ANDHRA KESARI UNIVERSITY:: ONGOLE

Model Syllabus for 4-Year UG Honours in B.C.A. (Computer Applications) as Major in consonance with Curriculum framework w.e.f. AY 2025-26

COURSE STRUCTURE (for Semester I to VI)

Year	Semester	Course	Title of the Course	No. of Hrs /Week	No. of Credits
I	I	1	Computer Fundamentals and Office Automation	3	3
			Computer Fundamentals and Office Automation-Practical	2	1
		2	Problem Solving Using C	3	3
			Problem Solving Using C-Practical	2	1
	II	3	Data Structures using C	3	3
			Data Structures using C-Practical	2	1
		4	Database Management System	3	3
			Database Management System-Practical	2	1
II	III	5	OOPS Through JAVA	3	3
			OOPS Through JAVA-Practical	2	1
		6	Computer Organisation	3	3
			Computer Organisation-Practical	2	1
		7	Probability and Statistics	4	4
	IV	8	Operating Systems	3	3
			Operating Systems-Practical	2	1
		9	Python Programming	3	3
			Python Programming-Practical	2	1
		10	Software Engineering	3	3
			Software Engineering-Practical	2	1
III	V	11	Data Mining	3	3
			Data Mining-Practical	2	1

Signatures of BoS Members

1) Approved 10/09/2025

2) 10/09/2025

3) [illegible]

4) V. Sateesh 10/09/25

Year	Semester	Course	Title of the Course	No. of Hrs /Week	No. of Credits
		12 A	Web Interface Design Technologies	3	3
			Web Interface Design Technologies-Practical	2	1
			OR		
		12 B	Computer Networks	3	3
			Computer Networks Lab-Practical	2	1
		13 A	Web Application Development using PHP & MySQL	3	3
			Web Application Development using PHP & MySQL-Practical	2	1
			OR		
		13 B	Cyber Security	3	3
			Cyber Security-Practical	2	1
	VI	14 A	Mobile Application Development	3	3
			Mobile Application Development-Practical	2	1
			OR		
		14 B	Cloud Fundamentals and Security	4	4
		15 A	MERN Stack	3	3
			MERN Stack-Practical	2	1
			OR		
		15 B	Digital Forensics	3	3
			Digital Forensics-Practical	2	1

Note: In the III Year (during the V and VI Semesters), students are required to select a pair of electives from one of the Two specified domains. **For example: if set 'A' is chosen, courses 12 to 15 to be chosen as 12 A, 13 A, 14 A and 15 A.** To ensure in-depth understanding and skill development in the chosen domain, students must continue with the same domain electives in both the V and VI Semesters.

SEMESTER-III

COURSE 5: OOPS THROUGH JAVA

Theory	Credits: 3	3 hrs/week

Course Objectives

1. Introduce core OOP principles and contrast procedural and object-oriented paradigms within the Java ecosystem.
2. Equip learners with foundational and advanced Java syntax including variables, control statements, arrays, strings, and classes.
3. Enable the use of inheritance, polymorphism, interfaces, and exception handling to create maintainable and reusable code.
4. Train students in concurrent programming and stream-based I/O operations including file management and serialization.
5. Empower learners to design, build, and manage GUI programs using Swing components, layout managers, and event handling techniques.

Course Outcomes

At the End of the Course, The Students will be able to:

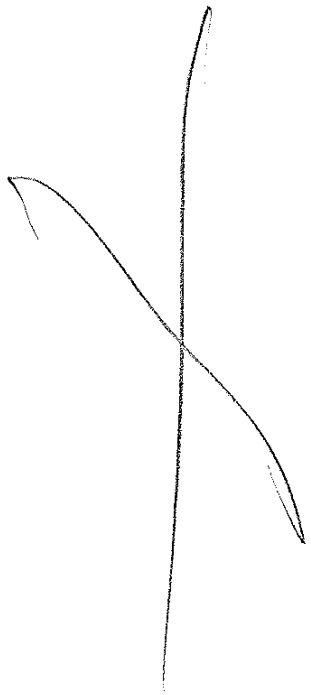
1. Apply OOP principles such as encapsulation, inheritance, and polymorphism in Java applications.
2. Write, compile, and debug Java code using control statements, arrays, classes, and methods effectively.
3. Construct modular code leveraging interfaces, abstract classes, and package hierarchies.
4. Manage thread lifecycles, synchronization, and I/O streams for file handling and console interaction.
5. Design user interfaces using Swing and handle keyboard/mouse input through event-driven programming.

Unit 1. OOPs Concepts and Java Programming:
Introduction to Object-Oriented concepts, procedural and object-oriented programming paradigm

Java programming: An Overview of Java, Java Environment, Data types, Variables, constants, scope and life time of variables, operators, type conversion and casting, Accepting Input from the Keyboard, Reading Input with Java.util.Scanner Class, Displaying Output, Displaying Formatted Output with String.format(), Control Statements

Unit 2. Arrays and OOP Constructs:
Arrays, Command Line Arguments, Strings-String Class Methods

Classes & Objects: Creating Classes, declaring objects, Methods, parameter passing, static fields and methods, Constructors, and 'this' keyword, overloading methods and access Inheritance:



Inheritance hierarchies, super and subclasses, member access rules, 'super' keyword, preventing inheritance: final classes and methods, the object class and its methods; Polymorphism: Dynamic binding, method overriding, abstract classes and methods;

Unit 3. Interfaces, Packages & Exception Handling:
Interfaces VS Abstract classes, defining an interface, implement interfaces, accessing implementations through interface references, extending interface;

Packages: Defining, creating and accessing a package, importing packages.

