ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

Model Syllabus for 4-Year UG Honours in B.Sc. (Computer Science) 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	Digital Logic Design	3	3
			Digital Logic Design -Practical	2	1
II	III	5	OOPS Through JAVA	3	3
			OOPS Through JAVA-Practical	2	1
		6	Data Base Management	3	3
			Data Base Management -Practical	2	1
		7	Computer Organisation	3	3
			Computer Organisation-Practical	2	1
	IV	8	Operating Systems	3	3
			Operating Systems-Practical	2	1
		9	Computer Networks	3	3
			Computer Networks	2	1
		10	Python Programming	3	3
			Python Programming-Practical	2	1

Signatures of BoS Members

Year	Semester	Course	Title of the Course	No. of Hrs /Week	No. of Credits
III	V	11	Software Engineering	3	3
			Software Engineering-Practical	2	1
		12 A	Web Interface Design Technologies	3	3
			Web Interface Design Technologies-Practical	2	1
			OR		
		12 B	SEB1:Data Science with R	3	3
			SEB1:Data Science with R-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	Python for Data Science	3	3
			Python for Data Science Practical	2	1
	VI	14 A	Mobile Application Development	3	3
			Mobile Application Development-Practical	2	1
			OR		
		14 B	Data Visualization Tools	3	3
			Data Visualization Tools-Practical	2	1
		15 A	MERN Stack	3	3
			MERN Stack-Practical	2	1
			OR		
		15 B	Machine Learning	3	3
			Machine Learning-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-I

COURSE 1: COMPUTER FUNDAMENTALS AND OFFICE AUTOMATION

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives

1. **Understand foundational computing concepts**, including number systems, the evolution of computers, block diagrams, and generational progress.
2. **Develop knowledge of computer architecture**, focusing on system organization and networking fundamentals.
3. **Acquire practical skills in document creation**, formatting, and digital presentations using word processing tools.
4. **Gain proficiency in spreadsheet operations**, such as data entry, formulas, functions, and charting techniques.
5. **Introduce data visualization and basic modelling principles**, fostering analytical thinking in structuring and interpreting data sets.

Course Outcomes

1. At the End of the Course, The Students will be able to **explain different number systems**, the historical evolution of computers, and identify key components in a block diagram.
2. Learners will demonstrate **basic blocks of a computer and fundamental networking knowledge.**
3. Learners will create professional-level documents and **design visually appealing presentations** using word processing software and presentation software.
4. Learners will manipulate data within spreadsheets, apply formulas, and **generate accurate summaries and visualizations.**
5. Learners will apply data modelling techniques to **analyze, organize, and represent data effectively** in various scenarios.

Unit 1. Number Systems, Evolution, Block Diagram and Generations:

Change the order accordig to Nicembers

(4) **Number Systems**: Binary, Decimal, Octal, Hexadecimal; conversions between number systems.

(1) **Evolution of Computers**: History from early mechanical devices to modern-day systems.

(3) **Block Diagram of a Computer**: Components like Input Unit, Output Unit, Memory, CPU (ALU + CU).

(2) **Generations of Computers**: First to Fifth Generation – technologies, characteristics, examples.

Unit 2. Basic organization and N/W fundamentals:

Computer Organization: Functional components – Input/Output devices, Storage types, Memory hierarchy.

Types of Computers: Micro, Mini, Mainframe, and Supercomputers.

Networking Fundamentals: Definition, need for networks, types (LAN, WAN, MAN), topology (Star, Ring, Bus).

Internet Basics: IP Address, Domain Name, Web Browser, Email, WWW.

Unit 3. Word Processing and presentations:
Word Processing Basics: Using MS Word/Google Docs – formatting, styles, tables, mail merge.

Presentation Tools: Using PowerPoint/Google Slides – slide design, animations, transitions.

Applications: Creating resumes, reports, brochures, and presentations.

Keyboard Shortcuts

Unit 4. Spreadsheet Basics:
Spreadsheet Concepts: Understanding rows, columns, cells in tools like MS Excel/Google Sheets, cell referencing.

Functions and Formulae: SUM, AVERAGE, IF, COUNT.

Charts and Graphs: Creating visual representations

Data Handling: Sorting, filtering, conditional formatting.

Text Functions: LEFT, RIGHT, MID, LEN, TRIM, CONCAT, TEXTJOIN

Advanced Functions: **Logical**: IF, AND, OR, IFERROR, **Lookup**: VLOOKUP, HLOOKUP, XLOOKUP, INDEX, MATCH

Unit 5. Data Analysis and Visualization:
Conditional Formatting: Custom rules, Color scales, Icon sets, Data bars

Data Analysis Tools: Pivot Tables and Pivot Charts, Data Validation (Drop-downs, Input Messages, Error Alerts), What-If Analysis: Goal Seek, Scenario Manager, Data Tables

Charts and Dashboards: Creating Interactive Dashboards, Using slicers with Pivot Tables,Combo Charts and Sparklines

Remove

~~**Productivity Tips:** Using Named Ranges, Freeze Panes, Split View~~

Textbooks:

1. **Fundamentals of Computers,** Reema Thareja, Oxford University Press, Second Edition
2. **Fundamentals of Computers**, V. Rajaraman – PHI Learning
3. **Introduction to Computers** by Peter Norton – McGraw Hill
4. **Microsoft Office 365 In Practice** by Randy Nordell – McGraw Hill Education

References:

1. **Excel 2021 Bible** by Michael Alexander, Richard Kusleika – Wiley
2. **Networking All-in-One For Dummies** by Doug Lowe – Wiley
3. **Microsoft Official Docs and Training:** https://learn.microsoft.com
4. **Google Workspace Learning Center**: https://support.google.com/a/users/

Activities:

Outcome: At the End of the Course, The Students will be able to **explain different number systems**, the historical evolution of computers, and identify key components in a block diagram.

Activity: Create a digital poster or infographic comparing number systems (binary, decimal, octal, hexadecimal) and illustrating the timeline of computer generations with key innovations.

Evaluation Method: Rubric-based assessment of the poster presentation on a 10-point scale focusing on:

- Accuracy of number system conversions
- Correct identification of block diagram components
- Visual organization and creativity

Outcome: Learners will demonstrate **basic blocks of a computer and fundamental networking knowledge.**

Activity: Design a concept map showing the internal architecture of a computer and types of networks (LAN, WAN, MAN), including devices and topologies.

Evaluation Method: Checklist-based peer review and instructor validation:

- Completeness of the map
- Correctness of networking concepts
- Use of appropriate terminology
- Logical flow and structure of the map

Outcome: Learners will create professional-level documents and **design visually appealing presentations** using word processing software and presentation software.

Activity: Prepare a formal report (e.g., project proposal) in a word processor and present it using a slide deck with transitions, embedded media, and design elements.

Evaluation Method: Performance-based evaluation using a 10-point scoring scale:

- Formatting and structure of the document
- Presentation aesthetics and clarity
- Communication skills during presentation

Outcome: Learners will manipulate data within spreadsheets, apply formulas, and **generate accurate summaries and visualizations.**

Activity: Analyze a dataset (e.g., student scores or sales data) using spreadsheet software. Apply formulas (SUM, AVERAGE, IF, VLOOKUP) and create relevant charts.

Evaluation Method: Practical test with a rubric:

- Correct use of formulas
- Accuracy of data summaries

Outcome: Learners will apply data modelling techniques to **analyze, organize, and represent data effectively** in various scenarios.

Activity: Prepare an interactive dashboard for a given data set using EXCEL.

Evaluation Method: Evaluation of the dashboard on a 10-point scoring scale:

- Presentation aesthetics and clarity
- Interactiveness
- Communication skills during presentation

SEMESTER-I

COURSE 1: COMPUTER FUNDAMENTALS AND OFFICE AUTOMATION

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

List of Experiments:

1. Demonstration of Assembling and Dessembling of Computer Systems.
2. Identify and prepare notes on the type of Network topology of your institution.
3. Prepare your resume in Word.
4. Using Word, write a letter to your higher official seeking 10-days leave.
5. Prepare a presentation that contains text, audio and video.
6. Using a spreadsheet, prepare your class Time Table.
7. Using a Spreadsheet, calculate the Gross and Net salary of employees(Min 5) considering all the allowances.
8. Generate the class-wise and subject-wise results for a class of 20 students. Also generate the highest and lowest marks in each subject.
9. Using IF, AND, OR, and IFERROR to Automate Grade Evaluation.
    a. Create a table of student scores in different subjects.
    b. Use IF to assign grades (A/B/C/Fail).
    c. Use IFERROR to handle missing scores or invalid data.
10. *Employee Database Search Using VLOOKUP, HLOOKUP, XLOOKUP, INDEX, and MATCH*
    a. Create a database of employees (Name, ID, Department, Salary).
    b. Implement VLOOKUP to search by employee ID.
    c. Use HLOOKUP to extract department heads by role.
    d. Apply XLOOKUP for more flexible searches.
    e. Use INDEX + MATCH as an alternative to VLOOKUP.
11. Sales Report Analysis Using Pivot Tables and Charts
    a. Use a dataset of product sales (Product, Region, Date, Quantity, Revenue).
    b. Create Pivot Tables to summarize data by region/product.
    c. Insert Pivot Charts for visual analysis (e.g., bar, line).
    d. Add slicers to make the dashboard interactive.
12. Designing a Data Entry Form with Drop-downs and Input Rules
    a. Create a student registration form.
    b. Add drop-down lists for course selection using Data Validation.
    c. Add input messages to guide users.
    d. Add error alerts for wrong entries.
13. Monthly Budget Planning using Goal Seek and Scenario Manager
    a. Create a simple personal budget (income, expenses, savings).
    b. Use Goal Seek to determine income needed to save a desired amount.
    c. Use Scenario Manager to compare different budgeting scenarios (best/ worst/ realistic case).

d. Create a one-variable Data Table to analyze how different expenses affect savings.

14. Dashboard Creation Using Combo Charts, Sparklines & Slicers
    a. Use existing sales or attendance data.
    b. Insert combo charts (e.g., column + line).
    c. Add sparklines to show trends.
    d. Use slicers with Pivot Tables to control dashboard elements.
    e. Finalize and format for interactivity.

SEMESTER-I

COURSE 2: PROBLEM SOLVING USING C

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. Understand the fundamentals of computer programming, Apply structured problem-solving approaches using algorithms, flowcharts, and C programming constructs.
2. Develop efficient logic using decision-making, loop, and jump control statements.
3. Utilize derived data types like arrays and strings for modular program design.
4. Design and implement modular solutions using functions, recursive logic, pointer operations, and dynamic memory management.
5. Handle complex data structures including structures, unions, and text file operations.

Course Outcomes:

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

1. Understand basic computing concepts, programming paradigms and write structured C programs.
2. Apply control flow statements to solve logical and repetitive tasks in C.
3. Implement arrays and string operations to manage and manipulate data efficiently.
4. Design modular code using functions, recursion, and appropriate parameter passing.
5. Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Unit 1. Introduction to computer programming:
Introduction, Types of software, Compiler and interpreter, Concepts of Machine level, Assembly level and high-level programming, Flowcharts and Algorithms, Fundamentals of C: History of C, Features of C, C Tokens-variables and keywords and identifiers, constants and Data types, Rules for constructing variable names, Operators, Structure of C program, Input /output statements in C-Formatted and Unformatted I/O

Unit 2. Control statements:
Decision making statements: if, if else, else if ladder, switch statements. Loop control statements: while loop, for loop and do-while loop. Jump Control statements: break,continue and goto.

Unit 3. Derived data types in C:
Arrays: One Dimensional arrays - Declaration, Initialization and Memory representation; Two Dimensional arrays -Declaration, Initialization and Memory representation. Strings: Declaring & Initializing string variables; String handling functions, Character handling functions

Pointers and Functions

Unit 4. ~~Functions:~~

Pointers: Pointer data type, Pointer declaration, initialization, accessing values using pointers. Pointer arithmetic, Pointers and arrays.

Function Prototype, definition and calling. Return statement. Nesting of functions. Categories of functions. Recursion (Basic Concept only). Parameter Passing by address & by value. Local and Global variables. Storage classes: automatic, external, static and register.

Unit 5. ~~Dynamic Memory Management:~~ Structures, Unions and File management

Introduction, Functions-malloc, calloc, realloc, free Structures: Basics of structure, structure members, accessing structure members, nested structures, array of structures, structure and functions, structures and pointers. Unions - Union definition; difference between Structures and Unions. Working with text files - modes: opening, reading, writing and closing text files.

Text Books:

1. Programming in ANSI C, E. Balagurusamy, Tata McGraw Hill, 6 th Edn,
2. Computer fundamentals and programming in C, Reema Theraja, Oxford University Press

Reference Books:

1. Let us C, Y Kanetkar, BPB publications
2. Head First C: A Brain-Friendly Guide, David Griffiths, Dawn Griffiths

Activities:

Outcome: Understand basic computing concepts, programming paradigms and write structured C programs.

Activity: Create a concept map of computing fundamentals and programming paradigms (procedural, structured, object-oriented). Then, they write a structured C program (e.g., a calculator or student grade system) using proper syntax, indentation, and modular design.

