# COURSE STRUCTURE AND SYLLABUS

**For**

## Bachelor of Computer Applications (BCA)

*(Applicable for batches admitted from 2024-2025)*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GURAJADA VIZIANAGARAM**
**VIZIANAGARAM-535003, Andhra Pradesh, India**

Verified Token: JNTUGV-085CF0586C643F51

# COURSE STRUCTURE

| I Year – I SEMESTER | | | | | | |
|---|---|---|---|---|---|---|
| **S. No** | **Category** | **Courses** | **L** | **T** | **P** | **Credits** |
| 1 | AEC | Communicative English | 3 | 0 | 0 | 3 |
| 2 | VAC | Computer Fundamentals | 3 | 0 | 0 | 3 |
| 3 | CORE | Digital Logic Design | 3 | 0 | 0 | 3 |
| 4 | CORE | Programming in C | 3 | 0 | 0 | 3 |
| 5 | MDE | Numerical and Statistical Methods | 3 | 0 | 0 | 3 |
| 6 | AEC | Communication Skills Lab | 0 | 0 | 2 | 1 |
| 7 | VAC | Office Automation Tools Lab | 0 | 0 | 2 | 1 |
| 8 | CORE | Programming in C Lab | 0 | 0 | 3 | 1.5 |
| 9 | MDE | Statistics using R Lab | 0 | 0 | 3 | 1.5 |
| **Total Credits** | | | **15** | **0** | **12** | **20** |

| I Year – II SEMESTER | | | | | | |
|---|---|---|---|---|---|---|
| **S. No** | **Category** | **Courses** | **L** | **T** | **P** | **Credits** |
| 1 | MDE | Discrete Mathematics | 3 | 0 | 0 | 3 |
| 2 | VAC | Environmental Science | 2 | 0 | 0 | 2 |
| 3 | MDE | Business Organization | 2 | 0 | 0 | 2 |
| 4 | CORE | Computer Organization | 3 | 0 | 0 | 3 |
| 5 | CORE | Data Structures | 3 | 0 | 0 | 3 |
| 6 | CORE | OOPs through C++ | 3 | 0 | 0 | 3 |
| 7 | CORE | Data Structures through C Lab | 0 | 0 | 3 | 1.5 |
| 8 | CORE | OOPs through C++ Lab | 0 | 0 | 3 | 1.5 |
| 9 | VAC | Indian Constitution | 2 | 0 | 0 | 2 |
| **Total Credits** | | | **19** | **0** | **6** | **21** |

| II Year – I SEMESTER | | | | | | |
|---|---|---|---|---|---|---|
| **S. No** | **Category** | **Courses** | **L** | **T** | **P** | **Credits** |
| 1 | CORE | Operating Systems | 3 | 0 | 0 | 3 |
| 2 | SEC | Python Programming | 3 | 0 | 0 | 3 |
| 3 | CORE | Database Management Systems | 3 | 0 | 0 | 3 |
| 4 | CORE | Artificial Intelligence | 3 | 0 | 0 | 3 |
| 5 | CORE | Operating Systems Lab | 0 | 0 | 3 | 1.5 |
| 6 | SEC | Python Programming Lab | 0 | 0 | 2 | 1 |
| 7 | CORE | Database Management Systems Lab | 0 | 0 | 3 | 1.5 |
| 8 | SEC | Animation Design | 0 | 0 | 3 | 1.5 |
| 9 | SEC | Design Thinking and Innovation | 0 | 1 | 2 | 2 |
| 10 | MDE | Indian Knowledge System | 2 | 0 | 0 | 2 |
| | | **Total Credits** | **12** | **1** | **14** | **21.5** |

| II Year – II SEMESTER | | | | | | |
|---|---|---|---|---|---|---|
| **S. No** | **Category** | **Courses** | **L** | **T** | **P** | **Credits** |
| 1 | CORE | Computer Networks | 3 | 0 | 0 | 3 |
| 2 | SEC | JAVA Programming | 3 | 0 | 0 | 3 |
| 3 | SEC | Web Technologies | 3 | 0 | 0 | 3 |
| 4 | CORE | Software Engineering | 3 | 0 | 0 | 3 |
| 5 | CORE | Computer Networks Lab | 0 | 0 | 3 | 1.5 |
| 6 | SEC | JAVA Programming Lab | 0 | 0 | 3 | 1.5 |
| 7 | SEC | Web Technologies Lab | 0 | 0 | 3 | 1.5 |
| 8 | SEC | Android APP Development | 0 | 0 | 3 | 1.5 |
| 9 | AEC | English for Technical Writing | 2 | 0 | 0 | 2 |
| | | **Total Credits** | **12** | **0** | **12** | **20** |
| Summer Internship during summer vacation and will be evaluated in the V Semester | | | | | | |

| I Year – I Sem | OPERATING SYSTEMS | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives:

The main objectives of the course are to make students

- Understand the fundamental concepts, functions and services provided by operating systems.
- Understand processes, threads, inter-process communication and CPU scheduling algorithms.
- Understand process synchronization mechanisms and different techniques for handling deadlocks.
- Analyze memory management techniques, virtual memory and page replacement algorithms.
- Understand file systems, file allocation methods, free-space management and disk scheduling techniques.

## Course Outcomes:

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

1. Explain the fundamental concepts, functions, services and structures of operating systems and system calls.
2. Analyze processes, threads, inter-process communication and CPU scheduling algorithms.
3. Explain process synchronization mechanisms and analyze different methods for deadlock prevention, avoidance, detection and recovery.
4. Analyze contiguous memory allocation, paging, virtual memory and page replacement techniques.
5. Explain file-system organization, file allocation, free-space management and mass-storage/disk scheduling techniques.

**Syllabus:**

**UNIT – I: Operating System Fundamentals**

**Operating Systems Overview:** Introduction to Operating Systems, Objectives and Functions of Operating Systems, Operating System Operations, Types of Operating Systems, Computing Environments, Open-Source Operating Systems.

**System Structures:** Operating System Services, User and Operating-System Interface, System Calls, Types of System Calls, System Programs, Operating-System Design and Implementation, Operating-System Structures.

**UNIT – II: Processes, Threads and CPU Scheduling**

**Processes:** Process Concept, Process States, Process Control Block, Process Scheduling, Operations on Processes, Inter-Process Communication.

**Threads:** Thread Concept, Benefits of Multithreading, Multithreading Models, Thread Libraries, Threading Issues.

**CPU Scheduling:** Basic Concepts, Scheduling Criteria, Scheduling Algorithms – First-Come, First-Served (FCFS), Shortest-Job-First (SJF), Priority Scheduling, Round Robin and Multilevel Queue Scheduling.

**UNIT – III: Process Synchronization and Deadlocks**

**Process Synchronization:** The Critical-Section Problem, Peterson's Solution, Mutex Locks, Semaphores, Monitors, Classical Problems of Synchronization – Bounded-Buffer Problem, Readers-Writers Problem and Dining-Philosophers Problem.

**Deadlocks:** System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Banker's Algorithm, Deadlock Detection and Recovery from Deadlock.

**UNIT – IV: Memory Management**

**Memory-Management Strategies:** Introduction, Swapping, Contiguous Memory Allocation, Paging, Page Tables, Address Translation.

**Virtual Memory Management:** Introduction, Demand Paging, Page Replacement, Page Replacement Algorithms – FIFO, Optimal and LRU, Allocation of Frames, Thrashing.

**UNIT – V: File Systems and Mass-Storage Management**

**File System:** File Concept, File Attributes, File Operations, Access Methods, Directory Structure, File-System Mounting and File Sharing.

**File-System Implementation:** File-System Structure, File-System Operations, Directory Implementation, File Allocation Methods – Contiguous, Linked and Indexed Allocation, Free-Space Management.

**Mass-Storage Structure:** Overview of Mass-Storage Structure, Disk Structure, Disk Scheduling, Disk Scheduling Algorithms – FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK, Swap-Space Management.

**Text Books:**

1. Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Operating System Concepts, 10th Edition, Wiley, 2018.
2. Andrew S. Tanenbaum and Herbert Bos, Modern Operating Systems, 4th Edition, Pearson, 2015.

**Reference Books:**

1. William Stallings, Operating Systems: Internals and Design Principles, 9th Edition, Pearson, 2018.
2. D. M. Dhamdhere, Operating Systems: A Concept-Based Approach, 3rd Edition, McGraw-Hill Education, 2012.
3. Gary Nutt, Operating Systems: A Modern Perspective, 3rd Edition, Pearson, 2004.

**Online Learning Resources:**

1. NPTEL – Operating Systems
2. Operating Systems – Peter India
3. Operating System Concepts – Wiley
4. GeeksforGeeks – Operating Systems

| II Year – I Sem | PYTHON PROGRAMMING | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives:

The main objectives of the course are to make students

- Understand the fundamental concepts, features and applications of Python programming.
- Develop programs using Python data types, operators, expressions and control structures.
- Apply Python built-in data structures and comprehensions to solve computational problems.
- Develop modular and reusable programs using functions, recursion and modules.
- Handle errors and exceptions and use Python packages for solving real-world problems.

## Course Outcomes:

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

1. Explain the fundamental concepts, features and applications of Python and develop simple Python programs.
2. Apply Python data types, operators, expressions and control structures to solve computational problems.
3. Implement programs using Python data structures such as lists, tuples, sets and dictionaries and apply comprehensions for data processing.
4. Develop modular and reusable programs using functions, recursion, lambda functions and modules.
5. Create and use Python modules and packages, install packages using PIP, and handle errors and exceptions in Python programs.

**UNIT – I:**

**Introduction:** History and evolution of Python, Features of Python, Applications of Python, Python Interpreter, Using the REPL (Shell), Running Python Scripts, Structure of a Python Program, Tokens, Identifiers and Keywords, Variables, Assignment Statements, Input and Output, Comments, Indentation, Type Conversion and Type Checking, Basic Problem Solving using Python.

**UNIT – II:**

**Types, Operators and Expressions: Data Types:** Integers, Floating-Point Numbers, Complex Numbers, Strings, Booleans. **Operators:** Arithmetic Operators, Comparison (Relational) Operators, Assignment Operators, Logical Operators, Bitwise Operators, Membership Operators, Identity Operators. Expressions and Order of Evaluation, Operator Precedence and Associativity. **Control Flow:** if, if-else, if-elif-else, Nested if, for, while, range(), Nested Loops, break, continue, pass.