Exception Handling: Benefits of exception handling, the classification of exceptions, exception hierarchy, checked exceptions and unchecked exceptions, usage of try, catch, throw, throws and finally, rethrowing exceptions, exception specification, built in exceptions, creating own exception sub classes.

Unit 4. Multithreading & Stream based I/O:
Differences between multiple processes and multiple threads, thread states, thread life cycle, creating threads, interrupting threads, thread priorities, synchronizing threads, inter thread communication.

Stream based I/O (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output, File class, Reading and writing Files, The Console class, Serialization

Unit 5. GUI Programming with Swing:
Introduction, MVC architecture, components, containers. Understanding Layout Managers - Flow Layout, Border Layout, Grid Layout, Card Layout, GridBag Layout. Event Handling- The Delegation event model- Events, Event sources, Event Listeners, Event classes, Handling mouse and keyboard events, Adapter classes.

Text Books:

1. Java The complete reference, Herbert Schildt, 9th edition, McGraw Hill.
2. Programming in Java, S. Malhotra, S. Chudhary, 2nd edition, Oxford Univ. Press.

Reference Books:

1. Programming with JAVA - A Primer, E Balaguruswamy, 3rd Edition, McGraw Hill
2. Head First Java: A Brain-Friendly Guide , Katty Sierra, Bert Bates, 2nd Edition, O'Reilly

Activities:

Outcome: Apply OOP principles such as encapsulation, inheritance, and polymorphism in Java applications

Activity: Develop a class hierarchy for a zoo management system using inheritance and polymorphism (e.g., Animal → Mammal → Dog). Implement encapsulation through private fields and public getters/setters.

Evaluation Method: Code review and oral explanation focusing on class relationships, method overriding, and encapsulation practices.

Outcome: Write, compile, and debug Java code using control statements, arrays, classes, and methods effectively
Activity: Create a console-based student grade calculator using loops, conditionals, arrays, and modular methods.
Evaluation Method: Practical test with debugging tasks and output validation across multiple input scenarios.

Outcome: Construct modular code leveraging interfaces, abstract classes, and package hierarchies
Activity:
Design a payment processing system with abstract classes for Payment, interfaces for Taxable, and organize classes into packages (e.g., com.billing, com.tax).
Evaluation Method:
Project submission assessed for modularity, interface implementation, abstraction usage, and package structure.

Outcome: Manage thread lifecycles, synchronization, and I/O streams for file handling and console interaction
Activity: Build a multithreaded logger that reads input from the console and writes to a file using synchronized threads and buffered streams.
Evaluation Method: Lab demonstration with thread state tracing and file output verification under concurrent input.

Outcome: Design user interfaces using Swing and handle keyboard/mouse input through event-driven programming
Activity: Create a GUI-based quiz application using Swing components (JFrame, JButton, JTextField) with event listeners for mouse clicks and key presses.
Evaluation Method: Live demo and rubric-based assessment of UI responsiveness, event handling accuracy, and layout design.

SEMESTER-III

COURSE 5: OOPS THROUGH JAVA

Practical **Credits: 1** **2 hrs/week**

List of Experiments

1. Write a Java program to print Fibonacci series.
2. Write a Java program to calculate multiplication of 2 matrices.
3. Write a Java program for sorting a given list of names in ascending order.
4. Create a class Rectangle. The class has attributes length and width. It should have methods that calculate the perimeter and area of the rectangle. It should have readAttributes() method to read length and width from the user.
5. Write a Java program that implements method overloading.
6. Write a Java program to implement various types of inheritance
   i. Single ii. Multi-Level iii. Hierarchical iv. Hybrid
7. Write a java program to implement runtime polymorphism.
8. Write a Java program which accepts withdrawal amount from the user and throws an exception In Sufficient Funds when withdrawal amount is more than available amount.
9. Write a Java program to create three threads and that displays good morning, for every one second, hello for every 2 seconds and welcome for every 3 seconds by using extending Thread class.
10. Write a Java program that creates three threads. The first thread displays OOPS, the second thread displays Through and the third thread displays JAVA by using Runnable interface.
11. Write a Java program that displays the number of characters, lines and words in a text file.
12. Implement a Java program for handling mouse events when the mouse entered, exited, clicked, pressed, released, dragged and moved in the client area.
13. Implement a Java program for handling key events when the key board is pressed, released, typed.
14. Write a Java swing program that reads two numbers from two separate text fields and displays the sum of two numbers in the third text field when the button add is pressed.
15. Write a Java program to design student registration form using Swing Controls. The form must have the following fields and button SAVE
    Form Fields are: Name, RNO, Mailid, Gender, Branch, Address.

