COURSE STRUCTURE AND SYLLABI

Bachelor of Computer Application (BCA)

2025-26 Batch

Centurion UNIVERSITY

Shaping Lives...
Empowering Communities...

SCHOOL OF ENGINEERING AND TECHNOLOGY
CENTURION UNIVERSITY OF TECHNOLOGY & MANAGEMENT
Odisha-761211, India

Web Site: - www.cutm.ac.in

CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT, ODISHA

CERTIFICATE

Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

This is to certify that the syllabus of the Programme Bachelor of Computer Application (BCA) of the School of Engineering and Technology is approved in the 15th Academic Council Meeting held on 22nd November 2025.

Dean
School of Engineering and Technology,
CUTM, Odisha

Pro Vice Chancellor
CUTM, Odisha

Centurion University of Technology and Management Odisha

Batchelor of Computer Application (BCA)

CHOICE BASED CREDIT SYSTEM

COURSE STRUCTURE & SYLLABUS

BASKET – I, II, III, IV & V

Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

School of Engineering & Technology

2025-26

Index

Programme Objectives; Job/Higher studies/Entrepreneurship

POs: Graduates will be able to;

PO	Outcomes
PO1	Engineering knowledge: Apply knowledge of mathematics, science, engineering fundamentals, and Computer Applications to the solution of engineering problems
PO2	Problem analysis: Identify, formulate, review the literature and analyze Computer Application problems to design, conduct experiments, analyze data and interpret data
PO3	Design /development of solutions: Design solutions for Computer Application problems and design system components or processes that meet the desired needs with appropriate consideration for public health and safety, and the cultural, societal and environmental considerations
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions in Computer Application
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to Computer Application activities with an understanding of the limitations
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to Computer Application practice
PO7	Environment and sustainability: Understand the impact of the Computer Application solutions in societal and environmental contexts, and demonstrate the knowledge and need for sustainable development
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the Computer Application practice
PO9	Individual and team work: Function affectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in Computer Application
PO10	Communication: Communicate effectively on complex engineering activities with the engineering committee and with society at large, such as, being able to

	comprehend and write affective reports and design documentation, make effective presentations in Computer Application
PO11	Project Management and finance: Demonstrate knowledge & understanding of the Computer Application principles and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in multidisciplinary environments in Computer Application
PO12	Life-long learning: Recognize the need for, and the preparation and ability to engage in independent research and lifelong learning in the broadest contest of technological changes in Computer Applications

PEOs/PSOs

PEO1: Prepare students to build competency in current technology and its application to meet the industry need for skilled Engineer

PEO2: Provide students with strong foundational concepts and also domain knowledge to pursue research to build solutions or systems of varying complexity to solve the problems identified

PEO3: Enable graduates to innovate, bring new ideas and become an entrepreneur

PSO1: The graduate will be able to work on high-end technology in the IT Services industries.

PSO2: Graduate can acquire the industry-certified level of competency and work on real-time IT application projects viz; Health/Agriculture/Security/Data Management etc.

PSO3: Graduate can start their own IT service company to provide technical solutions Course Outcomes Attributes

Course Outcomes	**Attributes**
CO1	Knowledge
CO2	Analytical skills and Critical Thinking
CO3	Problem Solving and Decision taking ability
CO4	Use of Tools, Design and Development (Hands-on/Technical skill)
CO5	Research, Ethics & Teamwork

Basket Structure - Undergraduate Study			
Basket	**Type of Course**	**Minimum Credit Required for UG**	
		4 Years	**3 Years**
I	**Major (Core) Courses**	**80**	**60**
II	**Minor / Domain (Multi-Disciplinary)**	**44**	**36**
III	**Value Added + AEC + SEC**	**20**	**20**
IV	**Summer Internship / Community Engagement**	**4**	**4**
V	**Research Project/ Dissertation/Production Action Learning**	**12**	**0**
	TOTAL	**160**	**120**

Basket II – 16 Credit Minor + 12-22 Credit domain + 4 EVS + student may choose skill course, minor subject, MOOCs, NPTEL to fulfilling the basket requirement.

Basket III – 2 skill (8) + Job Readiness (6) + values added (6)

Major (Core) Courses					
Sl No	**Subject Code**	**Subject Name**	**Type (PP+PR+TUT)**	**Credits**	**NcRF Level**
1	CUBC1001	Fundamentals of Computer	2+2+0	4	4.5
2	CUBC1002	OOPs using C++	1+2+1	4	4.5
3	CUBC1003	Office Automation	2+2+0	4	4.5
4	CUBC1004	Data Structures using C++	1+3+0	4	4.5
5	CUBC1005	Database Management Systems	2+2+0	4	4.5
6	CUBC1006	Java Programming	1+2+1	4	4.5
7	CUBC1007	Internet and Web Technology	1+2+1	4	5
8	CUBC1008	Python Programming	1+2+1	4	5
9	CUBC1009	Operating System Concepts	2+2+0	4	5
10	CUBC1010	Dot Net Technology	1+2+1	4	6
11	CUBC1011	Fundamentals of Algorithm Design and Analysis	2+2+0	4	5
12	CUBC1012	Computer Communication and Networking	2+2+0	4	5
13	CUBC1013	Introduction to Software Engineering	2+2+0	4	5
14	CUBC1014	Computer System Architecture	2+2+0	4	5.5
15	CUBC1015	Android App Development	1+2+1	4	5.5
16	CUBC1016	Information Security	2+1+1	4	6
17	CUBC1017	Cloud Computing	2+2+0	4	6
18	CUBC1018	Internet of Things	1+2+1	4	6
19	CUTM1019	Machine Learning using Python	1+2+1	4	5
20	CUBC1019	Major Project	0+0+4	4	
	Total			**80**	