**UNIT – III:**

**Data Structures: Lists** – Creating and Accessing Lists, Indexing, Slicing, List Operations, List Methods, Nested Lists; **Tuples** – Creating and Accessing Tuples, Tuple Operations, Tuple Methods, Tuple Unpacking; **Sets** – Creating Sets, Set Operations, Set Methods, Union, Intersection, Difference and Symmetric Difference**; Dictionaries** – Creating and Accessing Dictionaries, Adding, Updating and Deleting Elements, Dictionary Methods, Nested Dictionaries; **Sequences** – Sequence Operations, Indexing and Slicing**; Comprehensions** – List Comprehensions, Set Comprehensions and Dictionary Comprehensions.

**UNIT – IV:**

**Functions and Modular Programming:** Defining Functions, Calling Functions, Passing Arguments, Positional Arguments, Keyword Arguments, Default Arguments, Variable-Length Arguments (*args, **kwargs), Fruitful and Non-Fruitful Functions, Functions Returning Values, Anonymous Functions (Lambda Functions), Recursive Functions; Scope of Variables in a Function – Local and Global Variables, global Keyword, LEGB Rule; Introduction to Functional Programming – map(), filter() and reduce(); Modular Programming – Need for Modules, Creating Modules, Using Functions from Modules.

**UNIT – V:**

**Modules, Packages, PIP, Errors and Exceptions:**

**Modules** – Creating Modules, import Statement, from...import Statement, Aliasing, Namespace, Built-in Modules, User-Defined Modules.

**Python Packages** – Creating and Using Packages, Introduction to PIP, Installing and Managing Packages using PIP, Using Python Packages.

**Errors and Exceptions** – Syntax Errors, Runtime Errors, Logical Errors, Difference between an Error and Exception, Exception Handling, try-except Block, Multiple Exceptions, try-except-else, try-except-finally, Raising Exceptions using raise, User-Defined Exceptions.

**Text Books:**

1. Wesley J. Chun, Core Python Programming, 3rd Edition, Pearson, 2016.
2. Kenneth A. Lambert, Fundamentals of Python: First Programs, 3rd Edition, Cengage, 2024.

**Reference Books:**

1. Vamsi Kurama, Python Programming: A Modern Approach, Pearson India, 2017.
2. Allen B. Downey, Think Python, 2nd Edition, O'Reilly Media, 2015.
3. Gowrishankar S. and Veena A., Introduction to Python Programming, 1st Edition, CRC Press, 2018.
4. Paul Barry, Head First Python, 3rd Edition, O'Reilly Media, 2023.

**Web-Resources:**

1. NPTEL – Programming, Data Structures and Algorithms using Python
2. Infosys Springboard – Python Programming
3. Python 3 Documentation

| II Year – I Sem | DATABASE MANAGEMENT SYSTEMS | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives:

The main objectives of the course are to make students

- Understand the fundamental concepts, characteristics, architecture and applications of database management systems.
- Understand the relational model, relational algebra and entity-relationship modeling for database design.
- Develop skills in SQL for creating, manipulating and querying relational databases.
- Apply normalization techniques to design well-structured and efficient relational database schemas.
- Understand transaction management, concurrency control, database recovery and indexing techniques.

## Course Outcomes:

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

1. Explain the fundamental concepts, characteristics, data models and architecture of database management systems.
2. Design conceptual database models using ER diagrams and transform them into relational schemas with appropriate keys and integrity constraints.
3. Develop SQL queries to create, manipulate and retrieve data from relational databases using joins, subqueries, functions, grouping and views.
4. Apply functional dependencies and normalization techniques to refine relational database schemas and eliminate data redundancy.
5. Explain transaction management, concurrency control, recovery mechanisms and indexing techniques used in database systems.

## Syllabus:

**UNIT – I: Introduction to Database Systems and ER Model**

**Introduction:** Database System, Characteristics of Database Systems, Database Systems versus File Systems, Database Users, Advantages of Database Systems, Database Applications. Introduction to Data Models – Relational, Hierarchical, Network and Object-Oriented Data

Models. Concepts of Schema and Instance, Data Independence, Three-Schema Architecture, Database System Structure and Environment, Centralized and Client-Server Database Architecture

**Entity-Relationship Model:** Introduction to ER Model, Entities, Attributes, Entity Sets, Relationships, Relationship Sets, Constraints, Keys, Weak Entity Sets, ER Diagrams, Generalization, Specialization, Superclass, Subclass and Inheritance.

**UNIT – II: Relational Model and Relational Algebra**

**Relational Model:** Introduction to Relational Model, Domains, Attributes, Tuples and Relations, Relational Schema, Keys – Super Key, Candidate Key and Primary Key; **Integrity Constraints –** Domain Constraints, Key Constraints and Referential Integrity Constraints, Null Values.

**Relational Algebra:** Selection, Projection, Rename, Set Operations, Cartesian Product, Joins and Division, Examples of Relational Algebra Queries.

**Relational Calculus:** Introduction to Tuple Relational Calculus and Domain Relational Calculus with simple examples.

**UNIT – III: SQL**

**Basic SQL:** Introduction to SQL, SQL Data Types, Creating and Modifying Tables using CREATE and ALTER, Data Manipulation using INSERT, UPDATE and DELETE.

**SQL Querying:** SELECT statement, WHERE clause, Arithmetic and Logical Operators, Comparison Operators, SQL Functions – Numeric, String and Date/Time Functions; Creating Tables with Relationships, Primary Key, Foreign Key and Integrity Constraints. Nested Queries and Subqueries, Aggregate Functions, GROUP BY, HAVING, ORDER BY. Joins – Inner Join, Outer Join and Self Join. **Set Operations** – UNION, INTERSECT and EXCEPT. **Views** – Creating and Using Views, Updatable and Non-Updatable Views.

**UNIT – IV: Database Design and Normalization**

**Schema Refinement and Normalization:** Need for Schema Refinement, Data Redundancy and Anomalies. Functional Dependencies, Functional Dependency Theory. Normal Forms-First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Boyce-Codd Normal Form (BCNF), Lossless-Join Decomposition, Dependency-Preserving Decomposition; Introduction to Multi-valued Dependencies, Fourth Normal Form (4NF) and Fifth Normal Form (5NF), Concept of Surrogate Keys.

**UNIT – V: Transaction Management and Indexing**

**Transaction Management:** Transaction Concept, Transaction States, ACID Properties, Concurrent Executions, Serializability and Recoverability.

**Concurrency Control** – Lock-Based Protocols, Two-Phase Locking, Timestamp-Based Protocols and Optimistic Concurrency Control, Deadlocks in Database Systems.

**Database Recovery:** Failure Classification, Storage and Recovery, Atomicity, Log-Based Recovery and Basic Recovery Algorithms.

**Indexing Techniques:** Introduction to Indexing, Primary and Secondary Indexes, ISAM, B+ Trees and Operations on B+ Trees, Hash-Based Indexing.

**Text Books:**

1. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill Education, 2003.
2. Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, 7th Edition, McGraw-Hill Education, 2020.

**Reference Books:**

1. C. J. Date, An Introduction to Database Systems, 8th Edition, Pearson, 2003.
2. Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson, 2016.
3. Carlos Coronel and Steven Morris, Database Systems: Design, Implementation, & Management, 13th Edition, Cengage Learning, 2018.
4. Thomas Connolly and Carolyn Begg, Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Pearson, 2014.

**Web Resources:**

1. NPTEL – Introduction to Database Systems
2. Infosys Springboard – Database Management Systems
3. MySQL Documentation
4. PostgreSQL Documentation
5. Oracle Database Documentation

| II Year I Sem | ARTIFICIAL INTELLIGENCE | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives:

The main objectives of the course are to make students

- Understand the fundamental concepts, foundations, history and applications of Artificial Intelligence.
- Understand intelligent agents, problem formulation and various search strategies used to solve AI problems.
- Understand different knowledge representation techniques and reasoning mechanisms used in Artificial Intelligence.
- Apply propositional and first-order logic, inference, unification and resolution for knowledge-based problem solving.
- Understand uncertainty handling, pl nning, game playing and other classical AI problem-solving techniques.

## Course Outcomes:

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

1. Explain the fundamental concepts, foundations, history, applications and intelligent-agent architectures of Artificial Intelligence.
2. Formulate AI problems and apply uninformed, informed and heuristic search techniques to find solutions.
3. Represent knowledge using logic, semantic networks, frames and rule-based systems and perform reasoning under uncertainty
4. Apply propositional and first-order logic, unification, forward and backward chaining, and resolution for AI problem solving.
5. Apply game-playing, planning and decision-making techniques to solve classical Artificial Intelligence problems.

**Syllabus:**

**UNIT – I: Introduction to Artificial Intelligence and Intelligent Agents**

**Introduction:** Artificial Intelligence – Definition, Foundations of AI, History of AI, AI Problems,

Characteristics of AI Problems, AI Techniques, Applications of AI, Advantages and Limitations of AI.

**Intelligent Agents:** Agents and Environments, Rationality, Nature of Environments, Types of Agents, Structure of Agents, Simple Reflex Agents, Model-Based Agents, Goal-Based Agents, Utility-Based Agents and Learning Agents.

**Problem Solving:** Problem-Solving Agents, Problem Formulation, State-Space Representation, Initial State, Goal State, Operators/Actions and Path Cost.

**UNIT – II: Search Techniques**

**Uninformed Search:** Searching for Solutions, State-Space Search, Breadth-First Search, Depth-First Search, Uniform-Cost Search, Depth-Limited Search and Iterative Deepening Search.

**Informed and Heuristic Search:** Heuristic Functions, Greedy Best-First Search, Hill Climbing, A* Search, Problem Reduction, AO* Algorithm.

**Search Performance:** Completeness, Optimality, Time Complexity and Space Complexity of Search Algorithms.

**UNIT – III: Game Playing and Knowledge Representation**

**Game Playing:** Introduction to Game Playing, Adversarial Search, Games and Game Trees, Minimax Algorithm, Optimal Decisions, Evaluation Functions, Alpha-Beta Pruning, Multiplayer Games and Problems in Game Playing.

**Knowledge Representation:** Knowledge Representation Issues, Knowledge Representation Approaches, Propositional Logic, Predicate Logic, Semantic Networks, Frames and Inheritance, Rule-Based Knowledge Representation and Production Systems.

**UNIT – IV: Logic, Inference and Reasoning**

**Mathematical Logic:** Propositional Logic, First-Order Logic, Syntax and Semantics, Propositional versus First-Order Inference.

**Inference Techniques:** Inference Rules, Substitution, Unification, Forward Chaining, Backward Chaining and Resolution.

**Reasoning:** Rule-Based Deduction Systems, Reasoning with Incomplete Knowledge, Non-Monotonic Reasoning – Introduction.