SEMESTER-III

COURSE 6: COMPUTER ORGANIZATION

Theory	Credits: 3	3 hrs/week

Course Objectives

1. To introduce foundational concepts of register transfer language and micro-operations, enabling understanding of basic computer organization.
2. To examine CPU architecture and the control unit's design through hardwired and microprogrammed approaches.
3. To explore various memory organization strategies, including hierarchy, cache mapping, and associative techniques.
4. To understand input-output systems and data transfer mechanisms using I/O interfaces and DMA methods.
5. To analyze arithmetic algorithms and the principles of parallel and pipelined processing.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Interpret register-level operations and perform arithmetic, logic, and shift micro-operations within a basic computer framework.
2. Illustrate CPU functionality, addressing modes, and control unit design with both hardware-based and microprogramming techniques.
3. Categorize types of memory and explain memory hierarchy, access methods, and mapping techniques.
4. Describe I/O organization, including asynchronous transfer modes, DMA, and interrupt-driven processing.
5. Apply arithmetic algorithms and demonstrate understanding of parallel and pipelined processing models.

Unit 1. Register Transfer Language and Micro Operations:
Introduction- Functional units, computer registers, register transfer language, register transfer, bus and memory transfers, arithmetic, logic and shift micro-operations, arithmetic logic shift unit. Basic Computer Organization and Design: Instruction codes, instruction cycle. Register reference instructions, Memory – reference instructions, input – output and interrupt.

Unit 2. CPU and Micro Programmed Control:
Central Processing unit: Introduction, instruction formats, addressing modes.Control memory, address sequencing, design of control unit - hard wired control, micro programmed control.

Unit 3. Memory Organization:
Memory hierarchy, main memory, auxiliary memory, associative memory, cache Memory and mappings.

Unit 4. Input-Output Organization:
Peripheral Devices, input-output interface, asynchronous datatransfer, modes of transfer-programmed I/O, priority interrupt, direct memory access, Input – Output Processor (IOP).

Unit 5. Computer Arithmetic:
Data representation- fixed point, floating point, addition and subtraction, multiplication and division algorithms.

Text Books:

1. Computer Systems Architecture, M. Moris Mano, 3rd edition, Pearson/ PHI
2. Computer Organization and Architecture, William Stallings, 8th edition, Pearson/PHI

Reference Books:

1. Computer Organisation and Architecture, V. Rajaraman, T. Radha Krishnan, PHI
2. Computer Organization and Architecture Hamacher, Vranesic, Zaky, 5th edition, McGraw Hill

Activities:

Outcome: Interpret register-level operations and perform arithmetic, logic, and shift micro-operations within a basic computer framework.

Activity: Use a simulation tool (e.g., Logisim or Digital Works) to design a simple 4-bit ALU that performs:

- Addition, subtraction
- AND, OR
- Logical and arithmetic shifts

Evaluation Method: Demonstration-based assessment where students explain each operation using test inputs. Include a short worksheet with expected vs. actual outputs.

Outcome: Illustrate CPU functionality, addressing modes, and control unit design with both hardware-based and microprogramming techniques.

Activity: Create a flowchart or block diagram showing:

- CPU components (ALU, registers, control unit)
- Addressing modes (immediate, direct, indirect, etc.)
- Control unit types (hardwired vs. microprogrammed)

Evaluation Method:
Oral presentation or peer-reviewed poster session. Use a rubric to assess clarity, completeness, and correct identification of modes and control logic on 1 10-point scale.

Outcome: Categorize types of memory and explain memory hierarchy, access methods, and mapping techniques.

Activity: Build a memory hierarchy chart using coloured cards or digital tools. Include:

- Registers, cache, RAM, secondary storage
- Access methods (direct, associative, etc.)
- Mapping techniques (direct, associative, set-associative)

Evaluation Method: Quiz with matching and short-answer questions. Include a scenario-based question where students choose the best memory type for a given task.

Outcome: Describe I/O organization, including asynchronous transfer modes, DMA, and interrupt-driven processing.

Activity: Role-play simulation: Assign students' roles (CPU, DMA controller, I/O device) and simulate data transfer using cards or tokens to represent data and control signals.

Evaluation Method: Reflective worksheet where students describe the flow of control and data in each mode. Include a diagram labelling key signals and steps.

Outcome: Apply arithmetic algorithms and demonstrate floating point arithmetic operations.

Activity: Use a spreadsheet or visual tool to simulate floating point arithmetic operations.

Evaluation Method: Submission of simulation results with explanation. Evaluate the report on a 10-point scale.

SEMESTER-III

COURSE 6: COMPUTER ORGANIZATION

Practical	Credits: 1	2 hrs/week

List of Experiments:

1. Simulate Register Transfer Using Logisim
2. Build a microprogrammed control unit using Logisim or VHDL (GHDL).
3. Simple Register Transfers and Arithmetic Operations using EMU8086 Simulator
4. Simulate a Simple Instruction Cycle in Emu8086
5. Demonstrate Addressing Modes (Immediate, Register, Direct, Indexed) in Emu8086
6. Implement Subroutines Using CALL and RET in Emu8086
7. Simulate DMA Transfer in Ripes or Logisim
8. Simulate Booth's Algorithm for multiplication and restoring division in Python/C or Logisim
9. Write assembly language code for A+B*(C-D) using various instruction formats in any open-source assembler.
10. Write assembly language code for A+B*C using various addressing modes in any open-source assembler.

