirement specification documents.
4. Apply validation techniques to ensure the accuracy and feasibility of gathered requirements, manage changes to requirements effectively, and maintain traceability throughout the software development lifecycle.

Course Objectives

1. Provide a thorough understanding of the significance of requirements engineering in the software development process, covering the essential stages of elicitation, analysis, specification, and validation of software requirements.
2. Equip students with practical knowledge and skills in various requirement elicitation techniques, such as interviews, surveys, workshops, and prototyping, ensuring comprehensive and accurate gathering of software requirements.
3. Enable students to model, analyze, and specify software requirements effectively, using tools and techniques like use case diagrams, activity diagrams, and state diagrams, and to prepare clear and unambiguous software requirement specification documents.
4. Introduce students to the methods for validating requirements, managing requirement changes, and ensuring traceability throughout the requirements engineering process, promoting the development of high-quality software systems.

Unit 1: Introduction to Requirements Engineering

Importance of requirements engineering in software development, Requirements engineering process: Elicitation, Analysis, Specification, and Validation, Types of requirements: Functional, Non-functional, System, and User requirements.

Unit 2: Requirement Elicitation Techniques

Interviews, surveys, questionnaires, and use case scenarios, Requirements workshops, brainstorming sessions, and role-playing, Prototyping, observation, and document analysis.

Unit 3: Requirement Analysis and Specification

Requirements modeling: Use case diagrams, activity diagrams, and state diagrams, Writing unambiguous and clear requirements, Software requirement specification document: Structure and importance.

Unit 4: Requirement Validation and Management

Techniques for validating requirements, Requirement change management, Traceability in requirements engineering.

Suggested Readings

1. "Software Requirements" by Karl E. Wiegers and Joy Beatty
2. "Mastering the Requirements Process: Getting Requirements Right" by Suzanne Robertson and James Robertson
3. "Requirements Engineering: Fundamentals, Principles, and Techniques" by Klaus Pohl

Practical Component

1. Conducting interviews and surveys for requirement gathering.
2. Developing use case and activity diagrams for a sample application.
3. Writing a software requirement specification document for a given case study.
4. Implementing requirement validation techniques on gathered requirements.
5. Using software tools for requirements management and traceability.

List of Experiments

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

FUNDAMENTALS OF CYBERSECURITY (DSE-1 / GE-3)

CREDIT DISTRIBUTION 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 Cybersecurity	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 clear understanding of cybersecurity principles (confidentiality, integrity, availability), recognize different types of cyber threats and attacks, and grasp the basics of symmetric and asymmetric encryption and hash functions.
2. Apply network security principles, configure and manage firewalls, IDS, and IPS, understand the role and configuration of VPNs, and design secure network architectures.
3. Integrate security into the software development lifecycle, identify and mitigate common web application vulnerabilities, and implement data protection techniques such as encryption, tokenization, and data masking.

Course Objectives

1. Provide students with a solid foundation in cybersecurity principles, types of cyber threats, and the basics of cryptography, along with an understanding of cybersecurity laws and ethics.
2. Equip students with knowledge of network security protocols, devices, and techniques, including firewalls, IDS/IPS, VPNs, and secure network design principles.
3. Enable students to understand and implement security measures throughout the software development lifecycle, identify and mitigate web application vulnerabilities, and apply data protection techniques.

Unit 1: Introduction to Cybersecurity

Principles of cybersecurity: confidentiality, integrity, availability (CIA), Types of cyber threats and attacks: malware, phishing, Denial of Service (DoS), Cybersecurity laws and ethics, Introduction to cryptography: symmetric and asymmetric encryption, hash functions.

Unit 2: Network Security

Basics of network protocols and devices, Firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS), Virtual Private Networks (VPNs) and their importance, Secure network design principles.

Unit 3: Application and Data Security

Security in software development lifecycle (SDLC), Web application security vulnerabilities: SQL injection, XSS, CSRF, Data protection techniques: encryption, tokenization, data masking, Cloud security fundamentals and best practices.

Unit 4: Incident Response and Cybersecurity Frameworks

Basics of incident response: preparation, detection, containment, eradication, recovery, Introduction to cybersecurity frameworks: NIST, ISO/IEC 27001, Role of physical security in cybersecurity, Future trends in cybersecurity.

Suggested Readings

1. "Cybersecurity Essentials" by Charles Brooks, Christopher Grow, Philip Craig, and Donald Short
2. "The Art of Invisibility" by Kevin Mitnick
3. "Cybersecurity and Cyberwar: What Everyone Needs to Know" by P.W. Singer and Allan Friedman

Practical Component

1. Setting up and configuring a firewall and a basic IDS.
2. Conducting a simple penetration test on a web application to identify vulnerabilities.
3. Implementing basic encryption and hash functions for data security.
4. Developing an incident response plan for a given scenario.