**Reasoning under Uncertainty:** Uncertainty in AI, Probability Basics, Conditional Probability, Bayes' Theorem and Bayesian Reasoning.

**UNIT – V: Planning, Decision Making and Classical AI Applications**

**Planning:** Introduction to Planning, Planning Problems, Components of Planning, State-Space Planning, Goal Stack Planning, Partial-Order Planning and Planning Graphs – Introduction.

**Decision Making:** Decision Making in AI, Utility-Based Decision Making, Decision Trees – Conceptual Introduction, Decision Making under Uncertainty.

**Classical AI Problem Solving:** Constraint Satisfaction Problems, Constraint Satisfaction Techniques, Backtracking Search, Constraint Propagation.

**Classical AI Applications:** Expert Systems – Architecture and Components, Knowledge Base and Inference Engine, Natural Language Processing – Introduction, Robotics – Introduction, Computer Vision – Introduction and AI-based Recommendation/Decision Support Systems.

**Text Books:**

1. Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson, 2021.
2. Elaine Rich, Kevin Knight and Shivashankar B. Nair, Artificial Intelligence, 3rd Edition, McGraw-Hill Education, 2017.

**Reference Books:**

1. David L. Poole and Alan K. Mackworth, Artificial Intelligence: Foundations of Computational Agents, 2nd Edition, Cambridge University Press, 2017.
2. George F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 6th Edition, Pearson, 2008.
3. Nils J. Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann, 1998.
4. Saroj Kaushik, Artificial Intelligence, Cengage Learning.

**Web Resources**

1. NPTEL – Artificial Intelligence
2. SWAYAM – Fundamentals of Artificial Intelligence
3. Poole & Mackworth – Artificial Intelligence: Foundations of Computational Agents
4. Google AI – AI concepts and applications

| II Year I Sem | OPERATING SYSTEMS LAB | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives:

The main objectives of the course are to make students

- Understand the basic UNIX/Linux operating system environment and execute shell commands and scripts.
- Implement process creation, process management and inter-process communication using C and system calls.
- Implement and analyze various CPU scheduling, process synchronization and deadlock handling algorithms.
- Implement memory allocation and page replacement algorithms and evaluate their performance.
- Implement different file allocation methods using C programming.

## Course Outcomes:

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

1. Demonstrate basic UNIX/Linux commands and shell scripts for performing common operating system tasks.
2. Implement process creation, process management and inter-process communication using C system calls.
3. Implement and analyze CPU scheduling algorithms such as FCFS, SJF, Priority and Round Robin using suitable performance measures.
4. Implement process synchronization, deadlock avoidance and detection, memory allocation and page replacement algorithms using C.
5. Implement different file allocation methods and analyze their operation using C programming.

## LIST OF EXPERIMENTS

**Exercise–1: UNIX/Linux Commands and Shell Programming**

a) Study and execute basic UNIX/Linux commands for file, directory and process management.

b) Write shell scripts to perform arithmetic operations, find the largest of given numbers, calculate the factorial of a number and check whether a number is prime.

**Exercise–2: Process Creation and System Calls**

a) Write a C program to create a child process using fork().
b) Write a C program to demonstrate the execution of parent and child processes.
c) Write a C program to demonstrate the use of wait() and exit() system calls.
d) Write a C program to demonstrate the use of the exec() family of system calls.

**Exercise–3: Inter-Process Communication**

a) Write a C program to demonstrate inter-process communication using **pipes**.
b) Write a C program to demonstrate communication between parent and child processes using pipes.

**Exercise–4: CPU Scheduling – FCFS and SJF**

a) Write a C program to implement First-Come, First-Served (FCFS) CPU scheduling.
b) Write a C program to implement Shortest Job First (SJF) CPU scheduling.

**Exercise–5: CPU Scheduling – Priority and Round Robin**

a) Write a C program to implement Priority Scheduling.
b) Write a C program to implement Round Robin Scheduling with a given time quantum.

**Exercise–6: Process Synchronization**

a) Write a C program to implement the Producer–Consumer Problem using semaphores.
b) Write a C program to implement the Readers–Writers Problem using appropriate synchronization mechanisms.
c) Write a C program to demonstrate the Dining Philosophers Problem.

**Exercise–7: Deadlock Avoidance**

a) Write a C program to implement the Banker's Algorithm.

**Exercise–8: Deadlock Detection**

a) Write a C program to implement the Deadlock Detection Algorithm and determine whether the given system is in a deadlock state.

**Exercise–9: Memory Allocation**

Write C programs to implement the following memory allocation strategies:

**a)** First Fit

**b)** Best Fit

**c)** Worst Fit

Calculate and compare the memory allocation and utilization for the given processes and memory blocks.

**Exercise–10: Page Replacement Algorithms**

a) Write a C program to implement the FIFO Page Replacement Algorithm.

b) Write C programs to implement Optimal Page Replacement Algorithm.

c) Write C programs to implement **Least** Recently Used (LRU) Page Replacement Algorithm.

**Exercise–11: File Allocation Methods**

Write C programs to simulate the following file allocation methods:

**a)** Contiguous File Allocation

**b)** Linked File Allocation

**c)** Indexed File Allocation

Display the blocks allocated to each file.

## Text Books:

1. Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Operating System Concepts, 10th Edition, Wiley, 2018.
2. Andrew S. Tanenbaum and Herbert Bos, Modern Operating Systems, 4th Edition, Pearson, 2015.

## Reference Books:

1. William Stallings, Operating Systems: Internals and Design Principles, 9th Edition, Pearson, 2018.
2. D. M. Dhamdhere, Operating Systems: A Concept-Based Approach, 3rd Edition, McGraw-Hill Education, 2012.
3. R. G. Kruse, Bruce P. Leung and Clovis L. L. Tondo, Data Structures and Program Design in C, 2nd Edition, Pearson.
4. Yashavant Kanetkar, Let Us C, 19th Edition, BPB Publications, 2023.

## Web Resources:

1. NPTEL – Operating Systems
2. Linux man-pages – System Calls and Library Functions
3. GNU C Library Documentation
4. The Open Group – POSIX System Interfaces
5. GeeksforGeeks – Operating Systems

| II Year I Sem | PYTHON PROGRAMMING LAB | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives:

The main objectives of the course are to make students

- Understand and practice the basic syntax, data types and programming constructs of Python.
- Develop Python programs using operators, expressions and control structures.
- Implement problem-solving applications using Python data structures.
- Develop reusable programs using functions, recursion, modules and packages.
- Apply file processing and exception handling techniques in Python programming.

## Course Outcomes:

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

1. Develop and execute Python programs using basic syntax, data types, operators, expressions and control structures.
2. Implement problem-solving applications using strings, lists, tuples, sets, dictionaries and comprehensions.
3. Develop modular and reusable Python programs using functions, recursion, lambda functions and modules.
4. Perform file processing, data manipulation and basic computational tasks such as statistical and matrix operations using Python.
5. Develop robust Python programs using packages, PIP and exception-handling mechanisms and apply Python programming concepts to solve practical problems.

## LIST OF EXPERIMENTS

### Exercise–1: Python Basics, Data Types and Strings

a) Write a Python program to demonstrate different numeric data types and type conversion.

b) Write a Python program to perform different arithmetic, relational and logical operations on numbers.

c) Write a Python program to create, concatenate and display strings and access substrings using indexing and slicing.

d) Write a Python program to demonstrate commonly used string methods.

**Exercise–2: Control Structures**

a) Write a Python program to convert temperatures from Celsius to Fahrenheit and vice versa.

b) Write a Python program to determine whether a given number is prime.

c) Write a Python program to find the factorial of a given number.

d) Write a Python program to generate the Fibonacci series up to *n* terms.

**Exercise–3: Problem Solving using Loops**

a) Write a Python program to check whether a given number is a palindrome.

b) Write a Python program to check whether a given number is an Armstrong number.

c) Write a Python program to print all prime numbers less than 100.

d) Write a Python program to generate different number and character patterns using nested loops.

**Exercise–4: Lists and Tuples**

a) Write a Python program to perform various operations on lists such as insertion, deletion, searching, sorting and slicing.

b) Write a Python program to find the largest, smallest, sum and average of elements in a list.

c) Write a Python program to find all duplicate elements in a list using a function dups().

d) Write a Python program to find all unique elements in a list using a function unique().

e) Write a Python program to demonstrate tuple operations, slicing, packing and unpacking.

**Exercise–5: Sets and Dictionaries**

a) Write a Python program to perform union, intersection, difference and symmetric difference operations on sets.

b) Write a Python program to count the occurrences of each character in a string and store the results in a dictionary.

c) Write a Python program to count the frequency of words in a given sentence using a dictionary.

d) Write a Python program to create a student dictionary containing student details and perform insertion, updating, searching and deletion operations.

**Exercise–6: Comprehensions and String Processing**

a) Write Python programs using list comprehensions to generate and filter elements from a list.
b) Write a Python program using set and dictionary comprehensions.
c) Write a Python program to demonstrate the use of split() and join() methods.
d) Write a Python function nearly_equal() to test whether two strings are nearly equal, where one string can be obtained from the other by a single mutation.

**Exercise–7: Functions**

a) Write Python programs to demonstrate functions with positional, keyword and default arguments.
b) Write a Python program to demonstrate variable-length arguments using *args and **kwargs.
c) Write a Python program to define a function that returns multiple values.
d) Write a Python program to demonstrate local and global variables.

**Exercise–8: Lambda Functions and Functional Programming**

a) Write Python programs using lambda functions for simple mathematical operations.
b) Write a Python program using map() to perform an operation on all elements of a list.
c) Write a Python program using filter() to extract elements satisfying a given condition.
d) Write a Python program using reduce() to perform an aggregate operation on a list.

**Exercise–9: Recursion and Modules**

a) Write a Python program to find the factorial of a number using recursion.
b) Write a recursive Python function to generate Fibonacci numbers.
c) Write a Python program to define a module containing functions for generating Fibonacci numbers and import the module into another program.
d) Write a Python program to create and use a user-defined module containing mathematical functions.

**Exercise–10: Standard Modules and Packages**

a) Write Python programs using the standard math, random and datetime modules.
b) Write a Python program to display the current date and time in a specified format.
c) Create a simple user-defined Python package and demonstrate importing and using its modules.
d) Demonstrate installing and using a Python package using **PIP**.

**Exercise–11: Classes and Objects**

a) Write a Python class to implement pow(x, n).
b) Write a Python class to reverse a string word by word.
c) Write a Python class to convert an integer into its Roman numeral representation.
d) Write a Python program to demonstrate the creation of objects and accessing attributes and methods.