SEMESTER-III

COURSE 7: PROBABILITY AND STATISTICS

Theory	Credits: 4	4 hrs/week

Course Objectives:

1. To introduce the foundational concepts of statistics including data types, graphical representation, and basic statistical measures.
2. To develop the ability to analyze relationships between variables using correlation, regression, and probability theory.
3. To enable students to apply probability distributions and hypothesis testing for decision-making and statistical inference.

Course Outcomes (COs):

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

1. Organize and summarize statistical data using frequency tables, charts, and descriptive measures.
2. Calculate and interpret measures of central tendency and dispersion for given data.
3. Apply correlation and linear regression techniques to examine relationships between variables.
4. Solve real-life problems using probability rules and standard probability distributions.
5. Conduct hypothesis testing using z-test, t-test, and chi-square test for statistical inference.

Unit 1. Basic concepts of Statistics:

Qualitative and quantitative data, classification of data, construction of frequency distribution, diagrammatic representation of data.

Measures of Central Tendency: Arithmetic mean, median and mode-their properties **Measures of Dispersion**: Range, mean deviation, quartile deviation, variance and standard deviation.

Unit 2. Correlation & Regression:

Correlation: Definition, scatter diagram, types of correlation, measures—Karl Pearson's correlation coefficient and Spearman's rank correlation coefficient.

Regression: Linear regression-fitting by least square method and interpretation.

Unit 3. Probability & Random Variables:

Concepts of probability: Experiment and sample space, events and operations with events, probability of an event, basic probability rules, applications of probability rules, conditional probability.

Random Variables: Discrete and continuous random variable, probability distribution of a random variable, probability mass function, probability density function, expectation and variance of a random variable.

Unit 4. Probability Distributions:

Standard Probability Distributions: Binomial probability distribution, Poisson probability distribution, Normal probability distribution.

Sampling Distribution: Concept of Population and Sample, parameter and statistic, sampling distribution of sample mean and sample proportion.

Unit 5. Statistical Inference Techniques:

Statistical Inference: Estimation and Hypothesis Testing (only concept).

Hypothesis Testing for a Single Population: Concept of a hypothesis testing, tests involving a population mean and population proportion (z test and t test). **Chi square test for independence of attributes and goodness of fit.**

Text Books

1. Manish Sharma, Amit Gupta, The Practice of Business Statistics, Khanna Book Publishing Company, 2010 (AICTE Recommended Textbook)
2. Das N. G., Statistical Methods, Combined Edition, Tata McGraw Hill, 2010.
3. Ross Sheldon M., Introduction to Probability and Statistics for Engineers and Scientists, 6th Edition, Elsevier, 2021.
4. Miller Irwin and Miller Marylees, Mathematical Statistics with Applications, Seventh Edition, Pearson Education, 2005

Reference Books

5. Pal Nabendu and Sarkar Sahadeb, Statistics: Concepts and Applications, Second Edition, PHI, 2013
6. Montgomery Douglas and Runger George C., Applied Statistics and Probability for Engineers, Wiley, 2016.
7. Reena Garg, Engineering Mathematics, Khanna Publishing House, 2024.

Web Resources

1. https://nptel.ac.in/courses/111106112

2. https://nptel.ac.in/courses/111105041

Outcome: Organize and summarize statistical data using frequency tables, charts, and descriptive measures.

Activity: Students will collect real or simulated data (such as student marks or preferences), organize it into a frequency table, and represent it visually using bar charts, pie charts, or histograms.
Evaluation Method: The teacher will evaluate the activity based on accuracy of data classification, correctness of charts, and clarity of presentation.

Outcome: Calculate and interpret measures of central tendency and dispersion for given data.
Classroom Activity: Students will be given a worksheet with datasets in class and asked to compute the mean, median, mode, range, variance, and standard deviation.
Evaluation Method: The worksheet will be reviewed for correctness of formulas used, accuracy of calculations, and proper interpretation of the results.

Outcome: Apply correlation and linear regression techniques to examine relationships between variables.
Classroom Activity: Working in pairs, students will gather small bivariate datasets (e.g., height and weight of classmates), plot scatter diagrams, calculate the correlation coefficient, and derive the regression equation.
Evaluation Method: The teacher will assess the submission based on accuracy of plotting, correct use of formulas, and the interpretation of results. Peer collaboration will also be observed.

Outcome: Solve real-life problems using probability rules and standard probability distributions.
Classroom Activity: Students will conduct simple random experiments in class—like rolling dice, flipping coins, or drawing colored balls from a bag—to understand probability. They will then apply theoretical probability concepts to solve related questions.
Evaluation Method: Students will be assessed based on their understanding and application of probability rules, logical reasoning, and active participation in the group task.

Outcome: Conduct hypothesis testing using z-test, t-test, and chi-square test for statistical inference.
Classroom Activity: Students will be given a case scenario or dataset (e.g., a small sample of test scores) and asked to choose the correct hypothesis test, carry it out, and interpret the result. This will be followed by a short classroom discussion or group explanation.
Evaluation Method: Students will be evaluated based on the accuracy of hypothesis formulation, selection of the appropriate test, computation steps, and clarity in explaining their conclusions.