Evaluation Method: Rubric-based Code Review & Viva to check the

- The correctness of the concept map
- Correct use of structure (main + functions)
- Identification of paradigm used
- Code readability and documentation

Outcome: Apply control flow statements to solve logical and repetitive tasks in C.

Activity: Implement a program that solves a logic puzzle (e.g., number guessing game, pattern generation, or prime number finder) using if, switch, for, while, and do-while.

Evaluation Method: Automated Test Cases + Peer Review to check the

- Correct use of control statements
- Logical correctness of output

- Efficiency and edge case handling
- Peer feedback on clarity and logic

Outcome: Implement arrays and string operations to manage and manipulate data efficiently.

Activity: Build a program that stores and arranges student marks in ascending and descending order using arrays and performs string operations like concatenation, comparing, and formatting names.

Evaluation Method: Functional Demonstration + Code Walkthrough to check the

- Correct array and string usage
- Memory efficiency
- Handling of invalid inputs
- Explanation of sorting/searching logic

Activity:

- **Recursive Problem Solver**
  Students write a modular program to solve a recursive problem (e.g., factorial, Fibonacci, or Tower of Hanoi) using functions with parameters and return values.

Evaluation Method:

- **Code Trace + Written Quiz**
  - Correct function decomposition
  - Proper parameter passing (by value/reference)
  - Recursion depth and base case handling
  - Quiz on tracing recursive calls

Outcome: Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Activity: Create a program that dynamically stores user input (e.g., survey responses) using pointers and writes/reads the data to/from a text file.

Evaluation Method: Memory Debugging + File I/O Assessment to check the

- Proper use of malloc, calloc, realloc, and free
- Pointer arithmetic and dereferencing
- File creation, reading, writing, and error handling
- Use of tools like Valgrind or manual memory trace (Optional for Unix flavours)

SEMESTER-I

COURSE 2: PROBLEM SOLVING USING C

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

List of Experiments:

1. Write a program to check whether the given number is Armstrong or not.
2. Write a program to find the sum of individual digits of a positive integer.
3. Write a program to generate the first n terms of the Fibonacci sequence.
4. Write a program to find both the largest and smallest number in a list of integer values
5. Write a program to demonstrate change in parameter values while swapping two integer variables using Call by Value & Call by Address
6. Write a program to perform various string operations.
7. Write a program to search an element in a given list of values.
8. Write a program that uses functions to add two matrices.
9. Write a program to calculate factorial of given integer value using recursive functions
10. Write a program for multiplication of two N X N matrices.
11. Write a program to sort a given list of integers in ascending order.
12. Write a program to calculate the salaries of all employees using Employee (ID, Name, Designation, Basic Pay, DA, HRA, Gross Salary, Deduction, Net Salary) structure.
    a. DA is 30 % of Basic Pay
    b. HRA is 15% of Basic Pay
    c. Deduction is 10% of (Basic Pay + DA)
    d. Gross Salary = Basic Pay + DA+ HRA
    e. Net Salary = Gross Salary - Deduction
13. Write a program to read / write the data from / to a file.
14. Write a program to reverse the contents of a file and store in another file.
15. Write a program to create Book (ISBN,Title, Author, Price, Pages, Publisher)structure and store book details in a file and perform the following operations
    a. Add book details
    b. Search a book details for a given ISBN and display book details, if available
    c. Update a book details using ISBN
    d. Delete book details for a given ISBN and display list of remaining Books

SEMESTER-II

COURSE 3: DATA STRUCTURES USING C

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. Understand fundamental concepts of algorithms and data structures with focus on complexity analysis and abstract data types.
2. Explore various types of linked lists and their dynamic memory representations and operations.
3. Analyze and implement linear data structures, such as stacks and queues, and examine their real-world applications.
4. Apply sorting and searching algorithms, understanding their performance implications and optimization strategies.
5. Design and manipulate hierarchical and graph-based structures, applying traversal algorithms and understanding their practical uses in computing.

Course Outcomes:

Learners will be able to:

1. Explain algorithm characteristics, time and space complexity, and asymptotic notations with clarity.
2. Implement and analyze different types of linked lists, including insertion, deletion, and traversal operations.
3. Develop stack and queue data structures using arrays and linked lists, and apply them in expression evaluation.
4. Apply efficient searching and sorting algorithms to solve computational problems and evaluate performance trade-offs.
5. Construct and traverse tree and graph structures, using them to solve problems like shortest path and spanning trees.

Unit 1. Basic Concepts:

Algorithm: Definition and characteristics, Complexity analysis: Space Complexity, Time Complexity, Asymptotic Notations.

Introduction to Data structures: Definition, Types of Data structures, Abstract Data Types (ADT), Introduction to Linked Lists, Representation of linked lists in Memory, Comparison between Linked List and Array.

Unit 2. Linked Lists:

Types of Linked Lists - Singly Linked list, Doubly Linked list, Circularly Singly Linked list, Circularly Doubly Linked list; Implementation of Single Linked List ADT: Creating a List, Traversing a linked list, Searching in linked list, Insertion and deletion into linked list (At first Node, Specified Position, Last node).

Unit 3. Stacks and Queues:

Introduction to stack ADT, Implementation of stacks using array and Linked List, Application of stacks - Polish Notations - Converting Infix to Post Fix Notation - Evaluation of Post Fix Notation.

Queues: Introduction to Queue ADT, Implementation of Queues using array and Linked List, Application of Queues, Types of Queues- Circular Queues, De-queues, Priority Queue, Heaps.

Move to end of the unit

Unit 4. Searching and Sorting:

Linear or Sequential Search, Binary Search, Hashing and collision resolution.

Sorting: Selection Sort, Bubble Sort, Insertion Sort, Quick Sort and Merge Sort

Unit 5. Trees and Graphs:

Tree Terminology, Binary Tree Representation, Traversal techniques, Expression Tree, Binary Search Tree- Definition, Operations on a Binary Search Tree: Creation, Search, Insertion & deletion.

Graphs: Introduction to Graphs, Terminology, Representation (Adjacency Matrix, Adjacency List), Traversal of Graphs (DFS, BFS), Applications of Graphs, Concept of Shortest Path Problems, Concept of Minimum Cost Spanning Tree

Textbooks:

1. Data Structures Using C, Balagurusamy E. Tata MCGraw Hill
2. Data Structures using C, Reema Thareja, Third Edition, Oxford University Press

Reference Books:

1. Data Structures, Lipschutz, Schaum's Outline Series, Tata Mcgraw-hill
2. Data Structures Using C, Ch. Vijay Kumar, Pen Press International

Activities:

Outcome: Explain algorithm characteristics, time and space complexity, and asymptotic notations with clarity

Activity: Create a comparative chart of algorithms with different notations related to time and space complexities.

Evaluation Method: Rubric-based assessment of the chart for correctness, clarity, and depth of explanation on a 10-point scale.

Outcome: Implement and analyze different types of linked lists, including insertion, deletion, and traversal operations

Activity: Code a menu-driven program in C to implement single linked lists with all basic operations.

Evaluation Method: Practical lab assessment with test cases and Viva-style questioning to explain pointer manipulation.

Outcome: Develop stack and queue data structures using arrays and linked lists, and apply them in expression evaluation

Activity: Build a program to convert infix expressions to postfix and evaluate them using stacks; Implement queues using both arrays and linked lists with enqueue/dequeue operations.

Evaluation Method: Code review and execution of programs for sample cases and evaluation based on correctness and efficiency.

Outcome: Apply efficient searching and sorting algorithms to solve computational problems and evaluate performance trade-offs

Activity: Implement and compare sorting algorithms (e.g., selection sort and bubble sort) and searching algorithms (e.g., Linear vs. Binary Search) on datasets of varying sizes. Record number of swaps and iterations for preparing a chart to assimilate the results.

Evaluation Method: Performance report with graphs and analysis. Oral presentation or peer review discussing trade-offs and algorithm selection rationale.

Outcome: Construct and traverse tree and graph structures, using them to solve problems like shortest path and spanning trees

Activity: Implement binary trees and graphs using adjacency lists/matrices.

Evaluation Method: Lab demo with sample inputs and visual output (e.g., tree traversal order, graph paths).

SEMESTER-II

COURSE 3: DATA STRUCTURES USING C

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

List of Experiments

1. Write a program to read 'N' numbers of elements into an array and also perform the following operation on an array
   a. Add an element at the beginning of an array

   b. Insert an element at given index of array

   c. Update an element using a values and index

   d. Delete an existing element

2. Write a program to implement Single Linked List with insertion, deletion and traversal operations
3. Write a program to implement Doubly Linked List with insertion, deletion and traversal operations
4. Write a program to implement the Stack operations using Arrays and Linked Lists.
5. Write a program to convert a given infix expression to a postfix expression using stacks.
6. Write a program to implement the Queue operations using Arrays and Linked Lists.
7. Write a program to implement the Circular Queue operations using Arrays.
8. Write a program for Binary Search Tree Traversals
9. Write a program to search an item in a given list using the following Searching Algorithms
   a. Linear Search

   b. Binary Search.

10. Write a program for implementation of the following Sorting Algorithms
    a. Bubble Sort

    b. Insertion Sort

    c. Quick Sort

    d. Merge Sort

SEMESTER-II

COURSE 4: DIGITAL LOGIC DESIGN

Theory **Credits: 3** **3 hrs/week**

Course Objectives

1. Introduce the fundamentals of number systems, their conversions, and binary arithmetic operations.
2. Explore digital logic through gates, Boolean algebra, and simplification techniques for logic functions.
3. Develop proficiency in designing basic combinational circuits like adders and subtractors.
4. Equip students with the skills to implement advanced combinational components such as multiplexers, encoders, and decoders.
5. Foster understanding of sequential circuits, flip-flops, counters, and shift registers for system-level design.

Course Outcomes

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

1. Apply concepts of number systems to perform radix conversions and binary arithmetic using signed and unsigned formats.
2. Simplify logic functions using Boolean algebra, Karnaugh maps, and universal gates.
3. Design and analyze combinational circuits such as half adders, full adders, and subtractors.
4. Construct advanced combinational logic modules, including multiplexers, demultiplexers, encoders, decoders, and their hierarchical versions. Realize complex Boolean functions using combinations of logic modules.
5. Develop and evaluate sequential circuits such as flip-flops, latches, counters, and shift registers.

Unit 1: Number Systems:

Conversion of numbers from one radix to another radix, r's, (r-1)'s complements, signed binary numbers, addition and subtraction of unsigned and signed numbers, weighted and unweighted codes.

Add floating point Representation.

Unit 2. Logic Gates and Boolean Algebra:

NOT, AND, OR, universal gates, X-OR and X-NOR gates, Boolean laws and theorems, complement and dual of a logic function, canonical and standard forms, two level realization of logic functions using universal gates, minimizations of logic functions (POS and SOP) using Boolean theorems, K-map (up to four variables), don't care conditions.

Unit 3. Combinational Logic Circuits – 1:
Design of half adder, full adder, half subtractor, full subtractor, ripple adders and subtractors, ripple adder / subtractor.

Unit 4. Combinational Logic Circuits – 2:
Design of decoders, encoders, priority encoder, multiplexers, demultiplexers, higher order decoders, demultiplexers and multiplexers, realization of Boolean functions using decoders, multiplexers.

Unit 5. Sequential Logic Circuits:
Classification of sequential circuits, latch and flip-flop, RS- latch using NAND and NOR Gates, RS, JK, T and D flip-flops, truth tables and excitation tables, conversion of flip- flops, flip-flops with asynchronous inputs (preset and clear). Registers- shift registers, bidirectional shift registers, universal shift register, design of ripple counters, modulus counters.

Text Books:

1. Digital Design, M. Morris Mano, Michael D Ciletti,5th edition, Pearson.
2. Digital Logic Design, K.C. Rao, Ramana, Pen International Press

Reference Books:

1. Digital Electronics and Logic Design, Jaydeep Chakravorty, Universities Press
2. Digital Logic Design, Sonali Singh, BPB Publications

Activities:

Outcome: Apply concepts of number systems to perform radix conversions and binary arithmetic using signed and unsigned formats

Activity: Design a calculator in a spreadsheet or simulation tool (e.g., Logisim) that performs: Decimal ↔ Binary ↔ Hexadecimal conversions and binary arithmetic (addition, subtraction).

Evaluation Method: Rubric-based evaluation on a 10point scale (conversion accuracy, arithmetic correctness)

Outcome: Simplify logic functions using Boolean algebra, Karnaugh maps, and universal gates

Activity: Provide students with complex Boolean expressions and truth tables. Ask them to: Simplify using Boolean laws, Minimize using Karnaugh maps and Implement using only NAND or NOR gates

Evaluation Method: Worksheet submission with step-by-step simplification and evaluation of gate-level implementation using a 10-point scale.

Outcome: Design and analyze combinational circuits such as half adders, full adders, and subtractors

Activity: Build and simulate: Half adder and full adder using logic gates, and half and full subtractor circuits

Evaluation Method: Evaluate the correctness of the circuits for different inputs on a 10-point scale.

Outcome: Construct advanced combinational circuits, including multiplexers, demultiplexers, encoders and decoders.