List of Experiments

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

FOUNDATIONS OF AUGMENTED AND VIRTUAL REALITY (DSE-1 / GE-3)

CREDIT DISTRIBUTION 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 Augmented and Virtual Reality	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 foundations of AR and VR
2. Apply principles of immersive design and cognitive aspects to create user interfaces and interactions, utilize storytelling and content creation techniques, and address ethical considerations in the development of AR and VR applications.
3. Evaluate case studies of AR and VR in various industries, understand the role of AR and VR in education and training, assess the social implications and business models, and navigate regulatory and legal considerations.

Course Objectives

1. Provide students with a comprehensive understanding of the history, evolution, and core technologies of Augmented Reality (AR) and Virtual Reality (VR), along with an exploration of current platforms and future trends.
2. Teach the principles of immersive design, cognitive aspects, user interface and interaction design, and the role of storytelling and content creation, while addressing ethical considerations in AR and VR.
3. Familiarize students with AR and VR development tools, basic 3D modeling, scripting and programming foundations, and the development, testing, and debugging of simple AR and VR applications.

Unit 1: Introduction to AR and VR

History and Evolution of AR and VR, Key Differences and Similarities, Core Technologies Behind AR and VR, Major AR and VR Platforms, Future Trends and Potential.

Unit 2: AR and VR User Experience

Principles of Immersive Design, Cognitive Aspects of AR and VR, User Interface and Interaction Design, Storytelling and Content Creation, Ethical Considerations in AR and VR.

Unit 3: AR and VR Development Basics

Introduction to AR and VR Development Tools, Basic 3D Modeling for AR and VR, Scripting and Programming Foundations, Simple AR and VR Application Development, Testing and Debugging AR/VR Applications.

Unit 4: AR and VR in Practice

Case Studies in Various Industries, AR and VR for Education and Training, Social Implications of AR and VR, Business Models in AR and VR, Regulatory and Legal Considerations.

Suggested Readings

1. "Augmented Reality: Principles and Practice" by Dieter Schmalstieg and Tobias Hollerer
2. "Understanding Virtual Reality: Interface, Application, and Design" by William R. Sherman and Alan B. Craig
3. "Learning Virtual Reality: Developing Immersive Experiences and Applications for Desktop, Web, and Mobile" by Tony Parisi

Practical Component

1. AR/VR Development Workshops and Lab Sessions
2. Prototype Development Projects
3. Case Study Analysis and Presentations

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 4

FUNDAMENTALS OF OPERATING SYSTEM (DSE-2 / GE-4)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Fundamentals of Operating System	4	2L	0T	2P	-

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

Unit 1: Introduction to Operating Systems

Overview of operating systems and their evolution, Functions and components of an operating system, Operating system structures: monolithic, layered, microkernel, Process management and scheduling

Unit 2: Memory and Storage Management

Basics of memory management: paging, segmentation, Virtual memory: concepts, demand paging, page replacement algorithms, File systems and storage management: structure, operations, access methods, I/O systems and management

Unit 3: Concurrency and Synchronization

Principles of concurrency: processes, threads, CPU scheduling, Synchronization mechanisms: semaphores, mutexes, condition variables, Deadlock: prevention, avoidance, detection, and recovery, Case studies: analyzing concurrency in real operating systems

Unit 4: Security and Protection

Operating system security fundamentals: threats, vulnerabilities, controls, User authentication and access control, Security policies and mechanisms in operating systems, Case studies of security and protection in modern operating systems

Suggested Readings

1. "Operating System Concepts" by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
2. "Modern Operating Systems" by Andrew S. Tanenbaum and Herbert Bos
3. "Windows Internals" by Mark Russinovich, David A. Solomon, and Alex Ionescu for practical insights into Windows OS

Practical Component

1. Simulating process scheduling algorithms to understand their efficiency and fairness.
2. Implementing simple memory management techniques (e.g., paging or segmentation).
3. Developing a small program to demonstrate the use of synchronization mechanisms.
4. Conducting a security audit of an operating system configuration.

List of Experiments

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

FUNDAMENTALS OF DATA ANALYTICS (DSE-2 / GE-4)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Fundamentals of 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. Demonstrate an understanding of the data analytics lifecycle, including data collection, cleaning, analysis, and interpretation, and recognize the impact of data analytics through case studies.
2. Apply descriptive and inferential statistical techniques to analyze data, perform exploratory data analysis (EDA), and create meaningful data visualizations to effectively communicate insights.
3. Develop and implement basic predictive models using regression analysis (linear and logistic), apply classification techniques, conduct time series analysis, and validate and select appropriate models for given datasets.
4. Perform hands-on data analysis tasks, including exploratory data analysis, implementing regression models, creating visualizations, and conducting time series analysis, thereby gaining practical experience with real-world data.