Value Added Courses					
Sl No	**Subject Code**	**Subject Name**	**Type (PP+PR+TUT)**	**Credits**	**NcRF Level**
1	CUTM1674	Environmental Studies	3+0+1	4	4.5
2	CUVA4064	Universal Human Values	2+1+0	3	4.5
3	CUVA4060	GENDER, HUMAN RIGHTS AND ETHICS	3+0+0	3	4.5
4	CUVA4063	Introduction to Indian Knowledge Systems	2+0+1	3	4.5

Ability Enhancement Courses					
Sl No	**Subject Code**	**Subject Name**	**Type (PP+PR+TUT)**	**Credits**	
1	CUTM1016	Job Readiness	0+0+6	6	

Minor / Domain Courses					
Sl No	**Subject Code**	**Subject Name**	**Type (PP+PR+TUT)**	**Credits**	
1	FDCU1000	Full-Stack Development with MERN	18	1+8+9	
2	GACU1010	Generative AI	12	0+8+4	
3	MLCU1020	Data Analytics and Machine Learning	18	0+6+12	
4	CTCU1030	Cloud Technology	12	0+6+6	
5	DSCU1040	Drone Imaging and Spectral Analysis	12	0+6+6	
6	STCU1050	Software Technology	18	0+6+12	
7	MACU1070	Mobile App Development	12	1+6+5	
8	GICU1080	Gaming and Immersive Learning-AR/VR	20	5+5+10	
9	BDCU1090	Blockchain Development	18	0+7+11	
10	CSCU1100	Cyber Security	20	8+8+4	

Course Outline (Basket-I)

Fundamentals of Computers

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1001	Fundamentals of Computers	2+2+0	Nil

Course Objectives

- Identify the function of computer hardware components.
- Understand factors influencing individual or organizational decisions on computer equipment purchases.
- Learn how to maintain computer equipment and troubleshoot hardware issues.

Course Outcomes

- CO1: Remember the fundamental hardware components of a computer and their roles (Knowledge).
- CO2: Understand the difference between operating systems and application programs and their respective uses (Comprehension).
- CO3: Apply basic binary arithmetic and number system conversions in computing tasks (Application).
- CO4: Analyze the architecture of processors and memory management systems in computer operations (Analysis).
- CO5: Evaluate the impact of different computer systems on everyday technological products and communication systems (Evaluation).

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2												
CO2	3	3	3	2	2								2	2	2
CO3	2		2										1	1	1
CO4	2	3	3	2									2	2	2
CO5	3	3	3	2	2								2	3	2

*High-3, Medium-2, Low-1

COURSE CONTENT

Module 1 (14 hours)

Theory

Computer Basics: A simple model of computers, Digital and Analog Computers, Evolution of digital computers, Major Components of a digital computer, hardware, software, firmware, middle ware, freeware. Input/output Devices: Input Devices, Output Devices, Printers, and Plotters, Other forms of Output devices, Input and Output port.

Practice

1.1.Identifying and classifying computer hardware components
1.2.Setting up a basic computer system
1.3.Using printers and plotters for document outputs
1.4.Configuring I/O ports for various peripherals
1.5.Troubleshooting common hardware issues

Module 2 (14 hours)

Theory

Number System: Decimal Number System, Binary Number System, conversion of numbers(binary to decimal, decimal to binary), Addition of Binary Numbers, 1's complement and 2's complement representation of numbers, Binary Subtraction, Binary Multiplication, Binary Division, Hexadecimal and octal number system, ASCII and ISCII code, EBCDIC code, Gray codes, Fixed point and Floating Point Representation, Overflow and Underflow.

Practice

2.1. Converting between binary and decimal systems
2.2.Performing binary addition and subtraction
2.3.Working with 1's and 2's complement
2.4.Converting binary to hexadecimal and octal
2.5.Solving real-world problems using binary math

Module 3 (14 hours)

Theory

Logic Circuits: Switching circuits, AND/OR operations, NOT operations, Boolean Functions, Canonical Forms of Boolean Functions, Logic circuits.

Practice

3.1.Designing simple logic circuits
3.2.Implementing AND/OR/NOT logic gates
3.3.Simplifying Boolean expressions
3.4.Creating truth tables for basic circuits
3.5.Verifying circuit operations using simulation software

Module 4 (14 hours):-

Theory

Processor: CPU organization, Structure of Instruction, A machine level language Computer Memory: Read only memory, Serial Access memory, Main memory, Secondary memory: Magnetic hard disk, Floppy disk Drives, Compact Disk Read Only memory, Magnetic tape Drives.

Practice

4.1.Analyzing CPU organization and instruction sets
4.2.Differentiating between various memory types
4.3.Configuring main memory and secondary storage devices
4.4.Understanding memory management techniques
4.5.Exploring machine-level programming with basic commands

Module 5 (14 hours)

Theory

Computer Architecture: Interconnection of units, Processor to memory communication, I/O to processor communication, Interrupt Structure, Multi programming, Processor Features, RISC, Virtual Memory.