Activity: Design Multiplexers for function selection, Decoders for control signal generation and Encoders for input compression

Evaluation Method: Project-based evaluation with functional demo and assessments based on a 10-point scale.

Outcome: Develop and evaluate sequential circuits such as flip-flops, latches, counters, and shift registers

Activity: Implement and test SR, JK, D, T flip-flops, asynchronous and synchronous counters using a simulator (E.g. Logisim, Multisim)

Evaluation Method: Lab assessment on a 10-point scale to understand the correctness of the circuit and presentation of the design.

SEMESTER-II

COURSE 4: DIGITAL LOGIC DESIGN

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

List of Experiments

The laboratory work can be done by using physical gates and necessary equipment or simulators.

Simulators: https://sourceforge.net/projects/gatesim/ or https://circuitverse.org/ or any free open-source simulator

1. Introduction to digital electronics lab- nomenclature of digital ICs, specifications, study of the data sheet, concept of Vcc and ground, verification of the truth tables of logic gates using TTL ICs.
2. Implementation of the given Boolean functions using logic gates in both SOP and POS forms
3. Realization of basic gates using universal gates.
4. Design and implementation of half and full adder circuits using logic gates.
5. Design and implementation of half and full subtractor circuits using logic gates.
6. Verification of stable tables of RS, JK, T and D flip-flops using NAND gates.
7. Implementation and verification of Decoder and encoder using logic gates.
8. Implementation of 4X1 MUX and DeMUX using logic gates.
9. Implementation of 8X1 MUX using suitable lower order MUX.
10. Implementation of 7-segment decoder circuit.
11. Implementation of 4-bit parallel adder.
12. Design and verification of 4-bit modulus counter.

SEMESTER-III

No change in syllabus.

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, 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 which having the following fields and button SAVE
    Form Fields are: Name, RNO, Mailid, Gender, Branch, Address.

No changes

SEMESTER-III

COURSE 6: DATABASE MANAGEMENT SYSTEMS

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives:

1. To understand the fundamentals of data, information, and the evolution from file-based systems to modern database management systems.
2. To develop the ability to design conceptual data models using Entity-Relationship (ER) and Enhanced ER diagrams.
3. To explore relational model principles, such as keys, integrity constraints, relational algebra and calculus, and normalization.
4. To perform data definition and manipulation using SQL commands including queries, joins, subqueries, views, and set operations.
5. To apply procedural logic using PL/SQL, incorporating control structures, functions, procedures, and database triggers.

Course Outcomes:

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

1. **Describe** the fundamentals of data, database systems, and the differences between file-based and database approaches. **Compare and classify** various DBMS architectures, data models, and their components, including the three-schema architecture.
2. **Design** conceptual data models using Entity-Relationship and Enhanced ER diagrams, applying generalization, specialization, and constraints.
3. **Apply** relational model concepts, including CODD rules, relational algebra, relational calculus, and normalization techniques.
4. **Construct and execute** SQL queries for data definition, manipulation, aggregation, joining, and subqueries, including views and set operations.
5. **Develop** PL/SQL programs incorporating control structures, procedures, and functions to manage database behavior effectively.

Unit 1. Overview of Database Management System:
Introduction to data, information, database, database management systems, file-based system, Drawbacks of file-Based System, database approach, Classification of Database Management Systems, advantages of database approach, Various Data Models, Components of Database Management System, three schema architecture of data base, costs and risks of database approach.

Unit 2. Entity-Relationship Model:
Introduction, the building blocks of an entity relationship diagram, classification of entity sets, attribute classification, relationship degree, relationship classification, reducing ER diagram to tables, enhanced entity-relationship model (EER model), generalization and specialization, IS A relationship and attribute inheritance, multiple inheritance, constraints on specialization and generalization, advantages of ER modeling.

Unit 3. Relational Model:
Introduction, CODD Rules, relational data model, concept of key, relational integrity, relational algebra, relational algebra operations, advantages of relational algebra, limitations of relational algebra, Functional dependencies and normal forms.

Unit 4. Structured Query Language:
Introduction, Commands in SQL, Data Types in SQL, Data Definition Language, Selection Operation, Projection Operation, Aggregate functions, Data Manipulation Language, Table Modification Commands, Join Operation, Set Operations, View, Sub Query.

Unit 5. PL/SQL:
Introduction, Shortcomings of SQL, Structure of PL/SQL, PL/SQL Language Elements,Data Types, Operators Precedence, Control Structures, Steps to Create a PL/SQL, Program, Iterative Control, Procedures, Functions.

Textbooks:

1. Database System Concepts, Avi Silberschatz, Henry F. Korth,S. Sudarshan, Seventh Edition, McGraw-Hill
2. Database Management Systems by Raghu Ramakrishnan, McGrawhill

Reference Books:

1. Fundamentals of Database Systems, Elmasri Navathe Pearson Education
2. An Introduction to Database systems, C.J. Date, A.Kannan, S.Swami Nadhan, Pearson

Activities:

Outcome: Describe the fundamentals of data, database systems, and the differences between file-based and database approaches. Compare and classify various DBMS architectures, data models, and their components, including the three-schema architecture.

Activity: Create a comparative presentation or infographic illustrating:

- File-based vs. DBMS approaches
- Types of DBMS architectures (1-tier, 2-tier, 3-tier)
- Data models and the three-schema architecture

Evaluation Method: Rubric-based assessment of the presentation covering clarity, accuracy, and depth of comparison. Include a short quiz to test conceptual understanding.

Outcome: Design conceptual data models using Entity-Relationship and Enhanced ER diagrams, applying generalization, specialization, and constraints.

Activity: Model a university or hospital database using ER and Enhanced ER diagrams that shows:

- Entity sets, relationships
- Generalization/specialization
- Participation and cardinality constraints

Evaluation Method: Diagram submission with peer review and instructor feedback. Use a checklist to assess completeness, correctness, and notation usage.

Outcome: Apply relational model concepts, including CODD rules, relational algebra, relational calculus, and normalization techniques.

Activity: Normalize a given unstructured dataset up to 3NF. Then, write relational algebra expressions for sample queries.

Evaluation Method: Written assignment graded on:

- Correctness of normalization steps
- Accuracy of relational algebra expressions
- Short-answer questions on CODD rules and relational calculus

Outcome: Construct and execute SQL queries for data definition, manipulation, aggregation, joining, and subqueries, including views and set operations.

Activity: Implement a mini-project (e.g., Library or Inventory DB) using SQL. Include:

- Table creation (DDL)
- Data manipulation (DML)
- Aggregation, joins, subqueries, views, and set operations

Evaluation Method: Lab-based practical test with query execution and output validation. Include a viva to explain logic and optimization.

Outcome: Develop PL/SQL programs incorporating control structures, procedures and functions to manage database behaviour effectively.

Activity: Build a PL/SQL-based payroll or student grading system using:

- Procedures and functions
- Control structures (IF, LOOP)
- Triggers for automated updates

Evaluation Method: Code review and demonstration. Evaluate based on:

- Syntax correctness
- Logical flow

SEMESTER-III

COURSE 6: DATABASE MANAGEMENT SYSTEMS

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

Experiment 1 : Database: Inventory Management

Table 1: Products

Structure:

Column Name	Data Type	Constraints
product_id	INT	PRIMARY KEY
product_name	VARCHAR(50)	NOT NULL
price	DECIMAL(10,2)	CHECK(price > 0)
stock_qty	INT	CHECK(stock_qty >= 0)

Sample Data:

product_id	product_name	price	stock_qty
1	Pen	10.00	100
2	Notebook	50.00	200
3	Stapler	120.00	50
4	Marker	25.00	80
5	File Folder	60.00	150

Table 2: Suppliers

Structure:

Column Name	Data Type	Constraints
supplier_id	INT	PRIMARY KEY
supplier_name	VARCHAR(50)	NOT NULL
contact_no	VARCHAR(20)	UNIQUE
product_id	INT	FOREIGN KEY REFERENCES Products(product_id)

Sample Data:

supplier_id	supplier_name	contact_no	product_id
101	StationeryMart	9876543210	1
102	PaperWorld	9876500000	2
103	OfficeSupplies	9876512345	3
104	MarkerHub	9876522222	4
105	FileDepot	9876533333	5

Section A: DDL (Data Definition Language)

1. Create a database called InventoryDB.
2. Create a table Products and table Suppliers with the specified columns and constraints:

Section B: DML (Data Manipulation Language)

4. Insert at least 5 rows into the Products table.
5. Insert at least 5 rows into the Suppliers table.
6. Update the stock quantity of product 'Pen' to 120.
7. Delete a supplier with a specific supplier_id.
8. Write a query to rename 'Notebook' to 'NoteBook A4'

Section C: DQL (SELECT Queries)

9. Display all records from the Products table.
10. Display only product_name and price of all products.
11. List all products that have a stock quantity less than 100.
12. Show all products between 20 and 100 price range.
13. Find all suppliers whose contact number starts with '98765'.
14. Find the average price of products.
15. Display the total number of products in the inventory.
16. Show the maximum and minimum stock quantities.
17. Count how many suppliers supply each product.
18. Show all products where price > 50 AND stock_qty > 100.
19. Show all products where price < 20 OR stock_qty < 80.
20. Display suppliers whose supplier_name contains the word 'Mart'
21. List all suppliers along with the product they supply (use INNER JOIN).
22. Display suppliers whose name starts with 'S'.
23. Find products whose name has exactly 5 characters
24. Find suppliers who supply products costing more than 100.

Experiment 2 : ONLINE BOOKSTORE DB

An online book store wants to implement a BOOKSTORE DB for managing their online transactions by using the following tables.

Authors Table

Column Name	Data Type	Constraints
author_id	INTEGER	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
nationality	VARCHAR	NULL allowed

Books Table

Column Name	Data Type	Constraints
book_id	INTEGER	PRIMARY KEY
Title	VARCHAR	NOT NULL
author_id	INTEGER	FOREIGN KEY REFERENCES Authors
publication_year	INTEGER	
Price	DECIMAL	

Customers Table

Column Name	Data Type	Constraints
customer_id	INTEGER	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
Email	VARCHAR	UNIQUE, NOT NULL
Address	VARCHAR	NOT NULL

Orders Table

Column Name	Data Type	Constraints
order_id	INTEGER	PRIMARY KEY
customer_id	INTEGER	FOREIGN KEY REFERENCES Customers
book_id	INTEGER	FOREIGN KEY REFERENCES Books
order_date	DATE	NOT NULL
quantity	INTEGER	NOT NULL

SAMPLE DATA SET for BOOKSTORE DB

Authors Table

author_id	first_name	last_name	nationality
1	Jane	Austen	British
2	George	Orwell	British
3	Gabriel	Garcia Marquez	Colombian
4	Toni	Morrison	American
5	Mark	Twain	American
6	Harper	Lee	American
7	Fyodor	Dostoevsky	Russian

Books Table

book_id	Title	author_id	publication_year	price
101	Pride and Prejudice	1	1813	12.99
102	1984	2	1949	9.50
103	One Hundred Years of Solitude	3	1967	15.00
104	Beloved	4	1987	11.25
105	Animal Farm	2	1945	8.75
106	Adventures of Huckleberry Finn	5	1884	10.50
107	To Kill a Mockingbird	6	1960	14.00

Customers Table

customer_id	first_name	last_name	Email	address
201	Alice	Smith	alice.s@example.com	12 Oak St, London
202	Bob	Johnson	bob.j@example.com	45 Pine Ave, Oxford
203	Charlie	Brown	charlie.b@example.com	78 Maple Rd, Bristol
204	Diana	Prince	diana.p@example.com	34 Queen St, York
205	Edward	Norton	edward.n@example.com	22 River Ln, Leeds
206	Fiona	Hall	fiona.h@example.com	56 Lake Dr, Bath
207	Greg	Miller	greg.m@example.com	89 Park Ave, Glasgow

Orders Table

order_id	customer_id	book_id	order_date	Quantity
301	201	101	2025-07-20	1
302	202	102	2025-07-21	2
303	201	105	2025-07-22	1
304	203	103	2025-07-23	1
305	204	106	2025-07-24	1
306	205	107	2025-07-25	3
307	206	104	2025-07-26	2

Section A: DDL (Schema Design & Constraints)

1. Write SQL statements to create all 4 tables (Authors, Books, Customers, Orders) with:
    - Primary Keys
    - Foreign Keys
    - Appropriate data types
    - NOT NULL constraints where necessary.

2. Alter the Books table to add a constraint that price must be greater than 0.
3. Add a new column phone_number to the Customers table (VARCHAR(15)) and ensure it is unique.
4. Drop the phone_number column from the Customers table.

Section B: DML (Data Manipulation)

5. Insert at least 7 records for each table (use sample dataset above).
6. Update the price of the book titled *Animal Farm* by increasing it by 10%.
7. Delete all orders made before 2025-07-21.
8. Change the nationality of Gabriel Garcia Marquez to "Latino-American".