SEMESTER-IV

COURSE 8: OPERATING SYSTEMS

Theory	Credits: 3	3 hrs/week

Course Objectives:

1. **Understand the evolution and core functions** of operating systems, including resource management and system types.
2. **Analyze process and thread management**, focusing on system calls, kernel modes, scheduling algorithms, and threading models.
3. **Evaluate process synchronization and deadlock handling**, using classical problems and inter-process communication methods.
4. **Apply memory management techniques**, including paging, segmentation, and virtual memory implementation.
5. **Examine file, I/O, and device management strategies**, along with basic OS-level security features.

Course Outcomes:

At the End of the Course, The Students will be able to:

1. Explain the foundational principles and evolution of operating systems, including resource abstraction and core management functions.
2. Classify and compare different types of operating systems, such as multiprogramming, batch, time-sharing, real-time, and personal device-based systems.
3. Analyze process and thread management techniques, including processor modes, system calls, kernel functions, and scheduling algorithms.
4. Evaluate process synchronization and deadlock handling approaches, applying classical concurrency solutions and inter-process communication methods.
5. Apply memory, file, and I/O management strategies, incorporating allocation techniques, virtual memory models, disk scheduling, and OS-level security features.

Unit 1. Operating System Fundamentals:

Operating System Definition, History and Evolution of OS, Basic OS functions, Types of Operating Systems– Multiprogramming Systems, Batch Systems, Time Sharing Systems; Operating Systems for Personal Computers, Workstations and Hand-held Devices, Process Control & Real time Systems.

Unit 2. Process & Thread:

Processor and User Modes, Kernels, System Calls and System Programs, System View of the Process and Resources, Process Abstraction, Process Hierarchy, Threads, Threading Issues, Thread Libraries; Process Scheduling- Non-Preemptive and Preemptive Scheduling Algorithms.

Unit 3. Process Management:
Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Inter process Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

Deadlock, Deadlock Characterization, Necessary and Sufficient Conditions for Deadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery.

Unit 4. Memory Management:
Physical and Virtual Address Space; Memory Allocation Strategies–Fixed and -Variable Partitions, Paging, Segmentation, Virtual Memory.

Unit 5. File and I/O Management:
Directory Structure, File Operations, File Allocation Methods, Device Management, Pipes, Buffer, Shared Memory, Disk Scheduling algorithms.

Text Books:

1. Operating System Principles, Abraham Silberschatz, Peter Baer Galvin and GregGagne, 7th Edition, Wiley India Edition.
2. Operating Systems: Internals and Design Principles, Stallings , Pearson edition

Reference Books:

1. Operating Systems Design and Implementation, Andrew S. Tanenbaum, 3rd Edition, Pearson
2. A Text Book of Operating Systems, Singh, Kaur and Gupta, Khanna Publishers

Activities:

Outcome: Explain Foundational Principles and Evolution of Operating Systems

Activity: Timeline Creation & Concept Mapping - Students will collaboratively build a visual timeline showing the evolution of operating systems from early batch systems to modern distributed and mobile OS. They will also create a concept map illustrating core management functions (e.g., process, memory, file, device management).

Evaluation Method: Rubric-based assessment of timeline and concept map:

- Historical accuracy
- Inclusion of core OS functions
- Presentation and teamwork

Outcome: Classify and Compare Types of Operating Systems

Activity: Comparative Analysis Table & Case Study Discussion - Students will research and fill out a comparison table for OS types (multiprogramming, batch, time-sharing, real-time, personal device-based), focusing on architecture, scheduling, responsiveness, and use cases. Followed by a group discussion using real-world examples (e.g., RTOS in pacemakers vs. Android in smartphones).

Evaluation Method: Students will be evaluated on a 10-point scale based on

- Individual submission of comparison table
- Group participation score in discussion

Outcome: Analyze Process and Thread Management Techniques

Activity: Simulation & Code Walkthrough - Students will simulate process scheduling using tools or pseudocode (e.g., Round Robin, Priority Scheduling). They will also analyze thread creation and system calls using a simple multithreaded program in Java.

Evaluation Method: Students will be assessed on a 10-point scale based on

- Scheduling algorithm simulation results
- Explanation of processor modes and system calls
- Code annotations for kernel functions

Outcome: Evaluate Process Synchronization and Deadlock Handling

Activity: Role-play & Coding Challenge - Students will role-play classical problems (e.g., Dining Philosophers, Producer-Consumer) to understand synchronization. Then, they'll implement semaphore or monitor-based solutions in code and simulate deadlock detection or avoidance.

Evaluation Method: Students will be evaluated for 10 marks based on

- Code review for correctness and use of synchronization primitives
- Peer evaluation of role-play clarity and engagement
- Written test with deadlock scenarios and solution strategies

Outcome: Apply Memory, File, and I/O Management Strategies

Activity: Interactive Lab & Design Task - Students will use OS simulators to experiment with memory allocation (paging, segmentation), disk scheduling (FCFS, SSTF, SCAN), and file system operations.