Practice

5.1.Illustrating computer unit interconnections
5.2.Simulating I/O processor communication
5.3.Exploring interrupt handling mechanisms
5.4.Understanding multiprogramming
5.5.Implementing simple virtual memory management

Module 6 (14 hours)

Theory

Computer Languages: Programming Language, Introduction to Interpreter and compiler, Assembly Language, Higher Level Languages Operating System: Need of an OS, Batch operating system, multi programming Operating system, Time sharing Operating System, personal computer Operating system, on-line and real time system. Computers and Communications: Computer Generations.

Practice

6.1.Writing basic assembly language programs
6.2.Interpreting high-level language programs
6.3.Installing and configuring different operating systems
6.4.Simulating time-sharing in operating systems
6.5.Analyzing real-time system operations

Module 7 (14 hours)

Theory

Types of communications with and among computers, internet and World Wide Web, characteristics of communication channels, Physical Communication Model, Computer Network topologies, Local Area Network.

Practice

7.1.Configuring basic LAN networks
7.2.Analyzing network topologies

7.3.Setting up physical communication channels
7.4.Exploring internet protocols and WWW architecture
7.5.Troubleshooting network connectivity issues

Assignments: -

1. What are storage devices? Explain with examples?
2. Write the name of 5 internal and external commands of DOS?
3. What is the difference between Hardware and Software?
4. What are different network topologies?
5. What is Memory? Explain types of Memory in brief?
6. WAP in c to find factorial of a given number?
7. What is ROM?
8. What is primary memory?

Text Book:-

1.Fundamentals of computers, by V.Rajaraman (chapter 1,2,3,4,5,6,7,8,9,10,12,13)

Reference Books:

1. Computer Fundamentals, by B.Ram
2.Computer Fundamentals by P.K.Sinha
3. Fundamentals Of Information Technology, 2nd Edition, Alexis Leon, Mathew Leon,Vikas Publishing House Pvt Ltd.

OOPs using C ++

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1002	OOPs using C ++	1+2+1	NIL

Course Objectives

- Understand how C++ enhances procedural programming with object-oriented features such as encapsulation, inheritance, and polymorphism.
- Develop problem-solving skills using C++ by designing classes for code reuse, memory management, and efficient data manipulation.
- Apply exception handling, file manipulation, and polymorphism techniques to build robust and scalable programs

Course Outcomes

- CO1 : Recall the fundamental principles of object-oriented programming (OOP) in C++, such as classes, objects, and functions.
- CO2 : Explain how inheritance and polymorphism are used to achieve code reuse and dynamic binding (Comprehension).
- CO3 : Implement file handling techniques, virtual functions, and exception handling in C++ programs (Application).
- CO4 : Analyze complex problems and design appropriate class hierarchies and algorithms using object-oriented principles (Analysis).
- CO5 : Evaluate and debug C++ programs involving multiple classes, inheritance, and virtual functions to ensure efficiency and correctness (Evaluation).

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1										2	1	1
CO2	3	2	1										2	1	1
CO3	3	3	2			2							2	1	1
CO4	3	3	2	1		1							3	2	1
CO5	3	3	3	2		1							3	3	2

*High-3, Medium-2, Low-1

Course content

Module I: Basics of C++ (16 hrs)

Theory

Beginning with C++, Tokens, Static Members, Constant Members, Expressions, and Control Structure

Practice

1.1. Write a program to print your CV using cout.
1.2. Write a program to check if a number is prime.
1.3. Write a program to check if a number is palindrome.
1.4. Write a program to check if a number is magic or Armstrong.
1.5. Write a program to print patterns.

Module II: Basics of Object-oriented concepts (12)

Theory

Object-oriented concepts include Classes and Objects, Encapsulation, Abstraction, Overloading, Inheritance, and Polymorphism.
Functions: parameter passing, inline function, function overloading

Practice

1.1. Write a program to define a function without parameters.
1.2. Write a program to Implement function overloading for summing even and odd elements of an array.
1.3. Write a program to implement an inline function

Module II: Class-Object-Constructor (10 hrs)

Theory

Classes: data members, member function, array of objects, static data members, constant members function, and friend function.
Constructors, Encapsulating into an object, Destructors.

Practice

2.1. Define a class to represent a book in a library, including functions to issue and return books.
2.2. Implement a bank account class that handles deposits, withdrawals, and balance checking.
2.3. Create a class for handling fixed deposit accounts of 10 customers, including data management functions.
2.4. Use friend functions to add distances stored in different units (feet/inches and meters/centimeters).
2.5. Design and implement a simple class hierarchy with constructors and destructors.

Module IV: Inheritance (14 hrs)

Theory

Associations, Inner Classes, Memory Management and pointers
Inheritance: Derived classes, member accessibility, forms of inheritance, virtual base classes.

Practice

3.1. Design a program using inheritance to calculate areas of shapes (e.g., triangle, rectangle) using virtual functions.
3.2. Build a database for an educational institution using class hierarchies.
3.3. Implement multi-level inheritance with constructors and destructors for each class.
3.4. Write a program to simulate multiple inheritance in a real-world scenario.

3.5. An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in following figure. The figure also shows the minimum information required for each class. Specify all classes and define functions to create the database and retrieve individual information as and when required.

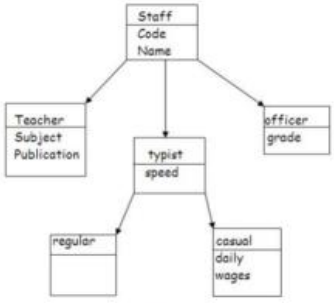

fig: class relationships (for exercise 8.3)

Module V: Polymorphism (14 hrs)

Theory

Polymorphism (Compile time Polymorphism, Run time Polymorphism), Virtual Functions, Abstract class, virtual destructors, Interfaces.