Section C: SELECT Queries (Data Querying)

9. List all books published between 1900 and 2000.
10. Find all customers whose email contains "example.com".
11. Retrieve books whose price is between 10 and 15 and published before 1950.
12. Show authors who are either 'British' or 'American'.
13. Find books that have a price less than 10 or are published after 1980.
14. Display all orders placed after 2025-07-22.
15. List all books written by author with author_id = 2.
16. Find customers whose last name starts with B.
17. Show all books with a price NOT between 9 and 13.
18. Display books whose publication_year is in (1813, 1945, 1987).
19. Find authors whose nationality is NOT 'British'.
20. List customers whose address contains the word Park.
21. Show all books sorted by price in descending order.
22. List authors in alphabetical order by last_name.
23. Display orders sorted by order_date (latest first).

Use of Date Functions

24. Show all orders placed in July 2025.
25. Show all orders with an estimated delivery date (5 days after order date).
26. Show customers who placed an order on a weekend.
27. Calculate how many days have passed since the last order was placed.

Aggregate Functions (COUNT, SUM, AVG, MIN, MAX)

28. Count the total number of books in the database.
29. Find the average price of all books.
30. Show the highest-priced book.

31. Count how many orders each customer has placed.
32. Calculate the total sales (price × quantity) for each customer.

GROUP BY and HAVING

33. Count how many books are written by each author.
34. Group orders by customer_id and display total quantity ordered.
35. Show customers who have ordered more than 2 books in total (use HAVING).
36. Find the total number of books sold per author (GROUP BY author).

Experiment 3: EMPLOYEE DB

An enterprise wants to automate its employee management process by implementing an Employee Database.The goal is to replace manual record-keeping with a centralized system that stores employee, department, and project details. Use the following table structures and data set to implement Employee DB.

EmployeeDB – Table Structures

1. Departments Table

Column	Type	Constraints
dept_id	INT	PRIMARY KEY
dept_name	VARCHAR	UNIQUE, NOT NULL
location	VARCHAR	NOT NULL

2. Employees Table

Column	Type	Constraints
emp_id	INT	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
email	VARCHAR	UNIQUE, NOT NULL
phone	VARCHAR	CHECK (phone LIKE '--____')
hire_date	DATE	NOT NULL
job_title	VARCHAR	NOT NULL
salary	DECIMAL	CHECK (salary > 0)

dept_id	INT	FOREIGN KEY REFERENCES Departments(dept_id)
manager_id	INT	FOREIGN KEY REFERENCES Employees(emp_id) (self-referential)

3. Projects Table

Column	Type	Constraints
project_id	INT	PRIMARY KEY
project_name	VARCHAR	NOT NULL
start_date	DATE	NOT NULL
end_date	DATE	NULL
dept_id	INT	FOREIGN KEY REFERENCES Departments(dept_id)

4. Employee_Project Table (Many-to-Many)

Column	Type	Constraints
emp_id	INT	FOREIGN KEY REFERENCES Employees(emp_id), PRIMARY KEY(emp_id, project_id)
project_id	INT	FOREIGN KEY REFERENCES Projects(project_id)
hours_allocated	INT	CHECK (hours_allocated > 0)

Sample Data Set

Departments Table

dept_id	dept_name	Location
1	HR	New York
2	IT	San Francisco
3	Finance	Chicago

4	Marketing	Boston
5	Operations	Seattle
6	Legal	Washington D.C.
7	Sales	Dallas
8	R&D	Austin
9	Procurement	Denver
10	Customer Care	Miami

2. Employees Table

emp_id	first_name	last_name	Email	phone	hire_date	job_title	salary	dept_id	manager_id
101	Alice	Johnson	alice.j@corp.com	123-456-7890	2020-03-15	HR Manager	75000	1	NULL
102	Bob	Smith	bob.s@corp.com	234-567-8901	2019-05-20	IT Analyst	65000	2	104
103	Charlie	Brown	charlie.b@corp.com	345-678-9012	2021-01-10	Finance Executive	58000	3	106
104	Diana	Prince	diana.p@corp.com	456-789-0123	2018-07-12	IT Manager	90000	2	NULL
105	Ethan	Hunt	ethan.h@corp.com	567-890-1234	2022-02-25	Marketing Lead	62000	4	NULL
106	Fiona	Hall	fiona.h@corp.com	678-901-2345	2017-11-01	Finance Manager	85000	3	NULL
107	Greg	Miles	greg.m@cor	789-012-	2023-	IT	45000	2	104

			p.com	3456	04-15	Support			
108	Hannah	White	hannah.w@corp.com	890-123-4567	2021-09-05	HR Executive	50000	1	101
109	Ian	Scott	ian.s@corp.com	901-234-5678	2020-11-20	Operations Analyst	56000	5	NULL
110	Julia	Adams	julia.a@corp.com	012-345-6789	2019-12-18	Legal Advisor	70000	6	NULL

3. Projects Table

project_id	project_name	start_date	end_date	dept_id
201	Payroll System	2023-01-01	NULL	3
202	Website Upgrade	2023-02-10	NULL	2
203	Recruitment Drive	2023-03-05	NULL	1
204	Ad Campaign	2023-05-20	NULL	4
205	New CRM Tool	2023-04-15	NULL	7
206	Compliance Portal	2023-06-10	NULL	6
207	Inventory System	2023-07-01	NULL	5
208	AI Research	2023-08-05	NULL	8
209	Customer Feedback	2023-09-10	NULL	10
210	Procurement System	2023-10-01	NULL	9

4. Employee_Project Table

emp_id	project_id	hours_allocated
102	202	120
104	202	80
103	201	100
106	201	150
101	203	50
105	204	70
107	202	60
109	207	90
110	206	110
108	203	40

Section A: DDL (Schema Creation & Modification)

1. Write SQL statements to create the above tables with the specified constraints
2. Alter the Employees table to add a column bonus DECIMAL(8,2) with default value0.
3. Drop the column bonus from Employees.

Section B: DML (Insert, Update, Delete)

4. Insert at least 10 rows into Departments, Employees, Projects, and Employee_Project.(use the above data set)
5. Try inserting an employee with a negative salary (should fail due to CHECK constraint).
6. Update the salary of the employee with emp_id = 103 by 15%.
7. Delete an employee record who has resigned (choose any emp_id).
8. Increase all employees' salaries in the IT department by 5%.
9. Change the department of an employee to "Research".(should fail due to FK constraint)

Section C: DQL (Select Queries)

10. List all employees and their details.
11. Show all employees in the "HR" department.
12. Find employees with salaries between 50,000 and 80,000.
13. Retrieve employees hired after 2020.
14. Show employees who are in either the IT or Finance department.
15. Find employees whose email ends with "@corp.com".
16. List all employees with salary > 60,000 AND located in "New York".
17. Display employees in descending order of salary.
18. Count the number of employees in each department.
19. Show the average salary of employees department-wise.
20. Display departments where the average salary is greater than 70,000.
21. Find the number of employees in each project.
22. Display departments with more than 3 employees.
23. Show the sum of all salaries department-wise.
24. List all distinct department IDs from the Employees table.
25. Show employee names with the year they were hired.
26. Show employees grouped by the year of hire.
27. List employees hired in the last 90 days.
28. List the no of years of experience of all the employees

Section D: Joins

29. List all employees with their department names (INNER JOIN).
30. Display all departments along with employees, including those departments without employees (LEFT JOIN).
31. Show employees and the projects they are working on (JOIN 3 tables: Employees, Employee_Project, Projects).
32. List projects along with total hours allocated by employees.
33. Write a query to find employees who are working on more than one project.

34. Show all projects handled by the 'Finance' department.

Section E: PL/SQL Programming

1. Write a procedure GetEmpInfo that takes emp_id as input and displays name, salary, and department.
2. Write a PL/SQL block that checks if an employee's salary is above 50,000.
   If yes, print "High Salary" ;Otherwise print "Standard Salary".
3. Write a PL/SQL program to display the top 10 rows in the Emp table based on their job and salary
4. Write a stored procedure GiveBonus that takes department ID and a designation as input, along with a bonus amount, and updates the salary of all employees in that department who have the specified designation by adding the bonus amount to their current salary.
5. Create a trigger to prevent inserting employees with a salary less than 30,000.
6. Create a trigger to avoid any transactions(insert, update, delete) on EMP table on Saturday & Sunday.

No change

No changes

SEMESTER-III

COURSE 7: 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, 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 7: 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.

No change

No change

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, 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 the foundational principles and evolution of operating systems, including resource abstraction and core management functions.

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: Analyze process and thread management, focusing on system calls, kernel modes, scheduling algorithms, and threading models.

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, including processor modes, system calls, kernel functions, and scheduling algorithms.

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 approaches, applying classical concurrency solutions and inter-process communication methods.

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, incorporating allocation techniques, virtual memory models, disk scheduling, and OS-level security features.

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 techniques

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

No change

No Change

SEMESTER-IV

COURSE 9: COMPUTER NETWORKS

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives

1. **Introduce foundational concepts** and architecture of computer networks, including OSI and TCP/IP models.
2. **Explain the functionalities of network layers**, from physical to application, highlighting their roles in communication systems.
3. **Explore data transmission technologies**, including guided and wireless media, and assess their real-world applications.
4. **Develop understanding of core protocols**, error handling mechanisms, and routing algorithms used in modern networks.
5. **Demonstrate how network services operate**, focusing on protocols like TCP, UDP, HTTP, DNS, and their relevance in global connectivity.

Course Outcomes

These define what students should be able to do after successful completion, At the End of the Course, The Students will be able to:

1. **Describe network models** (OSI, TCP/IP) and differentiate between network hardware and software components.
2. **Analyze data transmission techniques** and select appropriate media for specific networking scenarios.
3. **Apply error control and flow protocols** (e.g., sliding window, ALOHA) to optimize link-layer communication.
4. **Evaluate routing strategies and congestion control algorithms** within network environments, including Internet-based systems.
5. **Implement basic application-layer protocols** and illustrate how services like email, web browsing, and streaming are supported on networks.

Unit 1. Introduction to Computer Networks:

Network hardware, Network software, OSI, TCP/IP Reference models, Example Networks: ARPANET, Internet.

Physical Layer: Guided Transmission media: twisted pairs, coaxial cable, fiber optics, Wireless transmission.

Unit 2. Data link layer:
Design issues, framing, Error detection and correction. Elementary data link protocols,Sliding Window protocols.

Medium Access sub layer: The channel allocation problem.

Multiple access protocols: ALOHA, Carrier sense multiple access protocols, collision free protocols.

Wireless LANs, Data link layer switching.

Unit 3. Network Layer:
Design issues, Routing algorithms: shortest path routing, Flooding, Hierarchical routing, Broadcast, Multicast, distance vector routing, Congestion Control Algorithms, Quality of Service, Internetworking, The Network layer in the internet.

Unit 4. Transport Layer:
Transport Services, Elements of Transport protocols, Connection management, TCP and UDP protocols.

Unit 5. Application Layer:
Domain name system, SNMP, Electronic Mail, SMTP, World Wide Web, HTTP and HTTPS.

Textbooks:

1. Computer Networks -- Andrew S Tanenbaum, David. j. Wetherall, 5th Edition. Pearson Education/PHI
2. Data Communications and Networking – Behrouz A. Forouzan. Third Edition TMH.

Reference Books:

1. An Introduction to Computer Networks- Peter Lars Dordal,: Loyola University Chicago (2022); eBook (Creative Commons Licensed)
2. The TCP/IP Guide: A comprehensive, Illustrated Internet Protocols reference, Charles M. Kozierok

Activities:
Outcome: Describe Network Models (OSI, TCP/IP) and Differentiate Hardware vs. Software Components
Activity: Create a **layered diagram** of OSI and TCP/IP models using colored cards or digital tools. Label each layer with its function and examples of hardware (e.g., router, switch) and software (e.g., protocols, applications).
Evaluation Method: Short quiz with:

- Matching layers to functions
- Identifying hardware/software roles
- One scenario-based question (e.g., Which layer handles routing?)

Outcome: Analyze data transmission techniques and select appropriate media for specific networking scenarios.

Activity: Use a **scenario worksheet** with different environments (e.g., office LAN, rural broadband, mobile network). Students choose appropriate transmission media (e.g., fiber, coaxial, wireless) and justify their choices.

Evaluation Method: Peer-reviewed worksheet with rubric:

- Correct media selection
- Justification clarity
- Understanding of bandwidth, cost, and distance factors

Outcome: Apply error control and flow protocols (e.g., sliding window, ALOHA) to optimize link-layer communication.

Activity: Simulate **sliding window and ALOHA protocols** using tokens or cards to represent frames. Students act as sender/receiver and demonstrate retransmission, acknowledgments, and flow control.

Evaluation Method: Evaluate students on a 10-point scale based on:

- Correct protocol steps
- Handling of errors and retransmissions
- Flow control logic

Plus a brief reflection sheet explaining what they learned

Outcome: Evaluate routing strategies and congestion control algorithms within network environments, including Internet-based systems.

Activity: Use a **network simulation tool** (e.g., Cisco Packet Tracer or NetSim) to compare routing algorithms (e.g., Dijkstra, Distance Vector). Introduce congestion and observe how algorithms respond.