Course Objectives

1. Provide a comprehensive introduction to data analytics, emphasizing its significance, various data types, and the data analytics lifecycle from data collection to interpretation.
2. Equip students with the knowledge of descriptive and inferential statistics, exploratory data analysis (EDA), and data visualization techniques essential for analyzing and interpreting data.
3. Introduce students to predictive analytics methods, including regression analysis, classification techniques, time series analysis, and model validation, enhancing their ability to predict and forecast based on data.
4. Familiarize students with popular data analytics tools such as R and Python, explore the applications of data analytics in business decision-making, discuss ethical considerations, and highlight emerging trends in the field.

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

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

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.

List of Experiments

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

OBJECT ORIENTED SOFTWARE ENGINEERING (DSE-2 / GE-4)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Object Oriented 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. Demonstrate a solid understanding of object-oriented programming principles such as encapsulation, inheritance, and polymorphism, and utilize UML for object-oriented analysis and design.
2. Gather and analyze software requirements using use case diagrams, activity diagrams, class diagrams, sequence diagrams, and state diagrams, and apply analysis patterns to address common design problems.
3. Implement object-oriented design principles (SOLID, DRY, KISS) and advanced design patterns to develop robust software architectures, perform refactoring, and design systems for scalability, maintainability, and security.
4. Develop software applications using object-oriented programming languages, conduct unit and integration testing, and implement continuous integration/delivery pipelines, ensuring performance optimization and reliability of the systems.

Course Objectives

1. Provide a comprehensive understanding of object-oriented programming concepts, principles of software engineering, and agile methodologies as they apply to object-oriented software development.
2. Equip students with skills in object-oriented analysis, including requirements gathering, use of UML diagrams, and applying analysis patterns to model and solve common problems in software design.
3. Enable students to apply object-oriented design principles and patterns, understand architectural frameworks, and perform refactoring techniques to ensure scalable, maintainable, and secure software systems.
4. Train students in implementing object-oriented designs using programming languages, conducting unit and integration testing, and optimizing performance in object-oriented systems, along with understanding deployment and CI/CD practices.

Unit 1: Introduction to Object-Oriented Concepts

Principles of object-oriented programming: encapsulation, inheritance, polymorphism, Object-oriented analysis and design using UML, Design patterns: creational, structural, behavioral, Principles of software engineering and their application in object-oriented development, Methodologies, object oriented software estimation, Agile methodologies in object-oriented software development.

Unit 2: Object-Oriented Analysis

Gathering requirements: use case diagrams, activity diagrams, Analysis models: class diagrams, sequence diagrams, state diagrams, Identifying classes and objects, modeling relationships and behavior, Applying analysis patterns to common problems in software design, Object oriented analysis using methods of Rumbaugh

Unit 3: Object-Oriented Design

Design principles: SOLID, DRY, KISS, Advanced design patterns and their applications, Object-oriented architecture and frameworks, Refactoring techniques: improving and maintaining the design of existing code, Designing for scalability, maintainability, and security.

Unit 4: Object-Oriented Implementation and Testing

Implementing object-oriented designs in programming languages (e.g., Java, C#, Python), Unit testing and test-driven development in an object-oriented context, Path Testing, State based Testing, Class Testing, Integration testing and system testing strategies, Deployment strategies and continuous integration/delivery (CI/CD) pipelines, Performance optimization in object-oriented systems.

Suggested Readings

1. “Object-oriented software engineering” by Yogesh Singh and Ruchika Malhotra. PHI Learning Pvt. Ltd., 2012.
2. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
3. "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development" by Craig Larman
4. "Refactoring: Improving the Design of Existing Code" by Martin Fowler

Practical Component

1. Designing and implementing a software application using object-oriented principles.
2. Applying design patterns to a software design problem and implementing the solution.
3. Conducting unit and integration testing on an object-oriented software project.
4. Participating in a group project that follows an agile development process.

List of Experiments

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

CRYPTOGRAPHY ESSENTIALS (DSE-2 / GE-4)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
Cryptography 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. Demonstrate a comprehensive understanding of the history and principles of cryptography, including symmetric (DES, AES) and asymmetric (RSA, ECC) key cryptography, and cryptographic hash functions (SHA, MD5).
2. Implement key exchange algorithms such as Diffie-Hellman, utilize digital signatures and certificates, understand the workings of Public Key Infrastructure (PKI), and apply cryptographic security protocols like SSL/TLS and PGP.
3. Understand and perform basic cryptanalysis techniques, identify and mitigate side-channel attacks, apply cryptographic techniques to secure data at rest, in transit, and in use, and navigate the legal and ethical landscape of cryptography.