Practice

4.1. Write a program to overload the ++ operator for a custom class.
4.2. Overload the + operator to concatenate two strings.
4.3. Implement virtual functions to display different shapes using polymorphism.
4.4. Design a program that uses pure virtual functions to create an abstract class.
4.5. Write a program to handle runtime polymorphism using pointers and virtual destructors.

Module VI: Exception Handling (14 hrs)

Theory

Exception Handling, Managing Console I/O Operations, Streams & Files: streams, hierarchy of stream classes, working with files

Practice

5.1. Write a Program to describe about exception handling mechanism.
5.2. Write a Program to describe multi catch statement.
5.3. Write a program to read a list containing item name, item code, and cost interactively and produce a three column output as shown below.

Name	Code	Cost
Turbo C++	1001	250.95
C primer	905	95.70
.....	.....	.....
.....	.....	.....

Note that the name and code are left-justified and the cost is right justified with a precision of two digits. Trailing zeros are shown.

5.4. Write a program that reads a text file and creates another file that is identical except that every sequence of consecutive blank spaces is replaced by a single space.
5.5. Write a program that reads character from the keyboard one by one. All lower case characters get store inside the file LOWER, all upper case characters get stored inside the file UPPER and all other characters get stored inside OTHERS.

Module VII: Templates (14 hrs)

Theory

Advance Topics in C++ Object Design and Templates STL (Standard Type Libraries) RTTI (Run Time Type Identification) Advanced Typecasting, new data types, new operators, class implementation, namespace scope , operator keywords, new headers , C++ Containers

Practice

6.1. Write a function template to find the minimum value in an array.
6.2. Design a publication class for books and tapes, using inheritance and templates for data manipulation.
6.3. Create a program that uses STL containers for efficient data storage and retrieval.
6.4. Implement a namespace for organizing classes in a large project.
6.5. Write a template function to manage complex data types and perform type casting

Text Books:

1. E Balagurusamy, “Object Oriented Programming with C++”, Tata McGraw Hill, Sixth Edition.
2. Herbert Schlitz, “The Compete Reference C++”, Tata McGraw Hill, Fourth Edition.

Reference Books:

1. Ashok Kamthane, “Object Oriented Programming with ANSI and Turbo C++”, Pearson.
2. Behrouz A. Forouzan& Richard F. Gilberg “A Structured approach using C++” Cengage Learning Indian Edition.

Office Automation

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1003	**Office Automation**	2+2+0	NIL

Course Objectives

- Understand the basic features of Microsoft Office programs, including Word, Excel, and PowerPoint.
- Develop skills in document creation, data management, and presentation using Microsoft Office tools.
- Apply office automation techniques to perform tasks related to document formatting, data analysis, and multimedia presentations.

Course Outcomes

- **CO1 :** Recall basic commands and functionalities in Microsoft Word, Excel, and PowerPoint.
- **CO2:** Explain the purpose of various Office Automation tools and their appropriate use cases.
- **CO3 :**Use Microsoft Office programs to create professional and academic documents, spreadsheets, and presentations.
- **CO4 :** Analyze data using Excel functions, formulas, and charts to solve real-world problems.
- **CO5:** Evaluate the quality of digital documents and presentations by applying advanced formatting, automation, and multimedia techniques.

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	**2**	**3**	**2**	2											**1**
CO2	**2**	**3**	**3**	**2**	2								**2**	**2**	**1**
CO3	**2**	**3**	**3**	**2**	2	**2**							**3**	**3**	**2**
CO4	**2**	**3**	**3**	**2**		**2**							**3**	**3**	
CO5	**2**	**2**	**2**	2									**2**		**2**

*High-3, Medium-2, Low-1

Course content

Module I:introduction to ms word (14 hrs)

Theory

Introduction to Office Automation, Creating & Editing Document, Formatting Document, Auto text, Auto correct, Spelling and Grammar Tool, Document Dictionary.

Practice

1.1. Write a paragraph about your institution and modify font size, type, and alignment.
1.2. Perform a spell check and grammar correction on a document.
1.3. Create and format a simple report with headings, subheadings, and justified text.
1.4. Use the auto text feature to quickly insert pre-defined content into a document.
1.5. Practice using the dictionary feature to check definitions within a document.

Module II: features of ms word (14 hrs)

Theory

Page Formatting, Bookmarks, Mail Merge, Macros, Tables, File Management, Printing, Styles, linking and embedding object, Template..

Practice

2.1. Prepare a bio-data with proper formatting using MS Word.
2.2. Create a table in Word with calculated fields (e.g., total marks) using formulas.
2.3. Use the mail merge feature to create personalized letters for multiple recipients.
2.4. Design a template for official letters and save it for future use.
2.5. Record and run a macro to automate repetitive tasks in a document.

Module III: introduction to ms excel(14 hrs)

Theory

Introduction to MS-Excel, Creating & Editing Worksheet, Formatting and Essential Operations.

Practice

2.1. Sort data in ascending and descending order (both numbers and text).
2.2. Format a spreadsheet with borders, shading, and custom number formats.
2.3. Create a simple budget sheet with labeled columns and calculated totals.
2.4. Practice inserting and deleting rows/columns while maintaining data integrity.
2.5. Explore different Excel views and use the freeze panes feature to manage large datasets

Module IV: features of ms excel (14 hrs)

Theory

Formulas and Functions, Charts, Pivot table & Pivot Chart, Linking and Consolidation, Sorting, Filtering, Table, Validation, Goal Seek, Scenario.