**Exercise–12: File Processing**

a) Write a Python program to copy the contents of one text file into another file.
b) Write a Python program to read a text file and print all unique words in alphabetical order.
c) Write a Python program to count the frequency of characters in a given file.
d) Write a Python program to count the number of lines, words and characters in a text file.

**Exercise–13: File Analysis and Data Processing**

a) Write a Python program to analyze character frequencies in a given file and identify whether it is likely to be a Python source file, C source file or general text file.
b) Write a Python program to search for a given word in a text file and display its occurrences.
c) Write a Python program to read student information from a file and generate a simple report.

**Exercise–14: Statistics and Matrix Operations**

a) Write a Python program to find the **mean, median and mode** of a given set of numbers stored in a list.
b) Write a Python program to define and display a matrix.
c) Write a Python program to perform addition of two square matrices.

d) Write a Python program to perform multiplication of two square matrices.

**Exercise–15: Exception Handling and Integrated Problem Solving**

a) Write a Python program to demonstrate exception handling using try-except.
b) Write a Python program to handle multiple exceptions using appropriate except blocks.
c) Write a Python program using try-except-else-finally.
d) Write a Python program to raise an exception using raise.
e) Define and use a user-defined exception for validating input.
f) Develop a simple **menu-driven application** integrating functions, lists/dictionaries and exception handling.

## Text Books:

1. Eric Matthes – Python Crash Course, 3rd Ed., 2023.
2. Paul Barry – Head First Python, 3rd Ed., 2023.

## Reference Books:

1. Mark Lutz – Learning Python, 6th Ed., 2025.
2. Allen B. Downey – Think Python, 2nd Ed., 2015.
3. William F. Punch & Richard Enbody – The Practice of Computing Using Python, 3rd Ed., 2016.

## Web-Resources:

1. Python Official Documentation – Python 3 — Official language tutorial, library reference, language reference and Python usage documentation.
2. Python Beginner's Guide — Beginner-oriented resources for learning Python and setting up the Python environment.
3. Python Documentation – Tutorial — Official tutorial covering Python syntax, data structures, functions, modules, errors and exceptions.
4. Python Package Index (PyPI) — Repository for discovering and installing Python packages.
5. NPTEL – Programming, Data Structures and Algorithms using Python — Supplementary lectures and programming material.

| II Year I Sem | DATABASE MANAGEMENT SYSTEMS LAB | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives:

This course will enable students to

- Develop and manage relational databases using SQL DDL, DML and DCL commands.
- Create tables with appropriate keys and integrity constraints and perform database operations.
- Formulate SQL queries using joins, nested queries, aggregate functions, set operations, views and built-in functions.
- Develop PL/SQL programs using control structures, exception handling, procedures, functions and cursors.
- Implement database triggers and transaction-control operations for developing reliable database applications.

## Course Outcomes:

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

1. Create and manipulate relational databases using SQL DDL, DML and DCL commands with appropriate integrity constraints.
2. Develop SQL queries using joins, nested queries, aggregate functions, set operations, built-in functions and views.
3. Develop PL/SQL programs using variables, control structures, loops and exception-handling mechanisms.
4. Develop and execute stored procedures, functions and cursors for implementing database processing tasks.
5. Implement database triggers and transaction-control operations to develop reliable and maintainable database applications.

### LIST OF EXPERIMENTS

**Exercise–1: Database and Table Creation**

a) Create a database/schema and tables using SQL DDL commands.

b) Demonstrate CREATE, ALTER, DROP and TRUNCATE commands.

c) Insert records into tables using INSERT.
d) Retrieve records using SELECT.
e) Modify and delete records using UPDATE and DELETE.
**f)** Create tables using appropriate Primary Key, Foreign Key, UNIQUE, NOT NULL and CHECK constraints**.**

**Exercise–2: Basic and Advanced SQL Queries**

Write SQL queries using:

g) Comparison and logical operators.
h) ORDER BY.
i) DISTINCT.
j) LIKE, BETWEEN, IN and IS NULL.
k) UNION, INTERSECT and EXCEPT/equivalent set operations.
l) ANY, ALL, EXISTS and NOT EXISTS.
m) Nested and subqueries.

**Exercise–3: Aggregate Functions, Grouping and Views**

a) Write queries using COUNT(), SUM(), AVG(), MAX() and MIN().
b) Use GROUP BY and HAVING.
c) Develop queries involving multiple tables and aggregate functions.
d) Create, use and drop database views.
e) Demonstrate simple updatable and non-updatable views.

**Exercise–4: SQL Built-in Functions**

Write SQL queries using:

a) Numeric functions.
b) String functions such as concatenation, LOWER(), UPPER(), LENGTH(), SUBSTR() and trimming functions.
c) Date and time functions.
d) Conversion functions such as TO_CHAR(), TO_NUMBER() and TO_DATE() where supported by the selected DBMS.

**Exercise–5: PL/SQL Basics and Transaction Control**

a) Write simple PL/SQL blocks containing declaration, executable and exception-handling sections.
b) Develop a PL/SQL program to retrieve student information and display the result based on specified conditions.
c) Demonstrate predefined and user-defined exception handling.
d) Demonstrate transaction-control operations using COMMIT, ROLLBACK and SAVEPOINT.

**Exercise–6: PL/SQL Control Structures and Loops**

Develop PL/SQL programs using:

a) IF, IF-ELSE and nested IF.
b) CASE statements and CASE expressions.
c) WHILE loops.
d) Numeric FOR loops.
e) Nested loops.
f) NULLIF and COALESCE functions.

**Exercise–7: PL/SQL Exception Handling**

Develop PL/SQL programs demonstrating:

a) Predefined exceptions.
b) User-defined exceptions.
c) RAISE statement.
d) RAISE_APPLICATION_ERROR.
e) Exception handling in database operations.

**Exercise–8: Stored Procedures**

a) Create and execute stored procedures.
b) Develop procedures with IN, OUT and IN OUT parameters.
c) Develop procedures to insert, update and retrieve database records.
d) Develop a procedure for a suitable application such as student, employee or banking database.

**Exercise–9: Stored Functions**

a) Create and execute stored functions.
b) Develop functions with parameters.
c) Invoke stored functions in SQL statements, wherever supported.
d) Develop functions to perform calculations and retrieve information from database tables.

**Exercise–10: Cursors**

Develop PL/SQL programs using:

a) Implicit and explicit cursors.
b) Parameterized cursors.
c) Cursor FOR loops.
d) FOR UPDATE cursors.
e) WHERE CURRENT OF clause.

**Exercise–11: Database Triggers**

Develop programs using:

a) BEFORE triggers.
b) AFTER triggers.
c) Row-level triggers.
d) Statement-level triggers.
e) Triggers for maintaining data integrity and auditing database operations.
f) INSTEAD OF triggers for views, where supported.

**Text Books:**

1. Nilesh Shah, Database Systems Using Oracle, PHI Learning, 2007.
2. Rick F. van der Lans, Introduction to SQL: Mastering the Basics, 4th Edition, Addison-Wesley/Pearson, 2007.
3. Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, 7th Edition, McGraw-Hill Education, 2020.

**Reference Books**

1. Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson, 2016.

2. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill, 2003.
3. Thomas Connolly and Carolyn Begg, Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Pearson, 2014.

**Web Resources**

1. Oracle Database Documentation
2. Oracle Live SQL
3. Oracle PL/SQL Language Reference
4. MySQL Documentation
5. NPTEL – Introduction to Database Systems
6. GeeksforGeeks – DBMS

| II Year I Sem | ANIMATION DESIGN | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives

The course aims to:

- Develop fundamental skills in 2D graphic design and animation.
- Introduce students to drawing, composition, colour, typography and visual design.
- Develop skills in creating 2D animations, character animations and motion graphics.
- Provide practical experience in scene creation, lighting, visual effects and audio-video synchronization.
- Enable students to design and produce short animated clips and promotional content using animation and multimedia tools.

## Course Outcomes

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

1. Create basic graphic designs such as logos, brochures, posters, visiting cards and digital backgrounds.
2. Develop 2D animations using layers, keyframes, motion, masking and transformation techniques.
3. Create basic character and object animations with appropriate movement and visual effects.
4. Design scenes with suitable backgrounds, lighting, rendering and special effects.
5. Produce short multimedia animations by integrating graphics, animation, audio, titles and video.

## List of Experiments

**Experiment 1: Graphic Design Fundamentals**

a) Design a logo for an organization or product.
b) Design a brochure/poster using appropriate layout, colour and typography.
c) Design a visiting card and cover page.
d) Create and edit images using selection, cropping, resizing and transformation techniques.
e) Remove/cut out an object or person from an image and place it on a suitable background.

**Experiment 2: Digital Background and Composition**

a) Create digital backgrounds using gradient, colour and texture techniques.
b) Combine multiple images to create a composite scene.
c) Apply basic image effects, filters and adjustments.
d) Design a web/social-media banner.
e) Create a simple animated logo or title.

**Experiment 3: 2D Animation**

a) Create a simple animation using frames and keyframes.
b) Demonstrate motion and shape transformation.
c) Create animation using layers and layer-based movement.
d) Create a simple masking animation.
e) Animate text and graphical objects.

**Experiment 4: Character Animation**

a) Design a simple cartoon character using basic shapes.
b) Create basic character movements such as walking, running or jumping.
c) Animate facial expressions and body movements.
d) Create a simple lip-sync animation.
e) Apply filters and special effects to enhance the character animation.

**Experiment 5: Object and Scene Animation**

a) Create a simple car/bike or other object model.
b) Animate the movement of the object through a scene.
c) Create a basic indoor or outdoor environment.
d) Animate environmental elements such as trees, clouds, water or moving objects.
e) Create a short animated scene combining characters and objects.

**Experiment 6: Lighting and Rendering**

a) Create an indoor or outdoor scene with suitable lighting.
b) Set lighting for morning, day and night scenes.
c) Apply basic lighting and shadow effects.
d) Render a still frame of the created scene.

e) Render the animation as a **video**.

**Experiment 7: Motion Graphics and Multimedia**

a) Create title graphics and animated text.
b) Create a 30-second animated commercial advertisement.
c) Synchronize audio with animation/video.
d) Add background music, voice or sound effects.
e) Create an animated introductory/outro sequence.

**Experiment 8: Special Effects and Mini Project**

a) Create an animation effect such as rain, snow, smoke, fire or water.
b) Create a simple explosion or impact effect.
c) Combine multiple effects in an animated scene.
d) Create a 30–60 second animated short clip incorporating graphics, animation, sound and effects.
e) Present the completed animation as a mini project.