Evaluation Method: Evaluate students on a 10-point scale based on Lab report submitted by students with:

- Routing table snapshots
- Congestion response analysis
- Efficiency comparison

Include a rubric for clarity, accuracy, and insight

Outcome: Implement basic application-layer protocols and illustrate how services like email, web browsing, and streaming are supported on networks.

Activity: Set up a **mini-network** or use simulation to demonstrate:

- Email (SMTP/POP3)
- Web browsing (HTTP/HTTPS)

Evaluation Method: Practical demo + oral explanation on a 10-point scale:

- Protocol identification
- Service flow (client-server interaction)

SEMESTER-IV

COURSE 9: COMPUTER NETWORKS

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

List of Experiments:

1. Understanding various network tools in Windows and Linux
2. Study different types of Network devices and Cables
3. Building a Local Area Network.
4. Concept of Network IP Address
5. Introduction to Network Simulator – Packet Tracer (PT)
6. Configuration of a Router using Packet Tracer
7. Implementation of a Network using Packet Tracer
8. Implementation of Static Routing using Packet Tracer
9. Implementation of RIP using Packet Tracer
10. Implementation of OSPF using Packet Tracer
11. Implement DNS using packet tracer
12. Implementation of a VLAN using Packet Tracer

No change

No change

SEMESTER-IV

COURSE 10: 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, 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 10: 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 numbers 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. Write a program to connect to the database and retrieve the required information using SQL commands.

No change

No change

SEMESTER-V

COURSE 11: 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, 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 (10-point scale):

- 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-V

COURSE 11: 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 IEEE standard SRS document. Also develop risk management and project plan (Gantt chart).
2. Understanding of System modeling: Data model i.e. ER – Diagram and draw the ER Diagram with 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

No change

SEMESTER-V

COURSE 12 A: WEB INTERFACE DESIGNING TECHNOLOGIES

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives

1. Understand the principles of web design and distinguish between web and desktop application architectures. Develop static web pages using HTML elements, attributes, and multimedia integration techniques.
2. Style web pages effectively using CSS, including layout control, responsive design, and UI enhancements.
3. Implement dynamic behaviors and form validations using JavaScript and the Document Object Model (DOM).
4. Explore client-side scripting techniques to build responsive, interactive interfaces.
5. Gain foundational knowledge of Content Management Systems (CMS) and apply practical skills in platforms such as WordPress.

Course Outcomes

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

1. Design and structure HTML-based webpages incorporating text, images, tables, forms, and multimedia content.
2. Apply CSS styling rules to manage layout aesthetics, interactivity, and responsiveness across devices.
3. Use JavaScript for string manipulation, event handling, arrays, object operations, and basic validation.
4. Employ client-side scripting to enhance form functionality, create dialog interactions, and add animation via mouse and keyboard events.
5. Analyze different types of CMS and operate WordPress features like posts, pages, themes, plug-ins, and SEO tools to create and deploy basic websites.

Unit 1.HTML:

Introduction to web designing, difference between web applications and desktop applications, introduction to HTML, HTML structure, elements, attributes, headings, paragraphs, images, tables, lists, blocks, symbols, embedding multi-media components in HTML, HTML forms

Unit 2.CSS:

CSS home, introduction, syntax, CSS combinators, colors, background, borders, margins, padding, height/width, text, fonts, tables, lists, position, overflow, float, pseudo class, pseudo elements, opacity, tool tips, image gallery, CSS forms, CSS counters.

Unit 3.Java Script:

What is DHTML, JavaScript, basics, variables, operators, statements, string manipulations, mathematical functions, arrays, functions. objects, regular expressions, exception handling.

Unit 4. Client-Side Scripting:
Accessing HTML form elements using Java Script object model, basic data validations, data format validations, generating responsive messages, opening windows using java script, different kinds of dialog boxes, accessing status bar using java script, embedding basic animative features using different keyboard and mouse events.

Unit 5. Content Management Systems
Introduction to CMS: What is a CMS?, Types: traditional, headless, cloud-based

Popular CMS Platforms: WordPress, Joomla, Drupal, Shopify, When to choose each

Wordpress Basics: Introduction to word press, features, and advantages, wordpress (hosted access and local access), working with posts, managing pages, working with media, working with widgets, working with themes, extending wordpress with plug-ins, SEO and deployment.

Text Book(s)

1. Web Programming Building Internet Applications, Chris Bates, Second Edition, Wiley
2. An Introduction to Web Design plus Programming, Paul S.WangSanda S. Katila, Thomson.

Reference Books

1. Head First HTML and CSS, Elisabeth Robson, Eric Freeman, O'Reilly Media Inc.
2. An Introduction to HTML and JavaScript: for Scientists and Engineers, David R. Brooks. Springer, 2007
3. Schaum's Easy Outline HTML, David Mercer, Mcgraw Hill Professional.
4. Wordpress for Beginners, Dr.Andy Williams

Activities:

Outcome: Design and structure HTML-based webpages incorporating text, images, tables, forms, and multimedia content.

Activity: Create a personal profile webpage using HTML that includes:

- Text (headings, paragraphs)
- Images
- Tables (e.g., contact info or skills)
- Forms (e.g., feedback form)
- Embedded multimedia (e.g., YouTube video or audio clip)

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

- Correct use of HTML tags
- Proper structure and nesting
- Functionality of form and media elements
- Visual clarity and completeness

Outcome: Apply CSS styling rules to manage layout aesthetics, interactivity, and responsiveness across devices.

Activity: Style the personal profile page using CSS:

- Apply colors, fonts, spacing
- Use Flexbox or Grid for layout
- Add hover effects and media queries for responsiveness

Evaluation Method: Rubric-based assessment on a 10-point scale:

- Visual appeal and consistency
- Responsive behaviour on different screen sizes
- Use of selectors and layout techniques
- Code cleanliness and organization

Outcome: Use JavaScript for string manipulation, event handling, arrays, object operations, and basic validation.

Activity: Enhance the profile page with JavaScript:

- Validate form inputs (e.g., email format)
- Display a greeting using string manipulation
- Use arrays/objects to store and display hobbies or skills
- Handle click events on button

Evaluation Method: Code demonstration and output testing (10-point scale):

- Correct use of JS syntax and logic
- Functionality of validation and event handling
- Use of arrays/objects
- Console output or alert behaviour

Outcome: Employ client-side scripting to enhance form functionality, create dialog interactions, and add animation via mouse and keyboard events.

Activity: Add interactive features to the form:

- Show/hide sections using mouse events
- Trigger animations on keypress
- Display confirmation dialog on form submission

Evaluation Method: Live demo and observation (10-point scale):

- Smooth interaction and feedback
- Correct use of event listeners
- Animation and dialog behavior
- User experience quality

Outcome: Analyze different types of CMS and operate WordPress features like posts, pages, themes, plug-ins, and SEO tools to create and deploy basic websites.

Activity: Create a basic website using WordPress:

- Add posts and pages
- Apply a theme and customize layout
- Install plug-ins (e.g., contact form, SEO tool)
- Configure basic SEO settings

Evaluation Method: Site walkthrough and checklist (10-point scale):

- Functionality of posts/pages
- Theme customization
- Deployment readiness

SEMESTER-V

COURSE 12 A: WEB INTERFACE DESIGNING TECHNOLOGIES

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

List of Experiments:

1. Create an HTML document with the following formatting options:

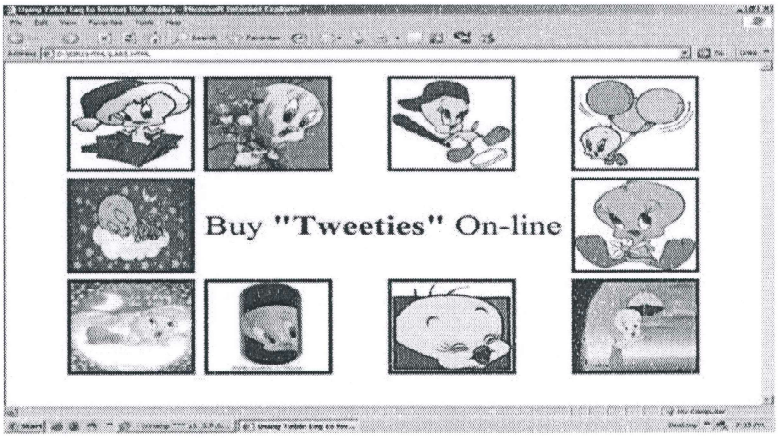
   (a) Bold, (b) Italics, (c) Underline, (d) Headings (Using H1 to H6 heading styles), (e) Font (Type, Size and Color), (f) Background (Colored background/Image in background), (g) Paragraph, (h) Line Break, (i) Horizontal Rule, (j) Pre tag

2. Create an HTML document which consists of:

   (a) Ordered List (b) Unordered List (c) Nested List (d) Image

3. Create a Table with four rows and five columns. Place an image in one column.

4. 4.Collect any ten images of your choice. Using table tag, align the images as follows:



5. Create a menu form using HTML.

6. Style the menu buttons using CSS.

7. Create a form using HTML which has the following types of controls:

   (a) Text Box (b) Option/radio buttons (c) Check boxes (d) Reset and Submit buttons

8. Embed a calendar object in your web page.

9. Create a form that accepts the information from the subscriber of a mailing system.

Word press:

10. Installation and configuration of word press
11. Access admin panel and manage posts
12. Access admin panel and manage pages
13. Add widgets and menus
14. Create users and assign roles
15. Create a site and add a theme to it

No change

SEMESTER-V

COURSE 12 B: DATA SCIENCE WITH R

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives

1. Introduce key mathematical and statistical tools essential for data analysis.
2. Explain the Data Science process, lifecycle, and its applications in diverse domains.
3. Develop proficiency in R programming for data manipulation and basic analytics.
4. Teach effective data handling, transformation, and visualization techniques using R.
5. Explore practical use cases in Data Science with modeling, clustering, and ethical awareness.

Course Outcomes

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

1. Apply statistical methods and probability distributions to analyze and interpret data.
2. Describe the Data Science workflow and perform exploratory data analysis (EDA).
3. Use R programming constructs and libraries for data input, control flows, and functions.
4. Handle and clean datasets using R tools like dplyr, tidyr, and manage missing/time-based data.
5. Implement basic machine learning models and evaluate performance using appropriate metrics and visual tools.

Unit 1. Mathematical & Statistical Foundations:
Sets, Functions, Probability, Random Variables, Descriptive Statistics: Mean, Median, Mode, Variance, Standard Deviation, Probability Distributions: Binomial, Normal, Poisson, Hypothesis Testing: t-test, Chi-square test, Correlation & Regression

Unit 2. Introduction to Data Science Process:
Introduction- Definition - Data Science in various fields - Examples - Impact of Data Science - Data Analytics Life Cycle - Data Science Toolkit - Data Scientist - Data Science Team, Exploratory Data Analysis (EDA), Feature Engineering & Data Transformation

Unit 3. Basics of R Programming:
Introduction to R and RStudio, Data Types, Variables, Operators, Control Structures (if, loops), Functions and Packages, Data Input/Output (CSV, Excel, XML, JSON).

Unit 4. Data Handling & Visualization in R:
Data Frames, Lists, Matrices, dplyr and tidyr for Data Wrangling, Handling Missing Data, Working with Date/Time in R.

Unit 5. Applications & Case Studies in Data Science:
Simple Linear Regression, Model Evaluation: Accuracy, Confusion Matrix, ROC.

K-Means Clustering, Text Mining & Word Clouds, Recommender Systems Basics, Ethical Issues in Data Science

Textbooks

1. An Introduction to Statistical Learning with Applications in R, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Springer, 2nd Edition, 2021
2. R for Data Science, Hadley Wickham and Garrett Grolemund, O'Reilly Media, 2017.

Reference Books

1. The Art of R Programming, Norman Matloff,, No Starch Press, 2011.
2. Modern Applied Statistics with S, W.N. Venables & B.D. Ripley, Springer, 2002.
3. Introduction to Data Science: Data Analysis and Prediction Algorithms with R, Rafael A. Irizarry, CRC Press, 2020.
4. Data Science from Scratch: First Principles with Python (for conceptual clarity only), Joel Grus,

Activities:

Outcome: Apply statistical methods and probability distributions to analyze and interpret data.

Activity: Analyze a dataset (e.g., student scores or sales data) to:

- Calculate mean, median, mode, variance, and standard deviation
- Fit and visualize a normal distribution
- Apply probability rules to answer questions (e.g., likelihood of scoring above 80)

Evaluation Method: Worksheet-based assessment (10-point scale):

- Accuracy of statistical calculations
- Correct use of probability formulas
- Interpretation of results and distribution plots

Outcome: Describe the Data Science workflow and perform exploratory data analysis (EDA).

Activity: Use a real-world dataset (e.g., Titanic or COVID data) to:

- Outline the steps of the Data Science workflow
- Perform EDA using summary statistics and visualizations (histograms, boxplots, scatterplots)

Evaluation Method: Presentation and checklist (10-point scale):

- Clear explanation of workflow stages
- Quality of EDA insights
- Use of appropriate plots and summaries

Outcome: Use R programming constructs and libraries for data input, control flows, and functions.