Practice

3.1. Create a grade sheet with total marks, percentage, and grades using formulas.
3.2. Generate different types of charts (bar, pie, line) to visualize student performance.
3.3. Use pivot tables to analyze sales data and summarize key trends.
3.4. Apply data validation rules to ensure correct data entry in specific cells.
3.5. Use the goal seek feature to determine the required sales needed to achieve a target.

Module V: introduction to mspowerpoint (14 hrs)

Theory

Presentations, Creating, Manipulating & Enhancing Slides

Practice

4.1. Create a simple presentation with 5 slides on a chosen topic using formatting tools.
4.2. Add headers, footers, and page numbers to all slides.
4.3. Apply transitions between slides for smooth visual effects.
4.4. Use bullet points to organize content on slides for a seminar presentation.
4.5. Insert a background image and apply a design template to all slides.

Module VI: function of mspowerpoint (14 hrs)

Theory

Organizational Charts, Excel Charts, Word Art, Layering art Objects

Practice

5.1. Create an organizational chart showing a company hierarchy.
5.2. Import Excel charts into PowerPoint to present sales data visually.
5.3. Use WordArt to create artistic titles for a presentation.
5.4. Layer multiple art objects and images on a slide for an enhanced visual effect.
5.5. Create and apply custom slide layouts using placeholders and guides

Module VII:features of powerpoint(14 hrs)

Theory

Animations and Sounds, Inserting Animated Pictures or Accessing through Object, Inserting Recorded Sound Effect or In-Built Sound

Practice

6.1. Create an animated slide show presentation with transitions and effects.
6.2. Insert recorded audio clips to narrate key points in a presentation.
6.3. Add animations to text and objects to create engaging slide content.
6.4. Insert animated images and GIFs into a slide presentation.
6.5. Use in-built sound effects to add background music or transitions in a presentation.

Text Books:

1. Microsoft Office – Complete Reference – BPB Publication
2. Learn Microsoft Office – Russell A. Stultz – BPB Publication

Data Structures using C++

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1004	Data Structures using C++	1+3+0	Nil

Course Objectives

- **Understand** and **analyze** the fundamental data structures such as arrays, stacks, queues, linked lists, trees, and graphs in terms of time and memory complexity.
- **Develop** the ability to **implement** both static and dynamic data structures and algorithms using C++ in real-world scenarios.
- **Apply** efficient problem-solving techniques through sorting, searching, and hashing methods to optimize program performance.

Course Outcomes

- **Recall** and **identify** the key concepts of data structures like arrays, stacks, queues, and linked lists, and their operations.
- **Explain** and **compare** the time and space complexity of various data structures and algorithms, and select appropriate strategies for solving problems.
- **Design** and **implement** recursive algorithms and complex data structures like trees, graphs, and hash tables using C++.
- **Analyze** and **evaluate** different sorting and searching algorithms in terms of efficiency and performance.
- **Debug** and **optimize** C++ programs by choosing suitable data structures and algorithms for specific use cases.

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	3	3										2		1
CO2	2	3	2	2		2							2	2	1
CO3	2	3	3	2		2							3	3	
CO4	3	3	3	2		2							3	3	
CO5	3	3	3	2	2								2		1

*High-3, Medium-2, Low-1

Course content

Module I: Problem Solving Analysis

Theory

- Definition, Importance, Algorithm, Pseudocode, and ADTs.
- Complexity Analysis & Asymptotic Notations.
- Introduction to problem-solving using control structures, arrays, and the Raptor tool.

Practice

1.1 Write a program using Raptor to implement conditional control structures.
1.2 Implement a Raptor flowchart to perform basic array operations like insertion and deletion.
1.3 Create a pseudocode to analyze the time complexity of simple algorithms.
1.4 Solve a problem using loops in Raptor and translate it into C++.
1.5 Write a Raptor flowchart to implement selection and iteration constructs.

Module II: Array & Stack

Theory

- Array operations, Stack applications (Recursion, Infix to Postfix conversion).
- Asymptotic analysis and algorithm complexity analysis.

Practice

2.1 Write a program in C++ to perform insertion, deletion, sorting, and merging of arrays.
2.2 Implement a C++ program to evaluate mathematical expressions using stacks.
2.3 Convert an infix expression to postfix using recursion and evaluate it.
2.4 Create a stack and implement push, pop, and display operations.
2.5 Write a program to demonstrate recursion using a stack data structure.

Module III: Queue & Linked List

Theory

- Types of Queues, Single Linked List, and Linked List operations (Creation, Insertion, Deletion, Sorting, Reverse).

Practice

3.1 Write a program to implement queue operations such as insertion and deletion.
3.2 Implement a single linked list in C++ and perform operations like sorting and reversing.
3.3 Write a C++ program to implement a priority queue using an array or linked list.

3.4 Create a doubly linked list and implement insertion at both ends.
3.5 Write a menu-driven program for performing various linked list operations.

Module IV: Stack & Queue Using Linked List

Theory

- Circular linked list, Double linked list, and implementing Stack and Queue using Linked List.