Evaluation Method: Students will be evaluated for 10 points based on

- Correct use of allocation techniques
- Disk scheduling output analysis
- Virtual memory behaviour observation

SEMESTER-IV

COURSE 8: OPERATING SYSTEMS

Practical | **Credits: 1** | **2 hrs/week**

List of Experiments:

1. Illustrate the LINUX commands
    a) pwd
    b) mkdir
    c) rmdir
    d) grep
    e) chmod
    f) ls
    g) rm
    h) cp

2. Write a program to calculate average waiting time and turn around time of each process using the following CPU Scheduling algorithm for the given process schedules.
    a) FCFS
    b) SJF
    c) Priority
    d) Round Robin

3. Simulate MVT and MFT memory management technique

4. Write a program for Bankers Algorithm for Dead Lock Avoidance

5. Implement Bankers Algorithm Dead Lock Prevention.

6. Write a program to simulate Producer-Consumer problem.

7. Simulate all Page replacement algorithms.
    a) FIFO
    b) LRU
    c) LFU
    d) Optimal

8. Simulate Paging Techniques of memory management

9. Simulate the following disk scheduling algorithms
    a) FCFS
    b) SSTF
    c) SCAN
    d) CSCAN

SEMESTER-IV

COURSE 9: PYTHON PROGRAMMING

Theory	Credits: 3	3 hrs/week

Course Objectives

1. **Introduce the foundational concepts** of Python programming including its syntax, IDEs, and control structures.
2. **Develop proficiency in modular programming** using functions, lambda expressions, recursion, and Python's built-in modules and packages.
3. **Explore core data structures** like strings, lists, tuples, and dictionaries for effective data manipulation.
4. **Teach exception handling mechanisms** and the use of regular expressions for pattern matching and text processing.
5. **Enable students to interact with files and databases** using Python to build real-world applications involving persistent storage and data retrieval.

Course Outcomes

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

1. **Write and execute structured Python programs** using variables, expressions, and flow control statements.
2. **Implement modular code** leveraging functions, argument types, recursion, and reusable libraries.
3. **Manipulate and organize data efficiently** using Python's string operations and complex data structures.
4. **Handle runtime errors and apply regular expressions** for robust and flexible program behaviour.
5. **Perform file operations and connect to databases** through Python scripts to store, retrieve, and manage data effectively.

Unit 1. Basics of Python Programming:

Features of python, history of python, Python IDEs, Writing and Executing Python Program, literal constants, variables and identifiers, Data types, input operation- comments, Reserved words, Indentation, Operators and Expressions: Expressions in python, Operations in Strings, Other Data types, Type conversion, Decision control Statements, Iterative Statements, Nested loops, break, Continue, Pass Statements, else statement used with loops.

Unit 2. Strings and Collections:

Strings: Built-in String Methods and Functions

Lists: Access values in List, Updating values in Lists, Nested lists, Basic list operations, List Methods.

Tuple: Creating, Accessing, Updating and Deleting Elements in a tuple,Nested tuples.

Dictionaries: Creating a dictionary, Accessing values, Modifying an Entry, Deleting items, Built-in Dictionary Functions and Methods

Unit 3. Functions and Modules:
Function Definition, Function Call, Variable Scope and lifetime, The return statement, Required Arguments, Keyword Arguments, Default Arguments and Variable Length Arguments, Lambda Functions, Recursive Functions.

Importing Libraries, Modules, Packages in python, Standard library modules- Globals(), Locals(), and Reload(), Function Redefinition.

Unit 4. Exception Handling:
Errors and Exceptions, Handling Exceptions, Multiple Except blocks, Multiple Exceptions in a single block, Except Block without Exception, The else clause, Built-in and user-defined Exceptions, The finally block, Re-raising Exception, Assertions in python

Unit 5. File Handling & Database Connectivity:
Types of files in Python, Opening and Closing files, Reading and Writing files: write() and writelines() methods- append() method, read() and readlines() methods, Splitting words, File Positions.

Database connectivity using Python, Executing SQL commands using python, Assimilating SQL command results using python.

Textbooks:

1. Python Programming using Problem Solving Approach Reema Thareja Oxford University Press 2020
2. Exploring Python, Budd T A, McGraw-Hill Education, 1st Edition, 2011.

Reference Book:

1. Python: The Complete Reference, Martin C. Brown, Mc Graw Hill, 2018
2. Fundamentals of Python, Kenneth A. Lambert. (2019), First Programs,2nd Edition, CENGAGE Publication.

Activities:

Outcome: Write and execute structured Python programs using variables, expressions, and flow control statements.

Activity: Create a calculator program that uses variables, arithmetic expressions, and flow control (if-else) to perform basic operations (add, subtract, multiply, divide) based on user input.

Evaluation Method: Code walkthrough and output validation. Use a checklist to assess on a 10-point scale to check the:

- Correct use of variables and expressions
- Proper flow control logic
- Accurate results for different inputs

Outcome: Implement modular code leveraging functions, argument types, recursion, and reusable libraries.

Activity: Build a factorial calculator using both iterative and recursive functions. Include parameterized functions and import the math library for comparison.

Evaluation Method: Code review and oral explanation. Assess on 10-point scale based on:

- Function structure and argument usage
- Recursion logic
- Use of reusable libraries

Outcome: Manipulate and organize data efficiently using Python's string operations and complex data structures.

Activity: Develop a contact manager that stores names and phone numbers using dictionaries and lists. Include string formatting and search functionality.