**Software/Tools**

The laboratory may use any suitable current animation and multimedia software, such as:

- Canva
- Figma – free plan for one user
- Adobe Animate or equivalent– 2D animation
- Blender– 2D/3D animation, modelling, lighting and rendering
- Adobe After Effects or equivalent– motion graphics and visual effects
- Audacity or equivalent– basic audio editing
- Vercel AI tools for design and creative content generation

**Text Books**

1. Russell Chun, Adobe Animate Classroom in a Book 2024 Release, Adobe Press, 2024.
2. Conrad Chavez, Adobe Photoshop Classroom in a Book 2025 Release, Adobe Press, 2025.

**Reference Books**

1. Richard Williams, The Animator's Survival Kit, Revised Edition, Faber & Faber, 2009.

2. Tony White, Animation from Pencils to Pixels: Classical Techniques for the Digital Animator, Focal Press, 2006.
3. Chris Georgenes, How to Cheat in Adobe Animate CC, Focal Press, 2017.
4. The Illusion of Life: Disney Animation, Disney Editions, 1981.

**Web Resources**

1. Adobe Animate – Official Tutorials and Help
2. Adobe Photoshop – Official Tutorials and Help
3. Blender – Official Tutorials
4. Adobe After Effects – Official Tutorials

| II Year I Sem | DESIGN THINKING AND INNOVATION | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 2 |

## Course Objectives

- Develop creativity, innovative thinking and a basic designer mindset.
- Understand the basic principles and stages of Design Thinking.
- Identify user needs and define real-world problems.
- Apply simple ideation, prototyping and testing techniques.
- Develop and present a simple user-centered solution.

## Course Outcomes

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

1. Explain the basic concepts and process of Design Thinking and innovation.
2. Identify user needs and formulate simple problem statements.
3. Generate and select suitable ideas for solving identified problems.
4. Develop and test simple prototypes using user feedback.
5. Apply Design Thinking to a real-world problem and present the proposed solution.

### UNIT – I: INTRODUCTION TO DESIGN THINKING

**Theory**

- Design Thinking – Meaning and importance
- Creativity, innovation and problem solving
- Designer mindset
- Human-Centered Design
- Design Thinking process
- Empathize, Define, Ideate, Prototype and Test
- Examples of Design Thinking applications

**Tasks**

**Task 1: Everyone is a Designer**

- Identify problems in everyday life.
- Observe how existing products/services can be improved.

**Task 2: Wallet/Bag Challenge**

- Understand the Design Thinking process through a simple design challenge.

- Create a basic solution and prototype.

## UNIT – II: EMPATHIZE AND DEFINE

**Theory**

- Understanding users and stakeholders
- Observation and active listening
- User interviews
- User needs and pain points
- Empathy Map
- User Persona
- Problem statement
- How Might We questions

**Tasks**

**Task 3: Identify a Problem**

- Observe the college/community environment.
- Select one simple real-world problem.

**Task 4: Understand the User**

- Conduct a short interview/observation.
- Prepare an **Empathy Map** or simple **User Persona**.
- Prepare a problem statement.

## UNIT – III: IDEATION

**Theory**

- Creative thinking
- Divergent and convergent thinking
- Brainstorming
- Brainwriting
- Mind mapping
- SCAMPER – Introduction
- Selection and prioritization of ideas

**Tasks**

**Task 5: Brainstorming**

- Generate multiple solutions for the selected problem.
- Use brainstorming or mind mapping.

**Task 6: Select the Best Idea**

- Compare the ideas based on usefulness and feasibility.
- Select one idea for prototyping.

**UNIT – IV: PROTOTYPING AND TESTING**

**Theory**

- Meaning and purpose of prototypes
- Low-fidelity prototypes
- Paper prototypes
- Sketches and storyboards
- Wireframes
- Basic digital prototypes
- User testing and feedback
- Iteration

**Tasks**

**Task 7: Create a Prototype**

- Develop a simple prototype of the selected solution.
- Use paper, sketches, wireframes or a suitable digital tool.

**Task 8: Test and Improve**

- Show the prototype to users/peers.
- Collect feedback.
- Make necessary improvements.

**UNIT – V: INNOVATION AND PRESENTATION**

**Theory**

- Design Thinking and innovation

- Value of a solution
- Desirability and feasibility
- Sustainable and inclusive design – introduction
- Design storytelling
- Presentation and pitching
- Design Thinking case studies

**Tasks**

**Task 9: Case Study**

- Study one successful product/service developed using user-centered design.
- Identify the problem, users and solution.

**Task 10: Final Design Challenge**

Students work individually or in small teams to:

1. Identify a simple real-world problem.
2. Understand the user.
3. Define the problem.
4. Generate ideas.
5. Create a prototype.
6. Test and improve it.
7. Present the final solution.

## Text Books:

1. Tim Brown, Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation, Revised and Updated Edition, Harper Business, 2019.
2. Jeanne Liedtka, Andrew King and Kevin Bennett, Solving Problems with Design Thinking: Ten Stories of What Works, Columbia University Press, 2013.

## Reference Books:

1. Tom Kelley and Jonathan Littman, The Art of Innovation: Lessons in Creativity from IDEO, America's Leading Design Firm, Profile Books, 2002.
2. Jeanne Liedtka, Randy Salzman and Daisy Azer, Design Thinking for the Greater Good: Innovation in the Social Sector, Columbia Business School Publishing, 2017.

3. Michael Lewrick, Patrick Link and Larry Leifer, The Design Thinking Toolbox, Wiley, 2020.

## Web Resources:

Other Useful Design Thinking Frameworks and Methodologies:

1. Human-Centered Design Toolkit – IDEO Design Kit
2. Design Thinking Resources – Stanford d.school
3. Design Thinking for Educators – IDEO
4. Interaction Design Foundation – User-Centered Design Resources

| II Year I Sem | INDIAN KNOWLEDGE SYSTEM | L | T | P | C |
|---|---|---|---|---|---|
| | | 2 | 0 | 0 | 2 |

## Course Objectives

The course aims to:

- Introduce students to the Indian Knowledge System (IKS), its sources, traditions and evolution.
- Familiarize students with India's contributions to mathematics, astronomy, science, logic and linguistics.
- Understand selected Indian approaches to knowledge, reasoning and problem solving.
- Explore Indian contributions to technology, medicine, architecture, agriculture and sustainable practices.
- Relate selected IKS concepts to computing, computational thinking, artificial intelligence and digital technologies.

## Course Outcomes

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

1. Explain the concept, sources, scope and major domains of the Indian Knowledge System.
2. Describe major contributions of Indian scholars to mathematics, astronomy and scientific traditions.
3. Explain basic concepts of Indian logic, linguistics and traditional methods of knowledge and reasoning.
4. Describe selected Indian contributions to technology, medicine, architecture, agriculture and sustainable practices.
5. Relate selected IKS concepts to computational thinking, Indian language computing, AI and digital preservation.

## SYLLABUS

### UNIT – I: Introduction to Indian Knowledge System

**Indian Knowledge System:** Meaning and scope of Indian Knowledge System, Bharatiya Jnana Parampara, Characteristics of Indian knowledge traditions, Holistic and interdisciplinary nature of IKS, Continuity and evolution of Indian knowledge traditions

**Sources of Indian Knowledge:** Vedas and Vedangas, Upavedas, Ramayana and Mahabharata, Puranas, Buddhist and Jain traditions, Classical Sanskrit and regional literature, Oral and community knowledge traditions

**Major Domains:** Mathematics and astronomy, Logic and philosophy, Linguistics and grammar, Medicine and health, Architecture and engineering, Agriculture and environmental knowledge, Arts and performing traditions

**UNIT – II: Indian Mathematics, Astronomy and Scientific Traditions**

**Indian Mathematics:** Place-value and decimal number systems, Concept and development of zero, Contributions of Aryabhata, Brahmagupta and Bhaskara, Arithmetic and algebraic methods, Combinatorics and prosody

**Indian Astronomy:** Development of astronomy in India, Aryabhata and astronomical calculations, Panchanga and calendar systems, Nakshatras and astronomical observations, Indian astronomical instruments

**Scientific Traditions:** Observation and measurement, Mathematical methods, Calculation and prediction, Transmission and development of knowledge

**UNIT – III: Indian Logic, Linguistics and Knowledge Methods**

**Indian Logic:** Introduction to Nyaya, Concept of reasoning, Pramana – means of valid knowledge, Pratyaksha – perception, Anumana – inference, Upamana – comparison, Shabda – verbal testimony

**Indian Linguistics:** Panini and Ashtadhyayi, Sanskrit grammatical tradition, Rules and formal structures, Phonetics and phonology, Word formation and linguistic analysis

**Methods of Reasoning:** Tantrayukti, Purva Paksha and Uttara Paksha, Argument and counter-argument, Structured reasoning and validation of knowledge

**UNIT – IV: Indian Technology, Medicine and Sustainable Practices**

**Traditional Health Knowledge:** Introduction to Ayurveda, Basic concepts of Ayurveda, Preventive health and lifestyle, Traditional knowledge of medicinal plants, Yoga and wellness

**Architecture and Engineering:** Sthapatya traditions, Vastu traditions, Temple architecture, Traditional construction practices, Water management and town planning

**Traditional Technology:** Indian metallurgy, Wootz steel, Agriculture and irrigation, Traditional water harvesting, Sustainable resource management

**UNIT – V: IKS and Contemporary Computing**

**IKS and Computational Thinking:** Systematic reasoning and problem solving, Classification and categorization, Pattern recognition, Rule-based thinking, Problem decomposition, Algorithmic thinking

**Indian Linguistics and Computing:** Paninian grammar and formal rules, Formal grammar and syntax, Parsing – basic concept, Computational linguistics, Indian language computing, Natural Language Processing – introduction

**IKS and Artificial Intelligence:** Knowledge representation, Logic and inference, Rule-based reasoning, Knowledge graphs – introduction, IKS perspectives and AI, Ethical considerations in AI

**Digital Preservation of IKS:** Digitization of manuscripts, OCR for Indian scripts, Digital libraries, Indian-language digital resources, Knowledge graphs, AI/NLP for preservation and dissemination of Indian knowledge

**Text Books:**

1. B. Mahadevan, Vinayak Rajat Bhat and Nagendra Pavana R.N., Introduction to Indian Knowledge System: Concepts and Applications, PHI Learning.
2. Kapil Kapoor and Avadhesh K. Singh (Eds.), Indian Knowledge Systems, D.K. Printworld, 2005.