Practice

4.1 Write a program to create a circular linked list and display its contents.
4.2 Implement a double linked list and perform insertion and deletion operations.
4.3 Create a program that implements a stack using a linked list.
4.4 Write a program to implement a queue using a linked list and perform enqueue and dequeue operations.
4.5 Implement a circular queue using a linked list in C++.

Module V: Trees

Theory

- Introduction to Trees, Binary trees, Search trees, AVL trees, and Tree traversals.

Practice

5.1 Write a C++ program to create and display a binary tree.
5.2 Implement a binary search tree and perform operations like insertion and deletion.
5.3 Write a program to print all pairs from two BSTs where the sum is greater than a given value.
5.4 Write a recursive function to remove duplicate entries from a BST.
5.5 Create and display an AVL tree with insertion and deletion.

Module VI: Searching & Sorting

Theory

- Searching & Sorting algorithms: Linear search, Binary search, Insertion sort, Bubble sort, Merge sort, Quick sort, Heap sort.

Practice

6.1 Implement a C++ program for linear and binary search.
6.2 Write a program to implement selection sort, bubble sort, and insertion sort.
6.3 Implement a C++ program for merge sort and quick sort.
6.4 Create a program to perform heap sort and display the sorted elements.
6.5 Compare the time complexity of various sorting algorithms with a sample input.

Module VII: Hashing

Theory

- Introduction to Hashing, Hash tables, Collision resolution methods (Linear Probing, Quadratic Probing, Double Hashing), Graph Terminology and Traversals.

Practice

7.1 Write a program to perform linear probing for hash table collision resolution.
7.2 Implement a C++ program using double hashing for collision resolution.
7.3 Create a program to represent a graph using an adjacency matrix and traverse it.
7.4 Implement depth-first search (DFS) and breadth-first search (BFS) algorithms for graph traversal.
7.5 Write a C++ program to find the shortest path in a graph using Dijkstra's algorithm.

Test Book

- **Data Structures and Algorithms in C++** by Adam Drozdek
- **Data Structures Using C++** by Michael Goodrich

Database Management Systems

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1005	Database Management Systems	2+2+0	Nil

Course Objectives

- **Understand Core DBMS Concepts**: Explain fundamental concepts of database management systems, including data abstraction, data models, data independence, and the overall structure and lifecycle of a DBMS.
- **Develop Proficiency in Data Modeling**: Design and create Entity-Relationship (ER) diagrams and object-oriented data models to effectively capture and represent complex real-world scenarios in databases.
- **Master SQL and Relational Database Design**: Execute SQL queries, manipulate data, and perform database normalization to ensure the design and implementation of efficient and reliable relational databases.

Course Outcomes

- **Knowledge**: Recall and describe fundamental concepts of DBMS, including data models, schemas, and the lifecycle of DBMS applications. (Module 1)
- **Comprehension**: Explain and differentiate between various data models (conceptual, physical, logical) and their components, including ER and object-oriented models. (Module 2)
- **Application**: Apply relational database design principles to normalize data to the Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF) in practical scenarios. (Module 4)
- **Analysis**: Analyze complex SQL queries and subqueries, and interpret their results to solve real-world problems related to data retrieval and manipulation. (Module 5 and 6)
- **Synthesis**: Design and implement a complete database system from schema creation to query formulation, ensuring effective data management and integrity. (Module 7)

Course Outcome to Program Outcome Mapping:

CO/PO/PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2	1	2	-	-	-	1	-	-	1	3	2	3
CO2	3	3	3	2	3	-	-	1	2	2	-	2	3	3	3
CO3	3	3	3	2	3	-	-	1	2	2	1	2	3	3	3
CO4	3	3	3	3	3	-	-	2	2	3	2	2	3	3	3
CO5	3	3	3	3	3	-	-	2	3	3	2	2	3	3	3

*High-3, Medium-2, Low-1

Course Content

Module I: DBMS Concepts

Theory: Introduction to data abstraction, data models, and the architecture of DBMS. Explore instances, schemas, and the overall lifecycle of DBMS applications.

Practice:

1.1: Define and describe data abstraction and the different levels of data models (conceptual, logical, and physical).

1.2: Explain the roles of instances and schemas in a database system.

1.3: Outline the components of a DBMS and their functions.

1.4: Discuss the lifecycle of a DBMS application, including development, deployment, and maintenance.

1.5: Use appropriate database terminology to describe common database operations and constraints.

Module II: Data Modelling

Theory: Study of various data models including ER models and object-oriented models. Learn to create and interpret ER diagrams and convert them to relational schemas.

Practice:

2.1: Create an ER diagram based on a given set of requirements, clearly identifying entities, attributes, and relationships.

2.2: Convert the given ER diagram into a relational schema.

2.3: Define the components and symbols used in ER modeling.

2.4: Illustrate the process of mapping ER diagrams to tables, including identifying primary and foreign keys.

2.5: Develop a complex ER diagram for a given scenario, such as a sports league or a library system, and explain your design choices.

Module III: Relational DBMS Model

Theory: Introduction to the relational model concepts, including attributes, domains, and the concept of integrity constraints. Overview of relational query languages like relational algebra and calculus.

Practice:

3.1: Define attributes and domains and illustrate their importance in the relational model.

3.2: Explain the concept of relational integrity constraints and how they are enforced.

3.3: Demonstrate basic operations of relational algebra and their SQL equivalents.

3.4: Solve problems using tuple and domain relational calculus.

3.5: Write SQL queries to demonstrate the application of relational algebra operations in practice.

Module IV: Relational Database Design

Theory: Understanding normalization principles, including 1NF, 2NF, 3NF, and BCNF, to design efficient relational schemas.