Activity: Write an R script that:

- Reads a CSV file
- Uses if, for, and while loops
- Defines and calls custom functions with arguments and return values

Evaluation Method: Code review and execution test to verify (10-point scale):

- Correctness of the syntax and logic
- Functionality of control structures
- Output accuracy and modularity

Outcome: Handle and clean datasets using R tools like dplyr, tidyr, and manage missing/time-based data.

Activity: Clean a messy dataset using:

- dplyr for filtering, selecting, and mutating
- tidyr for reshaping and handling missing values
- Time-based operations (e.g., filling gaps, formatting dates)

Evaluation Method: Before-and-after comparison (10 point score):

- Completeness of cleaning steps
- Use of appropriate functions
- Handling of missing/time data

Outcome: Implement basic machine learning models and evaluate performance using appropriate metrics and visual tools.

Activity: Build a simple classification model (e.g., logistic regression or decision tree) using R:

- Train/test split
- Predict outcomes
- Evaluate using confusion matrix, accuracy, precision, recall

Evaluation Method: Model report and demo (10 point scale):

- Correct implementation of model
- Use of evaluation metrics

SEMESTER-V

COURSE 12 B: DATA SCIENCE WITH R

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

List of Practicals:

1. Compute Mean, Median, Mode, Variance, and Standard Deviation
2. Visualize Binomial, Normal, and Poisson Distributions
3. Perform t-test and Chi-Square Test in R
4. Calculate Correlation and Build a Simple Linear Regression Model
5. Conduct Exploratory Data Analysis (EDA) on a Real-World Dataset
6. Apply Feature Engineering: Scaling, Normalization, and Encoding
7. Practice R Programming: Variables, Control Structures, and Functions
8. Read and Write Data from CSV, Excel, JSON, and XML Files
9. Use dplyr and tidyr for Data Wrangling Tasks
10. Handle Missing Data and Detect Outliers
11. Work with Dates and Times in R
12. Visualize Data Using ggplot2 (Bar, Scatter, Histogram, Boxplot)
13. Perform K-Means Clustering and Visualize Clusters
14. Evaluate Models Using Confusion Matrix, Accuracy, and ROC Curve
15. Perform Text Mining and Create a Word Cloud

No change

No Change

SEMESTER-V

COURSE 13 A: WEB APPLICATION DEVELOPMENT USING PHP & MySQL

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives:

1. Understand the foundational elements of PHP, including variables, data types, operators, and flow control functions.
2. Develop proficiency in managing arrays, objects, strings, dates, and time functionalities using PHP.
3. Design and process HTML forms integrated with PHP, including advanced operations like file uploads, redirection, and exception handling.
4. Implement session management and cookie handling to preserve user state and provide secure, personalized experiences.
5. Connect PHP with MySQL databases, enabling learners to perform CRUD operations and build dynamic web applications.

Course Outcomes

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

1. Demonstrate effective use of PHP building blocks such as variables, expressions, constants, control structures, and functions with appropriate scope and argument handling.
2. Manipulate complex data structures including arrays and objects, and utilize PHP string and date/time functions for dynamic content generation.
3. Create functional web forms using PHP, retrieve form inputs, manage file uploads, and execute form-based operations like redirection and email dispatch.
4. Apply secure techniques for maintaining user sessions and cookies, including session lifecycle control, session variable manipulation, and user authentication workflows.
5. Integrate PHP with MySQL to build database-driven web components, perform record management, and architect structured menu-based data operations.

Unit 1. The building blocks of PHP:
Variables, Data Types, Operators and Expressions, Constants.

Flow Control Functions in PHP: Switching Flow, Loops, Code Blocks and Browser Output.

Working with Functions: Creating functions, Calling functions, Returning the values from User-Defined Functions, Variable Scope, Saving state between Function calls with the static statement, arguments of functions

Unit 2.Working with Arrays:
Creating Arrays, Some Array-Related Functions.

Working with Objects: Creating Objects, Accessing Object Instances,

Working with Strings, Dates and Time: Formatting strings with PHP, Manipulating Strings with PHP, Using Date and Time Functions in PHP.

Unit 3.Working with Forms:
Creating Forms, Accessing Form Input with User defined Arrays, Combining HTML and PHP code on a single Page, Using Hidden Fields to save state, Page redirection, Sending Mail on Form Submission, **Working with File Uploads**, Managing files on server, **Exception handling.**

Unit 4. Working with Cookies and User Sessions:
Introducing Cookies, setting a Cookie with PHP, Session Function Overview, starting a Session, working with session variables, passing session IDs in the Query String, Destroying Sessions and Unsetting Variables, Using Sessions in an Environment with Registered Users.

Unit 5.Interacting with MySQL using PHP:
MySQL Versus MySQLi Functions, connecting to MySQL with PHP, Working with MySQL Data. Planning and Creating Database Tables, Creating Menu, Creating Record Addition Mechanism, Viewing Records, Creating the Record Deletion Mechanism.

Text Book(s)

1. SAMS Teach yourself PHP MySQL and Apache,, Julie C. Meloni, Pearson Education
2. PHP: The Complete Reference, Steven Holzner , McGraw-Hill

Reference Books

1. Learning PHP, MySQL, JavaScript, CSS & HTML5, Robin Nixon, Third Edition, O'reilly, 2014
2. The web warrior guide to Web Programming, Xue Bai Michael Ekedahl,, Thomson, 2006.

Activities:

Outcome: Demonstrate effective use of PHP building blocks such as variables, expressions, constants, control structures, and functions with appropriate scope and argument handling.

Activity: Write a PHP script that:

- Declares variables and constants
- Uses expressions and control structures (if, switch, loops)
- Defines and calls functions with arguments and return values
- Demonstrates variable scope (global, local)

Evaluation Method: Evaluate on a 10-point scale based on code review checklist to verify the

- Correct syntax and use of each building block
- Logical flow using control structures
- Proper function definition and scope handling
- Output accuracy and readability

Outcome: Manipulate complex data structures including arrays and objects, and utilize PHP string and date/time functions for dynamic content generation.

Activity: Create a PHP script that:

- Stores student data in arrays and objects
- Formats and manipulates strings (e.g., name formatting)
- Displays current date/time and calculates age from DOB

Evaluation Method: Rubric-based assessment on a 10-point scale to check the:

- Correct use of arrays and objects
- Effective string and date/time functions
- Dynamic output generation
- Code clarity and structure

Outcome: Create functional web forms using PHP, retrieve form inputs, manage file uploads, and execute form-based operations like redirection and email dispatch.

Activity: Build a contact form that:

- Accepts name, email, message
- Uploads a file (e.g., resume)
- Sends an email confirmation
- Redirects to a thank-you page

Evaluation Method: Evaluate on a 10-point scale based on functional testing to perform:

- Form input retrieval and validation
- File upload success
- Email dispatch and redirection
- Error handling and user feedback

Outcome: Apply secure techniques for maintaining user sessions and cookies, including session lifecycle control, session variable manipulation, and user authentication workflows.

Activity: Develop a login system that:

- Starts a session on login
- Stores user data in session variables
- Sets a cookie for Remember Me
- Logs out and destroys session securely

Evaluation Method: Security checklist (10-point scale):

- Session lifecycle control
- Cookie setup with secure flags
- Authentication logic
- Session/cookie cleanup on logout

Outcome: Integrate PHP with MySQL to build database-driven web components, perform record management, and architect structured menu-based data operations.

Activity: Create a student management system:

- Connect to MySQL database
- Add, view, update, delete student records
- Display menu-based navigation (e.g., by class or grade)

Evaluation Method: Database interaction test on a 10-point scale:

- Successful CRUD operations
- Structured menu navigation
- SQL query correctness

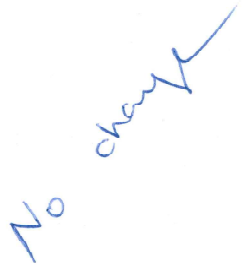
SEMESTER-V

COURSE 13 A: WEB APPLICATION DEVELOPMENT USING PHP & MySQL

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

List of Experiments:

1. Write a PHP program to Display Hello
2. Write a PHP Program to display today's date.
3. Write a PHP program to display Fibonacci series.
4. Write a PHP Program to read the employee details.
5. Write a PHP program to prepare the student marks list.
6. Create student registration form using text box, check box, radio button, select, submit button. And display user inserted values in the new PHP page.
7. Create Website Registration Form using text box, check box, radio button, select, submit button. And display user inserted values in the new PHP page.
8. Write a PHP script to demonstrate passing variables with cookies.
9. Write a PHP script to connect to the MySQL server from your website.
10. Write a program to keep track of how many times a visitor has loaded the page.
11. Write a PHP application to perform CRUD (Create, Read, Update and Delete) operations on a database table.
12. Create a web site using any open-source framework built on PHP and MySQL – It is a team activity wherein students are divided into multiple groups and each group comes up with their own website with basic features.



SEMESTER-V

No change

COURSE 13 B: PYTHON FOR DATA SCIENCE

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives:

1. Introduce foundational concepts of NumPy arrays and array operations for efficient numerical computing.
2. Teach key data structures and manipulation techniques using Pandas.
3. Enable students to perform data input/output operations and implement basic data cleaning workflows.
4. Explore string processing methods and feature engineering strategies in Pandas.
5. Guide learners in advanced data wrangling tasks including merging, reshaping, and hierarchical indexing.

Course Outcomes:

1. Demonstrate proficiency in creating and manipulating NumPy arrays for mathematical operations and simulations.
2. Apply Pandas Series and DataFrame operations for structured data handling and analysis.
3. Read, write, and clean diverse data formats using Python tools, addressing missing values and outliers.
4. Implement vectorized string operations and create derived features for enhanced model readiness.
5. Perform complex data wrangling tasks such as merging datasets, reshaping data structures, and generating group-level statistics.

Unit 1. NumPy Essentials:

NumPy ndarray: A Multidimensional Array Object, Creating ndarrays, Data Types for ndarrays, Arithmetic with Arrays, Basic Indexing and Slicing, Boolean Indexing, Fancy Indexing, Transposing Arrays, Swapping Axes, Universal Functions: Element-wise Operations, Basic Mathematical and Statistical Functions, Random Number Generation (basic use)

Unit 2. Pandas Basics and Data Structures:

Series, DataFrame, Index objects, Indexing and Selection, Filtering and Boolean Indexing, Arithmetic and Data Alignment, Sorting and Ranking, Dropping Entries, Handling Duplicate Indexes

Unit 3. Data Input, Output, and Cleaning:

Reading and Writing Data in Text Format (CSV, TXT), Working with JSON, Reading Microsoft Excel Files, Handling Missing Data, Dropping and Filling Missing Values, Replacing Values, Renaming Axis Indexes, Removing Duplicates, Filtering Outliers, Transforming Data Using Mapping or Functions

Unit 4. String Operations and Feature Engineering:
String Methods in pandas, Basic Regular Expressions, Vectorized String Functions, Creating Dummy/Indicator Variables, Permutation and Random Sampling.

Unit 5. Data Wrangling and Reshaping:
Merging and Joining Datasets, Concatenating Along an Axis, Combining Data with Overlap, Reshaping with Pivot, Stack, and Unstack, Basic Hierarchical Indexing, Summary Statistics by Group or Level

Textbooks

1. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter, Wes McKinney, 3rd Edition, O'Reilly Media, 2022.
2. Python for Data Science For Dummies, Yuli Vasiliev, 2nd Edition, Wiley, 2022.

Reference Books

1. Python Data Science Handbook: Essential Tools for Working with Data, Jake VanderPlas, 2nd Edition, O'Reilly, 2022.

2. Introduction to Machine Learning with Python, Andreas Müller & Sarah Guido, O'Reilly Media, Reprint Edition, 2023.

3. Foundations for Analytics with Python: From Non-programmer to Hacker, Clinton Brownley, 2nd Edition, Pearson, 2020.

Activities:

Outcome: Demonstrate proficiency in creating and manipulating NumPy arrays for mathematical operations and simulations.

Activity: Create a NumPy-based simulation:

- Generate a 2D array representing temperature data over time
- Apply mathematical operations (mean, std, element-wise addition)
- Simulate random noise and visualize the effect

Evaluation Method: Code-based assessment (10-point scale):

- Correct use of np.array, np.random, and math functions
- Accuracy of simulation logic
- Output clarity and reproducibility

Outcome: Apply Pandas Series and DataFrame operations for structured data handling and analysis.

Activity: Analyze a CSV dataset (e.g., sales or COVID data):

- Load into a DataFrame
- Perform Series operations (filter, map, value_counts)

- Apply DataFrame methods (groupby, sort, describe)

Evaluation Method: 10-point scale checklist and peer review to verify:

- Proper use of Series vs DataFrame
- Logical data manipulation
- Insightful summary statistics

Outcome: Read, write, and clean diverse data formats using Python tools, addressing missing values and outliers.

Activity: Work with multiple formats:

- Read CSV, Excel, and JSON files
- Identify and handle missing values (dropna, fillna)
- Detect and treat outliers using IQR or Z-score

Evaluation Method: Rubric-based evaluation to check (10-point scale):

- File handling accuracy
- Cleaning completeness
- Outlier detection logic

Outcome: Implement vectorized string operations and create derived features for enhanced model readiness.