Evaluation Method: Practical demo with test cases. Evaluate based on:

- Use of string methods (split, join, format)
- Data structure selection and manipulation
- Search and retrieval accuracy

Outcome: Handle runtime errors and apply regular expressions for robust and flexible program behaviour.

Activity: Create a form validator that checks email and phone number formats using regular expressions. Include try-except blocks to handle invalid inputs.

Evaluation Method: Scenario-based testing. Assess based on:

- Regex accuracy for pattern matching
- Robust error handling
- Program stability with edge cases

Outcome: Perform file operations and connect to databases through Python scripts to store, retrieve, and manage data effectively.

Activity: Build a student grade logger that reads from a CSV file, stores data in a SQLite database, and allows querying by student name.

Evaluation Method: Lab test with sample data. Evaluate on a 10-point scale:

- File read/write operations
- Database connection and query execution
- Data integrity and retrieval accuracy

SEMESTER-IV

COURSE 9: PYTHON PROGRAMMING

Practical | **Credits: 1** | **2 hrs/week**

List of Experiments

1. Display a welcome message using print().
2. Declare and use identifiers belonging to strings, integers, floats, and booleans.
3. Accept user input (name, age, height, student status) and display each value with its type using type().
4. Perform operations like .upper(), .find(), .replace() on strings.
5. Write a program to reverse the string, count vowels and words.
6. Write a program for slicing, sorting, and list comprehension.
7. Program to apply list methods: append(), extend(), insert(), remove(), pop(), sort().
8. Create tuples to store student (name, age, course) data and perform
    a. Accessing elements using indexing and slicing.
    b. Demonstrate immutability by attempting to modify a tuple.
    c. Create and navigate nested tuples.
9. Create a dictionary with student roll number as keys and names/marks as values.
    a. Accessing, adding, updating, and deleting key-value pairs.
    b. Iterating through keys, values, and items.
10. Write a program to demonstrate variable length arguments.
11. Write a program to illustrate lambda and recursive functions.
12. Write a program to demonstrate globals(), locals(), and reload() functions.
13. Demonstrate exception handling and assertions in Python.
14. Write a Python program to copy the contents of one file into another in reverse order.
15. Working with Datasets using Numpy and Pandas
16. Write a program to connect to the database and retrieve the required information using SQL commands.

SEMESTER-IV

COURSE 10: SOFTWARE ENGINEERING

Theory	Credits: 3	3 hrs/week

Course Objectives :

1. Understand the fundamental principles of software engineering, including software development life cycle models and their practical applications.
2. Analyze and document software requirements through feasibility studies and specification techniques.
3. Apply software design principles and development standards for building reliable and maintainable systems.
4. Evaluate software testing methodologies, including design and execution of test cases for various testing levels.
5. Examine software maintenance strategies, types, and metrics to sustain software performance over time.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Compare and contrast software development life cycle models such as Waterfall, Spiral, and Agile, and explain their appropriate use cases.
2. Conduct requirement analysis and distinguish between functional and non-functional requirements to develop a Software Requirements Specification (SRS) document.
3. Utilize design principles like modularity, cohesion, and coupling; implement Data Flow Diagrams (DFDs), structure charts, and follow coding standards and version control practices.
4. Design and perform different types of software tests—white-box, black-box, unit, integration, system—and differentiate between manual and automated testing approaches.
5. Categorize software maintenance types and propose strategies based on software maintenance metrics for effective long-term software sustainability.

Unit 1: Software Engineering Foundations and Requirements Engineering:
Definition of Software engineering, Software life cycle models: Waterfall, prototyping, Evolutionary, Spiral and Agile models. Comparison among development life cycles.

Unit 2: Requirement Analysis and Specification
Feasibility study, Requirements gathering, Functional and Non-functional requirements, Requirements analysis and specification, design of software requirement specification (SRS).

Unit 3: Software Design Principles and Development Practices
Introduction to software design, modularity, cohesion, coupling and layering, functional design, and solution design, use of DFD and Structure chart in software design, UML in software design, user interface design. Software Development Basics, Coding standards, Version Control and Code review techniques.

Unit 4: Software Testing
Fundamentals of testing (verification and validation), White-box, and black-box testing, unit testing, integration testing, system testing, acceptance testing (alpha testing and beta testing), test scenarios and test case design, automation testing and manual testing.

Unit 5: Software Maintenance
Introduction to software maintenance, corrective maintenance, perfective maintenance, adaptive maintenance, preventive maintenance, challenges in software maintenance, metrics related to software maintenance.

Textbooks:

1. Software Engineering: A Practitioner's Approach, Roger Pressman, McGraw Hill, 6th Edition
2. Software Engineering, Sommerville, Adison Wesley, 7th edition.

Reference Books:

1. Fundamentals of Software Engineering, Mall Rajib,PHI, Fifth Edition,
2. Fundamentals of Software Engineering, Hitesh, BPB Publications

Activities:

Outcome: Compare and contrast software development life cycle models such as Waterfall, Spiral, and Agile, and explain their appropriate use cases.

Activity: Create a comparison chart in groups showing key features, pros/cons, and use cases of Waterfall, Spiral, and Agile models. Include a real-world example for each.

Evaluation Method: Use a rubric to assess on a 10-point scale to check:

- Accuracy of model descriptions
- Clarity of comparison
- Relevance of examples
- Presentation skills (if shared in class)

Outcome: Conduct requirement analysis and distinguish between functional and non-functional requirements to develop a Software Requirements Specification (SRS) document.