Practice:

4.1: Normalize a given table to First Normal Form (1NF) and explain the process.

4.2: Apply Second Normal Form (2NF) to the normalized table and describe the changes.

4.3: Convert the table to Third Normal Form (3NF) and justify the normalization process.

4.4: Demonstrate how to identify and resolve anomalies in database design through normalization.

4.5: Perform normalization on a provided table and ensure it meets the requirements of BCNF.

Module V: SQL

Theory: Study SQL query structures including DDL, DML, TCL commands, and their applications. Understand the role of views and indexes.

Practice:

5.1: Write SQL commands to create tables with appropriate constraints and data types.

5.2: Insert sample data into the created tables and verify data integrity.

5.3: Write SQL queries to retrieve and manipulate data from the tables using SELECT, UPDATE, and DELETE commands.

5.4: Use aggregate functions to perform calculations such as SUM, AVG, MAX, and MIN.

5.5: Implement and use indexes and views to optimize query performance.

Module VI: Aggregate Functions

Theory: Understanding and using aggregate functions, set operations, predicates, joins, and data manipulation.

Practice:

6.1: Create tables and insert data for testing aggregate functions such as COUNT, SUM, and AVG.

6.2: Perform set operations (UNION, INTERSECT, EXCEPT) on sample data.

6.3: Write queries using different types of joins (INNER, LEFT, RIGHT, FULL) to retrieve combined data from multiple tables.

6.4: Apply predicates and conditions using the LIKE operator and analyze their effects.

6.5: Manipulate data with aggregate functions and sort results based on specific criteria.

Module VII: Transaction Management

Theory: Explore transaction management, concurrency control, and subqueries. Study techniques for maintaining database integrity and handling concurrent transactions.

Practice:

7.1: Write SQL queries to perform basic transaction management operations (COMMIT, ROLLBACK).

7.2: Design queries to handle data manipulation, including INSERT, UPDATE, and DELETE within transactions.

7.3: Use subqueries to solve complex queries and retrieve data based on conditions.

7.4: Implement concurrency control mechanisms to handle multiple transactions.

7.5: Apply grouping functions to aggregate data and analyze results based on different conditions.

Text Books

Database Management Systems: Raghu Ramakrishnan
ORACLE PL/SQL Programming – Scott Urman BPB Publications.

References

Database Systems Concepts – Henry F Korth, Abraham Silberschatz.
Database Management Systems – Alexis Leon, Mathews Leon – Leon, Vikas Publications

Java Programming

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1006	Java Programming	1+2+1	Nil

Course Objectives

- Introduce the Java programming language and its features and implement object-oriented programming concepts.
- Utilize the Java Collections Framework, manage exceptions, and perform file I/O operations.

Develop multithreaded applications and connect to databases using JDBC.

Course Outcomes

- **CO1**: Recall the features and basic syntax of Java. *(Remembering)*
- **CO2**: Explain object-oriented programming concepts. *(Understanding)*
- **CO3**: Apply collections, exception handling, and file I/O operations in Java. *(Applying)*
- **CO4**: Analyze multithreaded applications and concurrency issues. *(Analyzing)*
- **CO5**: Develop complex Java applications with advanced features. *(Creating)*

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2										2	2	2
CO2	3	3	3	2	2								1	1	1
CO3	2		2										2	2	2
CO4	2	3	3	2									2	3	2
CO5	3	3	3	2	2								2	2	2

*High-3, Medium-2, Low-1

Course Content

Module 1: Introduction to Java (10 hours)

Theory

- History, Features of Java
- Setting up JDK and IDE
- Basic Syntax, Data Types

Practice

- Experiment 1.1: Write a program to print "Hello, World!" in Java.
- Experiment 1.2: Implement a program to demonstrate the use of variables and data types in Java.
- Experiment 1.3: Develop a program to perform arithmetic operations in Java.
- Experiment 1.4: Write a program to demonstrate the use of conditional statements in Java.
- Experiment 1.5: Implement a program to demonstrate the use of loops in Java.

Module 2: Object-Oriented Programming in Java (12 hours)

Theory

- Classes, Objects, Constructors
- Inheritance, Polymorphism, Encapsulation
- Abstract Classes, Interfaces

Practice

- Experiment 2.1: Write a program to demonstrate the use of classes and objects in Java.
- Experiment 2.2: Implement a program to demonstrate the use of constructors in Java.
- Experiment 2.3: Develop a program to demonstrate the use of inheritance in Java.
- Experiment 2.4: Write a program to demonstrate the use of polymorphism in Java.
- Experiment 2.5: Implement a program to demonstrate the use of encapsulation in Java.

Module 3: Java Collections Framework (12 hours)

Theory

- Collections: List, Set, Map
- Iterators, Enhanced For-Loop
- Sorting and Searching Collections

Practice

- Experiment 3.1: Write a program to demonstrate the use of ArrayList in Java.
- Experiment 3.2: Implement a program to demonstrate the use of LinkedList in Java.
- Experiment 3.3: Develop a program to demonstrate the use of HashSet in Java.
- Experiment 3.4: Write a program to demonstrate the use of TreeSet in Java.
- Experiment 3.5: Implement a program to demonstrate the use of HashMap in Java.