Course Objectives

1. Introduce students to the history, principles, and essential concepts of cryptography, including both symmetric and asymmetric key cryptography and cryptographic hash functions.
2. Equip students with knowledge of key exchange algorithms, digital signatures, certificates, Public Key Infrastructure (PKI), and various cryptographic security protocols.
3. Familiarize students with the basics of cryptanalysis, side-channel attacks, and practical applications of cryptography in securing data, along with understanding the legal and ethical issues involved.

Unit 1: Foundations of Cryptography

History and principles of cryptography, Symmetric key cryptography: DES, AES, Asymmetric key cryptography: RSA, ECC, Cryptographic hash functions: SHA, MD5

Unit 2: Cryptographic Protocols and Practices

Key exchange algorithms: Diffie-Hellman, Digital signatures and certificates, Public Key Infrastructure (PKI) and its applications, Cryptographic security protocols: SSL/TLS, PGP

Unit 3: Cryptanalysis and Security

Basics of cryptanalysis: types and techniques, Side-channel attacks and countermeasures, Cryptography in practice: securing data at rest, in transit, and in use, Legal and ethical issues in cryptography

Unit 4: Advanced Cryptography and Emerging Trends

Introduction to quantum cryptography, Homomorphic encryption and its applications, Blockchain and cryptocurrency: cryptographic foundations, Future trends in cryptography

Suggested Readings

1. "Cryptography and Network Security: Principles and Practice" by William Stallings
2. "Understanding Cryptography: A Textbook for Students and Practitioners" by Christof Paar and Jan Pelzl
3. "Applied Cryptography: Protocols, Algorithms, and Source Code in C" by Bruce Schneier

Practical Component

1. Implementing basic encryption and decryption using symmetric and asymmetric algorithms.
2. Setting up a simple PKI environment and issuing digital certificates.
3. Conducting a simple cryptanalysis exercise on weak cryptographic algorithms.
4. Exploring the use of cryptographic libraries in software development.

List of Experiments

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

3D GRAPHICS AND RENDERING (DSE-2 / GE-4)

CREDIT DISTRIBUTION AND PRE-REQUISITES OF THE COURSE

Course title	Credits	Credit distribution of the course			Prerequisite of the course (if any)
		Lecture	Tutorial	Practical	
3D Graphics and Rendering	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. Utilize complex 3D modeling techniques, texture mapping, and material design to create detailed and optimized 3D models, applying lighting and shading models effectively.
2. Develop and optimize real-time rendering algorithms, use shader programming and GPU resources efficiently, apply real-time shadow and light techniques, post-processing effects, and ensure performance optimization in rendering.
3. Perform keyframe animation, motion capture, rigging, skeletal animation, facial animation, and lip-syncing, and apply physics-based and procedural animation techniques to bring 3D models to life.

Course Objectives

1. Provide students with comprehensive knowledge and techniques in 3D modeling, texture mapping, material design, lighting, and shading models.
2. Equip students with the skills to implement and optimize real-time rendering algorithms, shader programming, and GPU utilization.
3. Familiarize students with keyframe animation, motion capture, rigging, skeletal animation, facial animation, lip-syncing, physics-based animation, and procedural animation techniques.

Unit 1: Advanced 3D Modeling

Complex 3D Modeling Techniques, Texture Mapping and Material Design, Lighting and Shading Models, Particle Systems and Effects, Optimization Techniques for 3D Models.

Unit 2: Real-Time Rendering

Rendering Algorithms, Shader Programming and GPU Utilization, Real-Time Shadow and Light Techniques, Post-Processing Effects, Performance Optimization in Rendering

Unit 3: Animation and Rigging

Keyframe Animation and Motion Capture, Rigging and Skeletal Animation, Facial Animation and Lip Syncing, Physics-Based Animation, Procedural Animation Techniques

Unit 4: Advanced Graphics Project

Conceptualizing a Complex 3D Scene, Integrating Advanced Graphics Techniques, Implementing User Interactions, Performance Analysis and Optimization, Project Presentation and Critique.

Suggested Readings

1. "Real-Time Rendering, Fourth Edition" by Tomas Akenine-Möller, Eric Haines, and Naty Hoffman
2. "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
3. "3D Math Primer for Graphics and Game Development" by Fletcher Dunn and Ian Parberry

Practical Component

1. 3D Modeling and Animation Workshops
2. Real-Time Rendering Projects
3. Graphics Programming Labs
4. Portfolio Development

List of Experiments

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