**Reference Books:**

1. Kim Plofker, Mathematics in India, Princeton University Press, 2009.
2. George Gheverghese Joseph, The Crest of the Peacock: Non-European Roots of Mathematics, Princeton University Press.
3. Dharampal, Indian Science and Technology in the Eighteenth Century.
4. R. M. Pujari, Pradeep Kolhe and N. R. Kumar, Pride of India: A Glimpse into India's Scientific Heritage, Samskrita Bharati.

**Web Resources:**

1. Indian Knowledge Systems – IKS Division, Ministry of Education/AICTE
2. UGC – Indian Knowledge Systems
3. RKM Vivekananda – Indian Knowledge Systems Online Tutorials
4. PHI Learning – *Introduction to Indian Knowledge System*

| II Year II Sem | COMPUTER NETWORKS | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives

The course aims to:

- Introduce the fundamental concepts, components and applications of computer networks and data communication.
- Understand network architectures, protocol layering and the OSI and TCP/IP models.
- Explain data-link layer functions, error control, MAC protocols and basic networking technologies.
- Understand IP addressing, routing and major network-layer protocols.
- Understand TCP/UDP and common application-layer protocols and services.

## Course Outcomes

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

1. Explain the fundamentals of data communication, computer networks, network models and protocol architectures.
2. Explain data-link layer functions, error detection and correction, and medium access control techniques.
3. Explain IP addressing, packet switching, routing and network-layer protocols.
4. Explain transport-layer services and the working of TCP and UDP.
5. Explain common application-layer protocols, network services and basic quality-of-service concepts.

## SYLLABUS

### UNIT – I: Introduction to Computer Networks

Data communication, components and characteristics of data communication, computer networks, need and applications of computer networks, LAN, MAN, WAN and wireless networks, Internet and its basic architecture, network topologies, protocols and standards, protocol layering, OSI reference model, TCP/IP protocol suite, comparison of OSI and TCP/IP models, physical layer, transmission media, bandwidth, data rate, latency, throughput and basic network performance measures.

**UNIT – II: Data Link Layer and MAC**

Data-link layer, services and functions, framing, physical addressing, error detection and correction, parity, checksum, cyclic redundancy check (CRC), flow control, data-link layer protocols, HDLC, Point-to-Point Protocol (PPP), Medium Access Control (MAC), random access protocols, ALOHA, CSMA, CSMA/CD, CSMA/CA, controlled access, polling, token passing, Ethernet.

**UNIT – III: Network Layer and Routing**

Network-layer services, packet switching, logical addressing, IPv4 addressing, classful and classless addressing, subnetting, IPv4 datagram, Internet Protocol (IP), ICMP, routing and forwarding, routing algorithms, distance vector routing, link state routing, unicast routing protocols, introduction to IPv6, IPv6 addressing, advantages of IPv6.

**UNIT – IV: Transport Layer**

Transport-layer services, process-to-process delivery, port numbers, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), TCP services and features, TCP segment, TCP connection establishment and termination, flow control, error control, congestion control, comparison of TCP and UDP.

**UNIT – V: Application Layer and Network Services**

Application-layer services and protocols, World Wide Web, HTTP and HTTPS, electronic mail, SMTP, POP3, IMAP, Domain Name System (DNS), File Transfer Protocol (FTP), DHCP, remote access, client-server applications, peer-to-peer applications, basic network security concepts, Quality of Service (QoS), bandwidth, delay, jitter and packet loss.

## Text Books:

1. Behrouz A. Forouzan, Data Communications and Networking, 5th Edition, McGraw-Hill Education, 2013.
2. James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, 8th Edition, Pearson, 2021.

## Reference Books:

1. Andrew S. Tanenbaum and David J. Wetherall, Computer Networks, 5th Edition, Pearson, 2011.
2. Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Edition, Morgan Kaufmann, 2011.
3. William Stallings, Data and Computer Communications, 10th Edition, Pearson, 2013.

## Web Resources:

1. Cisco Networking Academy
2. GeeksforGeeks – Computer Networks
3. Cloudflare Learning Center
4. Computer Networks: A Systems Approach
5. RFC Editor

| II Year II Sem | JAVA PROGRAMMING | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives

The learning objectives of this course are to:

- Understand the fundamentals of Java programming and object-oriented programming concepts.
- Develop Java programs using classes, objects, methods, inheritance and interfaces.
- Apply arrays, strings, packages, exception handling and file handling in Java applications.
- Understand multithreading and develop basic concurrent Java applications.
- Develop database-driven applications using Java Database Connectivity (JDBC).

## Course Outcomes

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

1. Explain Java programming fundamentals, data types, operators and control structures and develop basic Java programs.
2. Apply classes, objects, constructors, methods, inheritance and interfaces to develop object-oriented Java programs.
3. Develop Java programs using arrays, strings, packages, exception handling and file handling.
4. Implement multithreaded Java applications using threads, synchronization and inter-thread communication.
5. Develop basic database applications using JDBC and perform database operations from Java programs.

**UNIT – I: Java Fundamentals and Control Statements**

**Object-Oriented Programming:** Basic concepts of object-oriented programming, principles of OOP, features of Java, Java program structure, introduction to Java, writing simple Java programs, elements or tokens in Java programs, Java statements, command-line arguments, user input, escape sequences, comments, programming style, **Data Types, Variables and Operators:** data types in Java, declaration and initialization of variables, type casting, scope of variables, literal constants, symbolic constants, static variables and methods, final keyword, arithmetic

operators, assignment operators, increment and decrement operators, relational operators, logical operators, bitwise operators, ternary operator, operator precedence and associativity, **Control Statements:** if, if-else, nested if, switch, while, do-while, for, nested loops, enhanced for loop, break and continue statements.

**UNIT – II: Classes, Objects and Methods**

**Classes and Objects:** Class declaration, class modifiers, class members, creation of objects, accessing class members, access control, constructors, parameterized constructors, constructor overloading, assigning objects, keyword this, static members, final members and classes, passing arguments to methods, **Methods:** defining and invoking methods, method parameters and return values, method overloading, objects as method parameters, recursive methods, method overriding, access control, static and final methods, introduction to nested classes.

**UNIT – III: Arrays, Inheritance and Interfaces**

**Arrays:** Declaration and initialization of arrays, accessing array elements, operations on arrays, one-dimensional arrays, two-dimensional arrays, multidimensional arrays, arrays of objects, passing arrays to methods, searching and sorting arrays, **Inheritance:** introduction to inheritance, superclass and subclass, types of inheritance supported in Java, Object class, access control and inheritance, multilevel inheritance, super keyword, constructors and inheritance, method overriding, dynamic method dispatch, final classes and methods, abstract classes, **Interfaces:** declaration and implementation of interfaces, multiple interfaces, inheritance of interfaces, default methods, static methods in interfaces, functional interfaces, annotations.

**UNIT – IV: Packages, Exception Handling and File Handling**

**Packages and Java Libraries:** Defining packages, importing packages and classes, package access control, Java standard library, java.lang package, Object class, Math class, wrapper classes, autoboxing and unboxing, java.util package, Formatter class, Random class, date and time API, **Exception Handling:** exception hierarchy, Throwable class, checked and unchecked exceptions, try, catch and finally blocks, multiple catch clauses, throw and throws keywords, user-defined exceptions, **Java I/O and File Handling:** Java I/O API, standard input and output streams, byte streams, character streams, Scanner class, file creation and handling, reading and writing text files.

**UNIT – V: Strings, Multithreading and JDBC**

**String Handling:** CharSequence interface, String class, string creation, extracting characters, string comparison, searching and modifying strings, StringBuffer and StringBuilder, **Multithreaded Programming:** need for multithreading, processes and threads, Thread class, creating threads, thread life cycle, thread states, thread priorities, synchronization, race conditions, deadlock, inter-thread communication, **Java Database Connectivity:** introduction to JDBC, JDBC architecture, JDBC drivers, database connectivity, establishing database connections, executing SQL statements, PreparedStatement, ResultSet, inserting, updating, deleting and retrieving records, developing simple database applications using Java and MySQL.

## Text Books:

1. Anitha Seth and B. L. Juneja, Java: One Step Ahead, Oxford University Press.
2. Debasis Samanta and Monalisa Sarma, Joy with Java: Fundamentals of Object Oriented Programming, Cambridge University Press, 2023.
3. Paul Deitel and Harvey Deitel, Java: How to Program, Pearson.

## Reference Books:

1. Herbert Schildt, Java: The Complete Reference, 11th Edition, McGraw-Hill.
2. Y. Daniel Liang, Introduction to Java Programming and Data Structures, Pearson.
3. Cay S. Horstmann, Core Java, Volume I – Fundamentals, Pearson.

## Online Resources:

1. NPTEL – Programming in Java
2. Infosys Springboard
3. Oracle Java Documentation
4. W3Schools – Java Tutorial

| II Year II Sem | WEB TECHNOLOGIES | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives

The learning objectives of this course are to:

- Understand the fundamentals of the Internet, World Wide Web and web application architecture.
- Develop web pages using HTML5 and CSS3 with appropriate page structure, forms and multimedia.
- Develop interactive web pages using JavaScript, DOM and client-side form validation.
- Understand XML and server-side web programming concepts using Servlets and JSP.
- Develop basic dynamic web applications using PHP and MySQL with sessions, cookies and file handling.

## Course Outcomes

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

1. Explain Internet and Web concepts and develop structured web pages using HTML5.
2. Design responsive and presentable web pages using CSS and apply XML for structured data representation.
3. Develop interactive client-side web applications using JavaScript, DOM and form validation.
4. Explain and implement basic server-side web applications using Servlets and JSP.
5. Develop simple dynamic web applications using PHP and MySQL with database, session, cookie and file handling.

**UNIT – I: Internet, HTML and CSS**

**Web Basics:** Internet and World Wide Web, history and evolution of the Internet, Internet protocols, client-server architecture, web server, web browser, URL, domain names, HTTP and HTTPS, **HTML5:** introduction to HTML, structure of an HTML document, elements and attributes, headings, paragraphs, text formatting, lists, links, images, audio and video, tables, forms and form controls, semantic elements, HTML5 input types, **CSS:** introduction to CSS, inline, internal and external stylesheets, selectors, colors, fonts, text and background properties,

borders, margins and padding, box model, positioning, basic layouts, responsive web design concepts.

**UNIT – II: XML and Web Data Representation**

**XML:** Introduction to XML, features and applications of XML, XML documents and syntax, elements, attributes and values, well-formed XML documents, namespaces, Document Type Definition (DTD), XML Schema, XML validation, XML DOM, processing XML documents, DOM and SAX parsers, XML and HTML comparison, XML applications in web-based data exchange.