Activity: Prepare text data for modelling to:

- Use str methods to clean and standardize strings
- Extract features (e.g., domain from email, length of name)
- Encode categorical variables (e.g., get_dummies, LabelEncoder)

Evaluation Method: Feature report to check (10-point scale):

- Efficiency of vectorized operations
- Relevance of derived features
- Readiness for ML input

Outcome: Perform complex data wrangling tasks such as merging datasets, reshaping data structures, and generating group-level statistics.

Activity: Integrate and reshape datasets:

- Merge two datasets on a common key
- Reshape using pivot, melt, stack, unstack
- Generate group-level stats (e.g., mean sales per region)

Evaluation Method: Before-and-after comparison to validate:

- Accuracy of merge and reshape
- Correct use of aggregation
- Final structure suitability for analysis

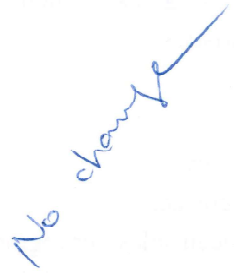
SEMESTER-V

COURSE 13 B: PYTHON FOR DATA SCIENCE

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

List of Practicals:

1. Create and Manipulate NumPy ndarrays; Explore Data Types
2. Perform Arithmetic Operations and Element-wise Calculations on Arrays
3. Practice Indexing, Slicing, Boolean, and Fancy Indexing on ndarrays
4. Use Universal Functions and Compute Basic Mathematical/Statistical Functions with NumPy
5. Create and Manipulate Pandas Series and DataFrames
6. Perform Indexing, Selection, Filtering, and Boolean Indexing in Pandas
7. Conduct Arithmetic Operations and Data Alignment in DataFrames
8. Sort, Rank, Drop Entries and Handle Duplicate Indexes in Pandas
9. Read and Write Data in CSV, TXT, JSON, and Excel Formats
10. Handle Missing Data: Detect, Drop, Fill, and Replace Missing Values
11. Rename Axis Indexes, Remove Duplicates, and Filter Outliers
12. Transform Data Using Mapping Functions and Apply String Operations
13. Perform String Operations and Use Regular Expressions on DataFrames
14. Create Dummy Variables and Perform Permutations and Random Sampling
15. Merge, Join, and Concatenate Datasets Using Pandas
16. Reshape Data Using Pivot, Stack, Unstack, and Perform Hierarchical Indexing
17. Compute Summary Statistics Grouped by Levels or Categories



No change

SEMESTER-VI

COURSE 14 A: MOBILE APPLICATION DEVELOPMENT

Theory | **Credits: 3** | **3 hrs/week**

Course Objectives

1. Understand core concepts of mobile app development and differentiate between native and cross-platform approaches.
2. Set up the Flutter and Dart development environment and apply foundational Dart programming constructs.
3. Design interactive and responsive UIs using Flutter widgets and implement custom design elements.
4. Develop multi-screen applications with effective state management and navigation techniques.
5. Integrate external data through APIs, manage local storage, and incorporate Firebase for cloud functionality.
6. Utilize advanced Flutter features, optimize performance, and deploy apps to the Google Play Store.

Course Outcomes

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

1. Configure the Flutter SDK and development tools, and write Dart programs using object-oriented principles and error-handling mechanisms.
2. Create visually consistent and interactive user interfaces using built-in Flutter widgets and custom styling.
3. Implement navigation between screens and manage application state using Provider and other built-in mechanisms.
4. Consume RESTful APIs, parse JSON data, and display structured content using dynamic UI components like ListView and GridView.
5. Incorporate device-level features (camera, location), apply animations, and package Flutter apps for deployment on Android platforms.

Unit 1. Introduction to Flutter and Dart:

Overview of mobile app development trends, Native vs. cross-platform development, Introduction to Flutter SDK and its architecture, Setting up Flutter & Dart environment (IDE, emulator, device) (Install Flutter SDK and set up IDE (VS Code / Android Studio), Dart syntax: variables, data types, control structures, Functions, classes, and object-oriented principles in Dart, Error handling and assertions.

Unit 2. Flutter Widgets and UI Design:
Stateless vs Stateful widgets, Layout widgets: Container, Row, Column, Stack, Input & selection widgets: TextField, Checkbox, Radio, Switch, Styling widgets: Padding, Margin, Fonts, Colors, Custom widgets and theming

Unit 3. Navigation and State Management:
Navigation: Navigator, routes, passing data between screens,

State management: setState, InheritedWidget, Provider, Dialogs, alerts, and snackbars, Forms and validation

Unit 4. Working with Data and APIs:
HTTP package for API calls, JSON parsing and model classes, Displaying data with ListView and GridView, Local storage: SharedPreferences, File handling, Firebase integration (basic setup & Firestore)

Unit 5. Advanced Features and Deployment:
Animations and custom transitions, Accessing device features: Camera, Location, Introduction to packages and plugins, Debugging and performance optimization, Building and releasing apps to Google Play Store

Textbooks:

1. Beginning Flutter: A Hands-On Guide to App Development, Marco L. Napoli, Wiley
2. Flutter for Beginners, Alessandro Biessek, Packt Publishing, 2nd Edition

Reference Books:

1. Flutter for Mobile Apps: Miguel Farmer, Rafael Sanders, Lincoln Publishers
2. Flutter Development Masterclass, E.M. Redwood, 2025

Activities:

Outcome: Configure the Flutter SDK and development tools and write Dart programs using object-oriented principles and error-handling mechanisms.

Activity: Set up Flutter SDK and create a Dart console app that:

- Uses object-oriented principles (classes, inheritance)
- Includes error-handling (try-catch-finally)
- Demonstrates basic input/output

Evaluation Method: Evaluate on a 10-point scale based on a checklist to verify:

- SDK and IDE properly configured
- Correct use of classes and inheritance
- Functional error-handling logic
- Output correctness and code readability

Outcome: Create visually consistent and interactive user interfaces using built-in Flutter widgets and custom styling.

Activity: Build a login screen using:

- Built-in widgets (TextField, Button, Column, etc.)
- Custom styling (colors, fonts, padding)
- Responsive layout

Evaluation Method: Rubric-based assessment on a 10-point scale:

- UI consistency and alignment
- Use of appropriate widgets
- Styling customization
- Responsiveness across screen sizes

Outcome: Implement navigation between screens and manage application state using Provider and other built-in mechanisms.

Activity: Create a multi-screen app with:

- Home, Profile, and Settings screens
- Navigation using Navigator
- State management using Provider (e.g., toggle dark mode)

Evaluation Method: Functional testing (10-point scale):

- Smooth navigation between screens
- Correct state updates via Provider
- Code structure and separation of concerns
- UI reflects state changes

Outcome: Consume RESTful APIs, parse JSON data, and display structured content using dynamic UI components like ListView and GridView.

Activity: Build a news app that:

- Fetches articles from a public REST API
- Parses JSON response
- Displays data in ListView and GridView

Evaluation Method: Live demo and code inspection (10-point scale):

- API integration and error handling
- JSON parsing accuracy
- Dynamic UI rendering
- Performance and responsiveness

Outcome: Incorporate device-level features (camera, location), apply animations, and package Flutter apps for deployment on Android platforms.

Activity: Create a photo journal app that:

- Captures images using device camera

- Retrieves current location
- Applies basic animations (e.g., fade-in)
- Packages and runs on Android device

Evaluation Method: Device-based testing to check (10-point scale):

- Camera and location permissions handled
- Feature functionality verified
- Animation smoothness
- Successful APK build and installation

SEMESTER-VI

COURSE 14 A: MOBILE APPLICATION DEVELOPMENT

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

List of Experiments:

1. Write Basic Dart Programs Using Variables, Functions, and Classes
2. Design a UI Layout Using Container, Row, Column, and Stack Widgets
3. Create an Interactive Form Using TextField, Checkbox, Radio, and Switch
4. Implement Custom Widgets and Apply Theming (Colors, Fonts, Styles)
5. Navigate Between Screens and Pass Data Using Navigator and Routes
6. Manage State Using setState and Provider Package
7. Create and Validate a Registration Form Using TextFormField and Validators
8. Fetch and Display JSON Data from a Public API Using HTTP Package
9. Parse JSON into Model Classes and Display Using ListView
10. Use SharedPreferences to Store and Retrieve Local Data
11. Integrate Firebase Firestore: Add and Retrieve Data
12. Implement Basic Animations Using AnimatedContainer and Hero Widgets
13. Access Device Camera or Location Using Flutter Plugins
14. Debug and Optimize Flutter Apps Using DevTools

SEMESTER-VI

COURSE 14 B: DATA VISUALIZATION TOOLS

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

- Understand the foundations and effective principles of data visualization.
- Learn to build visualizations using Python's ecosystem.
- Gain proficiency in R's ggplot2 for rich and layered data visualization.
- Understand Tableau for building intuitive, shareable dashboards.
- Apply tools in real-world data scenarios for exploration and insight.

Course Outcomes:

At the end of the course the Students will be able to:

- Analyze and critique data visualizations for effectiveness and ethical integrity.
- Develop static and interactive plots to analyze datasets.
- Construct customized visualizations with ggplot2.
- Create interactive visual stories using Tableau.
- Demonstrate complete visualization workflows from raw data to insight.

Unit 1. Fundamentals of Data Visualization:
Importance of visualization in data science,Types of data (categorical, numerical, time-series, geospatial), Chart types: bar, line, histogram, box plot, heatmaps, maps, Design principles: clarity, simplicity, integrity, aesthetics, Misleading visualizations and ethical considerations

Unit 2. Data Visualization Using Python:
Introduction to matplotlib: plots, customization, styling, Seaborn for statistical data , visualization, Plotly for interactive charts and dashboards, Advanced plots: pairplots, violin plots, heatmaps, time-series plots, Integrating visualizations in Jupyter and web apps

Unit 3. Data Visualization Using R (ggplot2):
Introduction to ggplot2 grammar of graphics,Aesthetics, geometries, scales, themes,Faceting and layering techniques,Visualizing categorical and numerical data, Customizing and exporting plots

Unit 4. Basics of Tableau:
Tableau basics: interface, data connection, Charts, filters,

Aggregation, Calculated values and table calculations, Using the calculation dialog box to create, Building formulas using table calculations, Using table calculation functions.

Unit 5. Interactive Dashboards with Tableau:
Maps, tooltips, and trendlines, generating new data with forecasts, Storytelling with dashboards, providing self evidence adhoc analysis with parameters, Editing views in tableau Server, Publishing and sharing reports

Text Books

1. Data Visualization: A Practical Introduction, Kieran Healy, Princeton University Press, 2019.
2. Learning Tableau 2022, Ben Jones, Packt Publishing, 2022.
3. Python Data Science Handbook, Jake VanderPlas, 2nd Edition, O'Reilly, 2022.

Reference Books

1. Fundamentals of Data Visualization, Claus Wilke, O'Reilly, 2019.
2. Better Data Visualizations, Jonathan Schwabish, Columbia University Press, 2021.
3. ggplot2: Elegant Graphics for Data Analysis, Hadley Wickham, Springer, 3rd Edition, 2023.
4. Tableau eLearning & Public Gallery Resources (https://public.tableau.com)

Activities:

CO1: Analyze and critique data visualizations for effectiveness and ethical integrity

Activity:
Conduct a **visualization audit and critique assignment**. Students select a set of published visualizations (from news, dashboards, or social media) and analyze their design effectiveness, ethical considerations (e.g., misleading visuals, omitted context), and audience impact.

Evaluation Method:

Rubric-based assessment for depth of critique, identification of ethical flaws, clarity of reasoning, and suggestions for improvement - scored on a 10-point scale.

CO2: Develop static and interactive plots to analyze datasets

Activity:
Interactive lab + peer showcase - students create a combination of static (e.g., matplotlib/seaborn) and interactive (e.g., Plotly) plots for a real dataset. Then, participate in a **gallery walk** where students view each other's work and provide structured peer feedback using a guided rubric.

Evaluation Method:

Rubric includes technical accuracy, interactivity, design quality, peer feedback contribution, and clarity of explanation - scored out of 10.

CO3: Construct customized visualizations with ggplot2

Activity:

Mini project in R using ggplot2 - students work on a domain-based dataset (e.g., environmental, healthcare, or education) and apply advanced ggplot2 features like custom themes, annotations, faceting, scales, and coordinate systems to convey insights.

Evaluation Method:

Rubric-based code review and visualization quality evaluation (design, customization, clarity of message) - scored on a 10-point scale.

CO4: Create interactive visual stories using Tableau

Activity:

Storyboarding + Tableau design - students first sketch a **storyboard** outlining the flow of their data story (with objectives, key visuals, and user interaction paths), and then implement the story in Tableau using dashboards and interactive filters.

Evaluation Method:

Rubric evaluates storyboard planning, narrative flow, interactivity, design consistency, and insightfulness - scored on a 10-point scale.