Activity: Analyze a simple system (e.g., online library or food ordering app). Identify 5 functional and 3 non-functional requirements. Draft a mini-SRS document using a template.

Evaluation Method: Checklist-based review on a 10-point scale:

- Correct classification of requirements
- Completeness of SRS sections
- Clarity and formatting
- Use of standard terminology

Outcome: Utilize design principles like modularity, cohesion, and coupling; implement Data Flow Diagrams (DFDs), structure charts, and follow coding standards and version control practices.

Activity: Design a basic login system using DFD (Level 0 and Level 1) and a structure chart. Highlight modules and discuss cohesion and coupling in pairs.

Evaluation Method: Peer review and instructor feedback on a 10-point scale:

- Correct use of DFD symbols
- Logical flow and modularity
- Explanation of cohesion/coupling
- Neatness and labelling

Outcome: Design and perform different types of software tests—white-box, black-box, unit, integration, system—and differentiate between manual and automated testing approaches.

Activity: Write simple test cases for a calculator app (e.g., addition, division by zero). Perform manual unit testing and simulate black-box and white-box testing.

Evaluation Method: Observation and worksheet to assess on a 10-point scale:

- Correct identification of test types
- Clear test case steps and expected output
- Execution and result recording
- Understanding of manual vs automated testing

Outcome: Categorize software maintenance types and propose strategies based on software maintenance metrics for effective long-term software sustainability.

Activity: Categorize sample maintenance tasks (e.g., fixing a bug, adding a feature) into corrective, adaptive, perfective, or preventive. Discuss sustainability strategies in small groups.

Evaluation Method: Group activity score:

- Correct classification of maintenance types
- Participation in discussion
- Suggestions for sustainability (e.g., documentation, modular design)
- Use of basic metrics (e.g., number of bugs fixed)

SEMESTER-IV

COURSE 10: SOFTWARE ENGINEERING

Practical | **Credits: 1** | **2 hrs/week**

Select domain of interest (e.g. College Management System) and identify multi-tier software applications to work on (e.g. Online Fee Collection). Analyze, design and develop this application:

1. Develop an SRS document. Also develop risk management and project plan (Gantt chart).
2. Understanding of System modeling: Data modeling using ER – Diagram-Draw an ER Diagram which also includes generalization, specialization and aggregation of specified problem statements.
3. Understanding of System modeling: Functional modeling: DFD Context and draw it
4. Understanding of System modeling: UML and draw it.
5. Develop a sample calculator program and perform Black-box Testing:
    a. Identify test scenarios without viewing the internal code.
    b. Write test cases for valid and invalid inputs.
    c. Execute the test cases and record outcomes.
6. Develop a sample login authentication page and perform White--box Testing:
    a. Analyze the code for all possible paths, conditions, and loops.
    b. Apply statement coverage and branch coverage techniques.
    c. Test internal functions with known inputs.
7. For the above login authentication page, perform the following:
    a. Simulate the following maintenance activities:
        - **Corrective Maintenance**: Fix an existing bug (e.g., wrong output, crash, miscalculation).
        - **Perfective Maintenance**: Add a new user-requested feature (e.g., sorting, filter, or improved UI).
        - **Adaptive Maintenance**: Modify the code to adapt to a new platform or library version.
        - **Preventive Maintenance**: Refactor the code to improve readability, performance, or security.
    b. Document each change with:
        - Problem description
        - Type of maintenance
        - Before and after screenshots
        - Change summary

SEMESTER-V

COURSE 11: DATA MINING

Theory	Credits: 3	3 hrs/week

Course Objectives:

- Understand the concepts, architecture, and applications of Data Warehousing and its role in business intelligence.
- Gain knowledge of data preprocessing techniques used in preparing data for analysis.
- Learn various data mining techniques including association rule mining, clustering, and classification using decision trees.
- Explore advanced algorithms for pattern discovery and knowledge extraction.
- Apply web mining and text mining techniques for extracting meaningful patterns from unstructured data sources.

Course Outcomes:

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

1. Understand the architecture and components of Data Warehousing, including multidimensional data models, OLAP operations, and data preprocessing techniques for effective data storage and analysis.
2. Demonstrate knowledge of core data mining concepts, processes, and techniques, and distinguish them from traditional database systems.
3. Apply various association rule mining algorithms (such as Apriori, FP-Growth, and others) to discover hidden patterns and relationships in large datasets.
4. Implement and evaluate clustering and classification techniques (like k-Medoid, DBSCAN, Decision Trees) for data grouping and predictive modelling.
5. Analyze and apply web and text mining methods to extract meaningful insights from semi-structured and unstructured data sources for real-world applications.

Unit-1: Data Warehousing:

Introduction, What is Data Warehouse? Definition, Multidimensional Data Model, OLAP Operations, Warehouse Schema, Data Warehouse Architecture, Warehouse Server, Metadata, OLAP Engine, Data Warehouse Backend Process, Other Features Data Preprocessing, Descriptive Data Summarization, Data Cleaning, Data Integration and Transformation, Data Reduction, Data Discretization and Concept Hierarchy Generation