**UNIT – III: JavaScript and Client-Side Scripting**

**JavaScript:** Introduction to JavaScript, embedding JavaScript in HTML, variables and constants, data types, operators and expressions, conditional and looping statements, functions, arrays, strings and objects, **Document Object Model (DOM):** DOM structure, accessing and modifying HTML elements, creating and modifying elements, event handling, mouse and keyboard events, form events, **Form Validation:** validating text fields, numbers, email and other form controls, displaying validation messages, basic client-side interaction and dynamic web page development.

**UNIT – IV: Server-Side Web Programming with Servlets and JSP**

**Servlets:** Introduction to server-side programming, CGI and Servlets, Servlet architecture, Servlet life cycle, Servlet API, creating and deploying Servlets, handling HTTP requests and responses, reading form parameters, initialization parameters, request and response objects, cookies, sessions and session tracking, connecting Servlets to databases using JDBC, **JSP:** introduction to JSP, JSP architecture and life cycle, anatomy of a JSP page, declarations, expressions, scriptlets, directives, implicit objects, JSP actions, JavaBeans, cookies and session tracking, database connectivity using JDBC, comparison of Servlets and JSP.

**UNIT – V: PHP and MySQL**

**PHP:** Introduction to PHP, PHP syntax and program structure, variables and constants, data types, operators and expressions, conditional and looping statements, arrays, strings, functions, handling HTML form data, GET and POST methods, form validation, **PHP and MySQL:** connecting PHP with MySQL, creating and selecting databases and tables, executing SQL queries, inserting, updating, deleting and retrieving records, processing query results, **Sessions and Cookies:** session

management, cookies, **File Handling:** opening, closing, reading, writing, appending and deleting files, handling text and binary files, file upload and download, directory handling, developing a simple database-driven web application using PHP and MySQL.

## Text Books:

1. Uttam K. Roy, Web Technologies, Oxford University Press.
2. Robert W. Sebesta, Programming the World Wide Web, 9th Edition, Pearson.
3. Steven Holzner, PHP: The Complete Reference, McGraw-Hill Education.

## Reference Books:

1. Jon Duckett, HTML and CSS: Design and Build Websites, Wiley.
2. Jon Duckett, JavaScript and JQuery: Interactive Front-End Web Development, Wiley.
3. Hans Bergsten, JavaServer Pages, O'Reilly Media.
4. David Flanagan, JavaScript: The Definitive Guide, O'Reilly Media.

## Online Resources:

1. MDN Web Docs
2. PHP Manual
3. MySQL Documentation
4. Oracle Java Documentation
5. W3Schools

| II Year II Sem | SOFTWARE ENGINEERING | L | T | P | C |
|---|---|---|---|---|---|
| | | 3 | 0 | 0 | 3 |

## Course Objectives

The objectives of this course are to:

- Understand the fundamentals of software engineering and software development life cycle models.
- Understand software project planning, requirements engineering and preparation of Software Requirements Specification (SRS).
- Apply software design principles, Agile methodologies and modelling techniques for software development.
- Understand software testing, debugging, software quality and reliability concepts.
- Understand software maintenance, software reuse and contemporary practices in software engineering.

## Course Outcomes

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

1. Explain software engineering principles, software processes and different software development life cycle models.
2. Analyze software requirements and prepare an appropriate Software Requirements Specification (SRS).
3. Apply software design principles, UML/structured modelling and Agile practices in software development.
4. Apply software testing techniques and explain software quality and reliability concepts.
5. Explain software maintenance, software reuse, configuration management and project management practices.

**UNIT – I: Introduction to Software Engineering and Process Models**

**Introduction:** Evolution of software, software development projects, software characteristics, software engineering concepts, software crisis, emergence of software engineering, software development activities, software process, computer system engineering, **Software Life Cycle**

**Models:** basic concepts, software process models, Waterfall model and its extensions, Prototyping model, Incremental and Iterative development, Rapid Application Development, Spiral model, Agile development model, comparison of software life cycle models, selection of an appropriate process model.

**UNIT – II: Software Project Management and Requirements Engineering**

**Software Project Management:** Software project management concepts, project planning, responsibilities of a software project manager, project size estimation, cost and effort estimation, LOC and Function Point estimation, COCOMO model, scheduling, resource allocation, risk identification and management, project monitoring and control, **Requirements Engineering:** requirements gathering and elicitation, stakeholders, functional and non-functional requirements, requirements analysis, feasibility study, requirements validation, requirements management, Software Requirements Specification (SRS), characteristics of a good SRS, SRS structure, use cases and basic requirements modelling.

**UNIT – III: Software Design, Modelling and Agile Development**

**Software Design:** design process, design principles, characteristics of good software design, modularity, abstraction, information hiding, cohesion and coupling, architectural design, layered architecture, component-based design, **Software Modelling:** introduction to UML, use case diagrams, class diagrams, sequence diagrams, activity diagrams, basic software design documentation, **Agile Software Development:** Agile principles and values, Agile manifesto, Agile process, Scrum framework, roles and activities in Scrum, user stories, product backlog, sprint, sprint planning, review and retrospective, Extreme Programming (XP), advantages and limitations of Agile methods.

**UNIT – IV: Software Coding, Testing and Quality Management**

**Coding:** coding standards, programming practices, code review, software documentation, **Software Testing:** objectives and principles of testing, verification and validation, test planning, test cases, black-box testing, white-box testing, equivalence partitioning, boundary value analysis, control flow testing, unit testing, integration testing, system testing, acceptance testing, regression testing, smoke testing, debugging, testing object-oriented programs, **Software Quality:** software quality characteristics, software quality assurance, software quality management, software

reliability, quality standards, ISO 9000, Capability Maturity Model (CMM), Six Sigma, continuous improvement.

**UNIT – V: Software Maintenance, Reuse and Configuration Management**

**Software Maintenance:** need for maintenance, characteristics of software maintenance, corrective, adaptive, perfective and preventive maintenance, software maintenance process, maintenance activities, maintenance cost estimation, software evolution, legacy systems, software reverse engineering, **Software Reuse:** concept of software reuse, benefits and challenges, reusable components, component-based development, reuse approaches, reuse at organization level, **Software Configuration Management:** configuration items, version control, change management, baseline, configuration management process, software documentation, introduction to CASE tools and their applications in software development.

**Text Books:**

1. Rajib Mall, Fundamentals of Software Engineering, 5th Edition, PHI Learning.
2. Roger S. Pressman and Bruce R. Maxim, Software Engineering: A Practitioner's Approach, 9th Edition, McGraw-Hill.

**Reference Books:**

1. Ian Sommerville, Software Engineering, 10th Edition, Pearson.
2. Deepak Jain, Software Engineering: Principles and Practices, Oxford University Press.
3. Pankaj Jalote, An Integrated Approach to Software Engineering, Narosa Publishing House.
4. Carlo Ghezzi, Mehdi Jazayeri and Dino Mandrioli, Fundamentals of Software Engineering, 2nd Edition, Pearson.

**Online Resources:**

1. NPTEL – Software Engineering — lectures on software processes, requirements, design, testing and project management.
2. Infosys Springboard — software engineering learning modules and learning materials.
3. Agile Manifesto — Agile principles and values.
4. Scrum Guide — Scrum framework, roles, events and artifacts.
5. Object Management Group – UML — UML concepts and specifications.
6. IEEE Computer Society — software engineering standards and professional resources.

| II Year II Sem | COMPUTER NETWORKS LAB | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives

The objectives of this course are to:

- Understand basic computer networking concepts, devices and TCP/IP networking.
- Gain practical knowledge of data-link layer framing and error detection techniques.
- Implement basic flow control, sliding window and congestion control algorithms.
- Implement routing algorithms and understand their practical applications.
- Use basic network commands and tools to analyze network connectivity and protocols.

## Course Outcomes

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

1. Identify networking devices and configure a basic Local Area Network using TCP/IP.
2. Implement data-link layer framing and error detection techniques.
3. Implement flow control, sliding window and congestion control algorithms.
4. Implement shortest-path and routing algorithms used in computer networks.
5. Use basic network utilities and tools to analyze network connectivity and network behaviour.

## List of Experiments

### Experiment 1: Network Devices and LAN Configuration

Study and identify network devices such as hub, switch, router, modem, repeater and access point, study different types of network cables and connectors, connect computers in a Local Area Network, configure IPv4 addresses, subnet mask and default gateway, test connectivity using basic network commands.

### Experiment 2: Network Utilities and TCP/IP Analysis

Study and use basic networking commands such as ipconfig/ifconfig, ping, tracert/traceroute, nslookup, netstat and arp, examine IP address, MAC address and routing information, analyze basic network connectivity.

**Experiment 3: Data Link Layer Framing**

Implement framing techniques using character stuffing and bit stuffing, demonstrate frame formation and recovery of the original data.

**Experiment 4: Error Detection using Checksum and CRC**

Implement checksum-based error detection, implement CRC using standard CRC polynomials, verify detection of errors in transmitted data.

**Experiment 5: Error Detection and Correction using Hamming Code**

Implement Hamming Code generation, introduce errors in transmitted data, detect and correct single-bit errors.

**Experiment 6: Stop-and-Wait Protocol**

Implement the Stop-and-Wait protocol, demonstrate transmission of frames, acknowledgement, timeout and retransmission.

**Experiment 7: Sliding Window Protocol – Go-Back-N**

Implement the Go-Back-N sliding window protocol, simulate frame transmission, acknowledgement, timeout and retransmission of lost or damaged frames.

**Experiment 8: Sliding Window Protocol – Selective Repeat**

Implement the Selective Repeat sliding window protocol, simulate selective retransmission of lost or damaged frames and compare its operation with Go-Back-N.

**Experiment 9: Congestion Control using Leaky Bucket**

Implement the Leaky Bucket algorithm, simulate packet transmission, control the output rate and demonstrate handling of bursty traffic.

**Experiment 10: Shortest Path Routing using Dijkstra's Algorithm**

Implement Dijkstra's algorithm to find the shortest path between nodes in a weighted network graph, display the shortest path and corresponding cost.

**Experiment 11: Distance Vector Routing**

Implement the Distance Vector Routing algorithm, construct routing tables for network nodes, update routing information based on neighbouring nodes and display the final routing tables.

**Experiment 12: Mini Network Configuration and Analysis**

Design and configure a small LAN, assign IP addresses, verify connectivity, examine MAC and IP addressing, perform basic troubleshooting using network utilities and document the network configuration.

## Text Books:

1. Behrouz A. Forouzan, Data Communications and Networking, 5th Edition, McGraw-Hill Education.
2. James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, 8th Edition, Pearson.