CO5: Demonstrate complete visualization workflows from raw data to insight

Activity:

Capstone visualization project - students select a dataset of interest, perform data cleaning, transformation, EDA, and build a complete multi-tool workflow (e.g., preprocessing in Python, visualization in Tableau/R). They then **present their insights in a recorded video or classroom seminar.**

Evaluation Method:

Rubric includes workflow completeness, tool integration, clarity of insights, storytelling effectiveness, and presentation quality - evaluated on a 10-point scale.

SEMESTER-VI

COURSE 14 B: DATA VISUALIZATION TOOLS

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

List of Practicals

1. Use matplotlib to generate line, bar, pie, and scatter graphs.
2. Use Seaborn on statistical data visualization.
3. Implement interactive charts and dashboards using Plotly.
4. Generate pairplots, violin plots, heatmaps, time-series plots for data visualization.
5. Generate scatter graph , bar graph, and histogram using ggplot2.
6. Visualize categorical and numerical data using R.
7. On a sample data set(eg., Supermarket / Showroom spread across the states),
    a. Create visualizations
        i. Bar chart showing Sales by Region.
        ii. Line chart of Profit over time.
        iii. Add filters (e.g., by Category or Year).
    b. Aggregations
        i. Use SUM, AVG, and COUNT aggregations.
        ii. Show totals and subtotals on the chart.
    c. Deliverables
        i. Dashboard with at least two charts and interactive filters.
8. Implement the following inTableau
    a. Create Calculated fields like profit ratio
    b. Use Table Calculation functions like RUNNING_SUM(), WINDOW_AVG(), or RANK().
    c. Worksheet showing
        i. Calculated fields in use
        ii. Table calculation in a visualization (e.g., trend chart)
        iii. Proper labels and tooltips
9. Mini-project: Create dashboard for Semester Results Analysis using Python or R.

SEMESTER-VI

COURSE 15 A: MERN STACK

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. Understand full-stack architecture and the individual roles of the MERN components-MongoDB, Express.js, React.js, and Node.js.
2. Develop interactive front-end applications using React.js and master state management and routing techniques.
3. Model and manipulate NoSQL databases using MongoDB and Mongoose, including CRUD operations and schema validations.
4. Build RESTful APIs using Express.js and integrate server-side logic with frontend and database components.
5. Implement full-stack application features such as authentication, session management, and deployment using modern platforms.

Course Outcomes:

1. Explain the architecture of the MERN stack and configure a Node.js development environment with essential modules and server setup.
2. Develop dynamic user interfaces using React functional components, hooks, forms, and routing for seamless user experiences.
3. Perform database operations using MongoDB and Mongoose, including schema design, data validation, and relational mapping.
4. Construct RESTful backend services using Express.js with routing, middleware, and error handling mechanisms.
5. Integrate frontend and backend components with secure data flow, apply JWT-based authentication, and deploy applications on platforms like Render, Netlify, or Vercel.

Unit 1. Introduction to MERN Stack & Node.js:

Introduction to Full Stack Web Development, Frontend vs Backend, What is the MERN stack? Architecture of MERN Applications

Introduction to Node.js, Installing Node.js & npm, Node.js fundamentals (Modules, Events, Streams), Asynchronous Programming & Event Loop, Node.js File System module, npm modules & custom modules, Setting up a basic server with Node.js

Unit 2. React.js (Frontend Framework):

Introduction to React, Functional Components & JSX, State & Props, Handling Events, useState, useEffect Hooks, Conditional Rendering & Lists, React Router (Routing), Forms in React (Controlled Components), Axios for HTTP requests

Unit 3. MongoDB with Mongoose:
Introduction to NoSQL Databases, MongoDB vs SQL Databases, Installing & using MongoDB (local & Atlas), CRUD Operations with MongoDB Shell & Compass, Mongoose ODM, Models, Schemas, Validation, Relationships (One-to-Many, Many-to-Many), Aggregation

Unit 4. Express.js (Backend Framework):
Introduction to Express.js, Creating RESTful APIs using Express, Routing (GET, POST, PUT, DELETE), Middleware in Express, Error Handling, Connecting to MongoDB using Mongoose, Environment variables and `.env` files

Unit 5. Full Stack Integration & Deployment:
Connecting Frontend (React) with Backend (Express), CORS and Proxy setup, Authentication with JWT, Protected Routes in Frontend, Role-based access control, Deployment: Deploying backend to Render/Heroku, Deploying frontend to Netlify/Vercel, Connecting MongoDB Atlas

Textbooks:

1. Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node, Subramanian, Vasan, 2nd Edition, Apress,
2. Learning React: Functional Web Development with React and Redux, Alex Banks & Eve Porcello, O'Reilly
3. MongoDB: The Definitive Guide, 3rd Edition, Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, O'Reilly

Reference Books:

1. Ultimate Full-Stack Web Development with MERN, Nabendu Biswas, Orange Education Pvt Ltd.
2. Full Stack Development with MERN, Thompson Carter, Lincoln Publishers

Activities:

Outcome: Explain the architecture of the MERN stack and configure a Node.js development environment with essential modules and server setup.

Activity: Set up a basic Node.js project with Express and required modules (express, nodemon, dotenv). Create a simple server that responds with Hello MERN Stack on a browser.

Evaluation Method: Checklist-based review (10-point scale):

- Correct installation of Node.js and modules
- Functional server response
- Use of environment variables
- Folder structure and code clarity

Outcome: Develop dynamic user interfaces using React functional components, hooks, forms, and routing for seamless user experiences.

Activity: Build a multi-page React app (e.g., user profile and contact form) using:

- Functional components
- useState and useEffect hooks
- Controlled form inputs

- React Router for navigation

Evaluation Method: Live demo and rubric (10-point scale):

- Component structure and reusability
- Hook usage and state management
- Form validation and routing functionality
- UI responsiveness and layout

Outcome: Perform database operations using MongoDB and Mongoose, including schema design, data validation, and relational mapping.

Activity: Create a student database using MongoDB and Mongoose. Implement:

- Schema with validation rules
- CRUD operations (Create, Read, Update, Delete)
- Reference between collections (e.g., student and course)

Evaluation Method: Code walkthrough and test cases (10-point scale):

- Schema correctness and validation
- CRUD functionality
- Relational mapping using ref
- Console output and error handling

Outcome: Construct RESTful backend services using Express.js with routing, middleware, and error handling mechanisms.

Activity: Develop a REST API for a task manager app using Express.js. Include:

- Routes for task operations
- Middleware for logging and JSON parsing
- Error handling for invalid routes

Evaluation Method: API testing with Postman(10-point scale):

- Route functionality and status codes
- Middleware implementation
- Error response structure
- Code readability and modularity

Outcome: Integrate frontend and backend components with secure data flow, apply JWT-based authentication, and deploy applications on platforms like Render, Netlify, or Vercel.

Activity: Build a login system with:

- JWT-based authentication
- Protected routes
- Frontend-backend integration
- Deploy frontend on Netlify/Vercel and backend on Render

Evaluation Method: Deployment demo and checklist (10-point scale):

- Token generation and validation
- Secure data flow between client and server
- Working login/logout flow
- Successful deployment and accessibility

SEMESTER-VI

COURSE 15 A: MERN STACK

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

List of Experiments:

1. Install Node.js and npm; Create and Use Built-in & Custom Node.js Modules
2. Demonstrate Asynchronous Programming Using Callbacks and Promises
3. Implement File Read/Write Operations Using Node.js File System Module
4. Create a React App and Build Functional Components Using JSX
5. Handle Events and Use useState & useEffect Hooks in React
6. Implement Navigation Between Pages Using React Router
7. Create and Validate a Form Using Controlled Components in React
8. Install and Connect to MongoDB (Local and MongoDB Atlas)
9. Perform CRUD Operations Using MongoDB Shell and MongoDB Compass
10. Create Mongoose Schemas and Models, and Connect Them to a Node.js App
11. Create a RESTful API Using Express.js (GET, POST, PUT, DELETE)
12. Use Middleware and Error Handling in an Express App
13. Connect Express App to MongoDB Using Mongoose
14. Implement User Authentication Using JWT in Express and React
15. Create Protected Routes and Role-Based Access Control in React
16. Deploy Backend to Render/Heroku and Frontend to Netlify/Vercel

SEMESTER-VI

COURSE 15 B: MACHINE LEARNING

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. Understand fundamental concepts, types, and applications of machine learning.
2. Develop, evaluate, and optimize machine learning models through preprocessing, training, and feature engineering techniques.
3. Apply supervised and unsupervised learning algorithms to real-world problems using appropriate tools and methods.

Course Outcomes:

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

1. Describe various machine learning paradigms, data types, and the overall structure of a machine learning pipeline.
2. Perform data preprocessing, feature engineering, and evaluate models using appropriate metrics.
3. Implement and analyze supervised learning algorithms for regression and classification tasks.
4. Apply unsupervised learning techniques for clustering and identify suitable machine learning approaches for specific application domains

Unit 1. Introduction to Machine Learning:

Introduction to Machine Learning: Types of human learning, What is machine learning?, Types of machine learning: supervised, unsupervised, semi-supervised and reinforcement learning, machine learning activities, applications of machine learning. Types of data in machine learning, Structure of data

Unit 2. Model Preparation, Evaluation and feature engineering:

Data pre-processing, Model selection and training (for supervised learning), Model representation and interpretability, Evaluating machine learning algorithms and performance enhancement of models. What is feature engineering?, Feature transformation, Feature subset selection. Principal component analysis.

Unit 3. Supervised Learning-Regression:

Regression: Introduction of regression, Regression algorithms: Simple linear regression, Multiple linear regression, Polynomial regression model, Logistic regression, Maximum likelihood estimation.

Unit 4. Supervised Learning- Classification:
Introduction of supervised learning, Classification model and learning steps, Classification algorithms: Naïve Bayes classifier, k-Nearest Neighbour (kNN), Decision tree, Support vector machines, Random Forest.

Unit 5. Unsupervised Learning:
Introduction of unsupervised learning, Unsupervised vs supervised learning, Application of unsupervised learning, Clustering and its types, Partitioning method: k-Means and KMedoids, Hierarchical clustering, Density-based methods – DBSCAN.

Case-study of ML applications: Image recognition, speech recognition, Email spam filtering, Online fraud detection and other.

Textbooks:

1. Introduction to Machine Learning, Ethem Alpaydin, MIT Press, Fourth Edition, 2020.
2. Machine Learning: Theory and Practice, M N Murthy, V.S Ananthanarayana, Universities press
3. Machine Learning, S. Sridhar, M. Vijayalakshmi, Second Edition, Oxford University Press

Reference Books:

1. Machine Learning: An Algorithmic Perspective, Second Edition, Stephen Marsland, CRC Press, 2014
2. Machine Learning, Tom Mitchell, McGraw Hill, 3rd Edition.
3. Python Machine Learning, Sebastain Raschka, Vahid Mirjalili , Packt publishing 3rd Edition, 2019.

Activities:

Outcome: Describe various machine learning paradigms, data types, and the overall structure of a machine learning pipeline.

Activity: Prepare a detailed comparative report/chart explaining supervised, unsupervised, and reinforcement learning paradigms, data types, and step-by-step machine learning pipeline stages.
Evaluation Method: Rubric-based assessment evaluating completeness, clarity, correctness, and presentation quality - scored on a 10-point scale.

Outcome: Perform data preprocessing, feature engineering, and evaluate models using appropriate metrics.

Activity: Conduct a hands-on lab exercise using a real dataset to perform data cleaning, normalization, feature extraction/selection, and evaluate model performance using metrics like accuracy, precision, recall, and F1-score.

Evaluation Method: Practical assessment including code correctness, applied techniques, and interpretation of evaluation metrics; assessed with a rubric out of 10.

Outcome: Implement and analyze supervised learning algorithms for regression and classification tasks.

Activity: Implement at least two supervised learning algorithms (e.g., Linear Regression and Decision Trees) to solve prediction tasks, followed by comparative analysis of their performance on test datasets.

Evaluation Method: Code and report evaluation focusing on implementation accuracy, performance comparison, and analysis depth; scored on a 10-point rubric.

Outcome: Apply unsupervised learning techniques for clustering and identify suitable machine learning approaches for specific application domains.

Activity: Perform clustering (e.g., K-Means, Hierarchical) on a given dataset and prepare a case study selecting and justifying machine learning methods suited for different application scenarios.
Evaluation Method: Lab practical combined with a written case study; assessed for correct algorithm application, cluster interpretation, and justification of approach - evaluated on a 10-point rubric.

SEMESTER-VI

COURSE 15 B: MACHINE LEARNING

Practical	Credits: 1	2 hrs/week

Lab Experiments:

1. Write a python program to import and export data using Pandas library functions.
2. Demonstrate various data pre-processing techniques for a given dataset
3. Implement Dimensionality reduction using the Principal Component Analysis (PCA) method.
4. Write a Python program to demonstrate various Data Visualization Techniques.
5. Implement MLE on a Dataset
6. Implement Simple and Multiple Linear Regression Models.
7. Develop Logistic Regression Model for a given dataset.
8. Develop Decision Tree Classification model for a given dataset and use it to classify a new sample.
9. Implement Naïve Bayes Classification in Python.
10. Develop K-Means for a Given Dataset
11. Build KNN Classification model for a given dataset.
12. Develop DBSCAN on a given Dataset