## Reference Books:

1. Andrew S. Tanenbaum and David J. Wetherall, Computer Networks, 5th Edition, Pearson.
2. Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Edition, Morgan Kaufmann.
3. William Stallings, Data and Computer Communications, 10th Edition, Pearson.

## Online Resources:

1. Cisco Networking Academy
2. Computer Networks: A Systems Approach
3. Wireshark
4. RFC Editor
5. GeeksforGeeks – Computer Networks

| II Year II Sem | ANDROID APP DEVELOPMENT | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives

The objectives of this course are to:

- Understand the basic components and architecture of Android applications.
- Develop Android applications using activities, layouts, views, menus and intents.
- Implement data storage using SharedPreferences, files and SQLite databases.
- Develop applications using multimedia, web, location and device services.
- Build, test and deploy simple Android applications.

## Course Outcomes

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

1. Develop basic Android applications using Android Studio and standard UI components.
2. Design Android user interfaces using layouts, views, menus, adapters and fragments.
3. Implement activities, intents, data storage and SQLite database operations in Android applications.
4. Develop applications using multimedia, web, location and selected device services.
5. Build, test and package simple Android applications for practical use.

## List of Experiments

1. Create an Android application to display a welcome message using TextView and Button.
2. Create an Android application to demonstrate Explicit and Implicit Intents.
3. Create an Android application using different layouts and basic UI components.
4. Create an Android application to display a list of items using ListView and ArrayAdapter.
5. Create an Android application to display items using a Custom Adapter.
6. Create an Android application using Spinner, AutoCompleteTextView and other selection controls.
7. Create an Android application using Fragments to display different screens.
8. Create an Android application to open and display a web page using WebView.
9. Create an Android application to store and retrieve data using SharedPreferences and local files.

**10.** Create an Android application to perform basic CRUD operations using SQLite database.

**11.** Create an Android application to play audio and video files.

**12.** Create an Android application to record audio or video using device facilities.

**13.** Create an Android application to display the current location using Location Services.

**14.** Create an Android application to display a location on Google Maps.

**15.** Create an Android application to demonstrate notifications and basic device services.

**16.** Create an Android application to access and display available Wi-Fi or Bluetooth devices.

**17.** Create an Android application to send an email using an Intent.

**18.** Develop a simple Android application as a mini project using UI components, data storage and one device or multimedia feature.

**Recommended Software**

Android Studio, Android SDK, Android Emulator or physical Android device, Kotlin or Java, Flutter

**Text Books:**

1. Neil Smyth, Android Studio Development Essentials, Payload Media.
2. Bill Phillips, Chris Stewart, Kristin Marsicano and Brian Gardner, Android Programming: The Big Nerd Ranch Guide, 5th Edition, Big Nerd Ranch Guides.

**Reference Books:**

1. Reto Meier and Ian Lake, Professional Android, 4th Edition, Wrox.
2. Dawn Griffiths and David Griffiths, Head First Android Development, 3rd Edition, O'Reilly Media.
3. Pradeep Kothari, *Android* Application Development (with KitKat Support), Black Book, Kogent Learning Solutions.

**Online Resources:**

1. Android Developers – Official Documentation
2. Android Developers – Training
3. Android Studio – Official Site
4. Kotlin Documentation
5. Android Developers – Jetpack Compose
6. Firebase Documentation

| II Year II Sem | JAVA PROGRAMMING LAB | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives

The objectives of this course are to:

- Develop problem-solving skills using Java programming.
- Implement object-oriented programming concepts using classes, objects and methods.
- Implement inheritance, interfaces, polymorphism and exception handling.
- Develop programs using packages, strings and multithreading.
- Develop simple database applications using JDBC.

## Course Outcomes

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

1. Develop Java programs using basic programming constructs, arrays and strings.
2. Implement classes, objects, constructors, methods and inheritance in Java.
3. Apply interfaces, polymorphism and exception handling in Java programs.
4. Develop Java applications using packages and multithreading.
5. Develop simple database applications using JDBC.

## List of Experiments

**Experiment 1: Java Basics**

a) Write a Java program to display the default values of primitive data types.

b) Write a Java program to find the roots of a quadratic equation.

c) Write a Java program to find the largest of three numbers using conditional statements.

**Experiment 2: Arrays and Searching**

a) Write a Java program to perform linear search on an array.

b) Write a Java program to perform binary search on a sorted array.

c) Write a Java program to find the largest, smallest and average elements of an array.

**Experiment 3: Sorting and String Handling**

a) Write a Java program to sort elements using Bubble Sort.

b) Write a Java program to perform basic operations on strings.

c) Write a Java program using StringBuffer or StringBuilder to insert, delete and modify characters.

**Experiment 4: Classes, Objects and Constructors**

a) Write a Java program to create a class with data members and methods and invoke them from the main method.

b) Write a Java program to implement method overloading.

c) Write a Java program to implement constructors.

d) Write a Java program to implement constructor overloading.

**Experiment 5: Inheritance and Polymorphism**

a) Write a Java program to implement single inheritance.

b) Write a Java program to implement multilevel inheritance.

c) Write a Java program to demonstrate the use of the super keyword.

d) Write a Java program to implement runtime polymorphism.

**Experiment 6: Abstract Classes and Interfaces**

a) Write a Java program using an abstract class to calculate the areas of different shapes.

b) Write a Java program to implement an interface.

c) Write a Java program to implement multiple interfaces.

**Experiment 7: Exception Handling**

a) Write a Java program to demonstrate exception handling using try, catch and finally.

b) Write a Java program to demonstrate multiple catch blocks.

c) Write a Java program to handle built-in exceptions.

d) Write a Java program to create and handle a user-defined exception.

**Experiment 8: Packages and Multithreading**

a) Write a Java program to create and use a user-defined package.

b) Write a Java program to create threads by extending the Thread class.

c) Write a Java program to create threads using the Runnable interface.

d) Write a Java program to demonstrate isAlive() and join() methods.

**Experiment 9: JDBC Database Connectivity**

a) Write a Java program to establish a connection with a MySQL database using JDBC.
b) Write a Java program to insert records into a database using JDBC.
c) Write a Java program to update and delete records using JDBC.
d) Write a Java program to retrieve and display records using JDBC.

**Experiment 10: Mini Application**

Design and develop a simple Java application using OOP concepts, exception handling and JDBC.

**Examples:** Student Management System, Library Management System, Employee Management System or Simple Banking Application.

**Text Books:**

1. Herbert Schildt, Java: The Complete Reference, 12th Edition, McGraw-Hill.
2. Cay S. Horstmann, Core Java: Fundamentals, 12th Edition, Pearson.

**Reference Books:**

1. Paul Deitel and Harvey Deitel, Java: How to Program, 11th Edition, Pearson.
2. Y. Daniel Liang, Introduction to Java Programming and Data Structures, 12th Edition, Pearson.
3. Kathy Sierra and Bert Bates, Head First Java, 2nd Edition, O'Reilly Media.

**Online Resources:**

1. Oracle Java Documentation
2. Oracle Java Tutorials
3. W3Schools – Java
4. GeeksforGeeks – Java
5. MySQL Documentation

| II Year II Sem | WEB TECHNOLOGIES LAB | L | T | P | C |
|---|---|---|---|---|---|
| | | 0 | 0 | 3 | 1.5 |

## Course Objectives

The objectives of this course are to:

- Develop static and responsive web pages using HTML5 and CSS.
- Develop interactive web pages using JavaScript and DOM.
- Create and process XML documents for structured data representation.
- Develop server-side web applications using PHP and MySQL.
- Understand and develop basic dynamic web applications using Servlets, JSP and JDBC.

## Course Outcomes

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

1. Design web pages using HTML5, CSS and responsive layouts.
2. Develop interactive web pages using JavaScript, DOM and form validation.
3. Create and process well-formed XML documents.
4. Develop dynamic web applications using PHP and MySQL.
5. Develop basic dynamic web applications using Servlets, JSP and JDBC.

## List of Experiments

### Experiment 1: HTML and CSS Web Pages

Develop static web pages for an online book store using HTML5 and CSS. The website should include a home page, registration page, login page, user profile page, books catalogue, shopping cart and order confirmation page.

### Experiment 2: HTML Forms and CSS

Create a registration and login form using HTML5 and CSS. Apply appropriate input types, form controls, styles and basic responsive design.

**Experiment 3: JavaScript Basics**

Write JavaScript programs to demonstrate variables, operators, conditional statements, loops, functions, arrays and strings.

**Experiment 4: JavaScript Form Validation**

Validate registration and login forms using JavaScript. Check required fields, email, password, phone number and other form inputs.

**Experiment 5: JavaScript and DOM**

Create a web page using JavaScript and DOM to dynamically change HTML elements, styles and content in response to user events.

**Experiment 6: XML Document**

Create a well-formed XML document containing information about books or users. Validate the XML document using DTD or XML Schema.

**Experiment 7: XML Processing**

Write a program to read an XML document and display the required information based on a given user ID or book ID.

**Experiment 8: PHP Basics and File Handling**

Write PHP programs to demonstrate variables, data types, operators, control statements, functions and strings. Write a PHP program to read data from one file and write it into another file.

**Experiment 9: PHP Forms and Database**

Create a PHP registration form and store the submitted user details in a MySQL database. Retrieve and display the stored details.

**Experiment 10: PHP and MySQL CRUD Operations**

Develop a PHP application to perform insert, update, delete and retrieve operations on a MySQL database.

**Experiment 11: Sessions, Cookies and File Upload**

Develop a PHP application to demonstrate sessions and cookies. Implement a simple file upload and file handling operation.

**Experiment 12: Servlets and JSP**

Develop a simple dynamic web application using Servlets and JSP. Implement form processing, session management and database connectivity using JDBC.

**Experiment 13: Mini Web Application**

Develop a simple database-driven web application using HTML, CSS, JavaScript, PHP and MySQL.

**Examples:** Online Book Store, Student Management System, Library Management System, Online Registration System or Employee Management System.

**Text Books:**

1. Uttam K. Roy, Web Technologies, Oxford University Press.
2. Steven Holzner, PHP: The Complete Reference, McGraw-Hill Education.
3. Robert W. Sebesta, Programming the World Wide Web, 9th Edition, Pearson.

**Reference Books:**

1. Jon Duckett, HTML and CSS: Design and Build Websites, Wiley.
2. Jon Duckett, JavaScript and JQuery: Interactive Front-End Web Development, Wiley.
3. David Flanagan, JavaScript: The Definitive Guide, O'Reilly Media.
4. Hans Bergsten, JavaServer Pages, O'Reilly Media.

**Online Resources:**

1. MDN Web Docs
2. PHP Manual
3. MySQL Documentation
4. Oracle Java Documentation
5. W3Schools