Module 4: Exception Handling and I/O (12 hours)

Theory

- Exception Handling Mechanisms
- Types of Exceptions: Checked, Unchecked

- File Handling: Reading, Writing, Serialization

Practice

- Experiment 4.1: Write a program to demonstrate the use of try-catch block in Java.
- Experiment 4.2: Implement a program to demonstrate the use of multiple catch blocks in Java.
- Experiment 4.3: Develop a program to demonstrate the use of nested try block in Java.
- Experiment 4.4: Write a program to demonstrate the use of finally block in Java.
- Experiment 4.5: Implement a program to demonstrate the use of throw and throws keyword in Java.

Module 5: Multithreading and Concurrency (10 hours)

Theory

- Threads: Creation, Management
- Synchronization, Concurrency Utilities
- Thread Pooling

Practice

- Experiment 5.1: Write a program to create a thread by extending Thread class in Java.
- Experiment 5.2: Implement a program to create a thread by implementing Runnable interface in Java.
- Experiment 5.3: Develop a program to demonstrate thread synchronization in Java.
- Experiment 5.4: Write a program to demonstrate inter-thread communication in Java.
- Experiment 5.5: Implement a program to demonstrate deadlock in Java.

Module 6: GUI Programming (15 hours)

Theory

- Introduction to AWT and Swing
- Container, Components
- Layout Managers
- Event Handling
- Introduction to Java Database Connectivity (JDBC)

Practice

- Experiment 6.1: Write a program to create a login page using Swing.
- Experiment 6.2: Validate the login form.
- Experiment 6.3: Develop a program to connect to a database using JDBC in Java.
- Experiment 6.4: Write a program to perform CRUD operations using JDBC in Java.
- Experiment 6.5: Implement a program to demonstrate the use of PreparedStatement.
- Experiment 6.6: Create a registration form and store data using JDBC.

Module 7: Advanced Java Basics (10 hours)

Theory

- Generics, Lambda Expressions, Stream API
- Annotations, Reflection

Practice

- Experiment 7.1: Write a program to demonstrate the use of generics in Java.
- Experiment 7.2: Implement a program to demonstrate the use of lambda expressions in Java.
- Experiment 7.3: Develop a program to demonstrate the use of Stream API in Java.
- Experiment 7.4: Write a program to demonstrate the use of annotations in Java.
- Experiment 7.5: Implement a program to demonstrate the use of reflection in Java.

Projects (10 hours):

- Project 1: Library Management System
- Project 2: E-commerce Application Backend
- Project 3: Multithreaded Chat Application

Textbooks:

1. "Java: The Complete Reference" by Herbert Schildt.
2. "Core Java Volume I - Fundamentals" by Cay S. Horstmann and Gary Cornell

Reference Books:

1. "Effective Java" by Joshua Bloch.
2. "Head First Java" by Kathy Sierra and Bert Bates.
3. "Programming with Java" by E. Balagurusamy.

Internet and Web Technology

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1007	Internet and Web Technology	1+2+1	Nil

Course Objectives

- **Understand** the core concepts and hardware components of the Internet, including servers, clients, and networking devices like routers and switches.
- **Apply** the principles of web development to design and create functional websites using HTML, CSS, and JavaScript.
- **Analyze** the structure and elements of web pages and implement XML for efficient data management and transmission on the web.

Course Outcomes

- **Identify** and **describe** the basic hardware components of the Internet and their interactions (servers, clients, routers, switches, etc.).
- **Design** and **create** web pages using HTML and CSS with an emphasis on structure, accessibility, and usability.
- **Develop** dynamic web pages with JavaScript, incorporating control statements, functions, and form validation.
- **Analyze** web pages to identify key elements, tags, and attributes and enhance the visual appearance using CSS.
- **Create** and **validate** XML documents and schemas to store and transfer structured data.

Course Outcome to Program Outcome Mapping:

CO/PO/PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2	1	2	-	-	-	2	2	2	2	3	3	3
CO2	3	3	3	2	3	-	-	-	2	2	2	2	3	3	3
CO3	3	3	3	3	3	-	-	-	2	2	2	2	3	3	3
CO4	3	3	3	2	3	-	-	-	2	2	2	2	3	3	3
CO5	3	3	3	3	3	-	-	-	2	2	2	2	3	3	3

*High-3, Medium-2, Low-1

Course Content

Module I: Introduction to the Internet and the World Wide Web (10 hrs)

Theory

- Introduction to the Internet and Web Technologies.
- Web Pages: Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP).
- Domain Names, URLs, Web Browsers, Web Servers, Web Hosting, HTML Basics.

Practice

1.1 Explore HTTP requests and responses using browser developer tools.
1.2 Demonstrate FTP operations to upload/download files.
1.3 Create a basic HTML webpage with heading and paragraph tags.
1.4 Set up a local web server and host a simple webpage.
1.5 Create a webpage that includes hyperlinks to different sections of the same document.

Module II: HTML Fundamentals and Web Design (15 hrs)

Theory

- Introduction to HTML, URI, Structure of HTML.
- Basic Tags: Heading, Paragraph, Links, Images.
- Planning and designing websites, creating links between documents.
- Tables, Frames, and Forms for structuring content.

Practice

2.1 Create a webpage with lists (ordered, unordered, and nested lists).
2.2 Develop a webpage using tables to display structured information.
2.3 Create a form using HTML to collect user data, including text fields, checkboxes, and buttons.
2.4 Build a webpage using frames to divide the page into different sections.
2.5 Design a webpage that includes images, text, and a clickable image as a hyperlink.

Module III: JavaScript and Cascading Style Sheets (CSS) (10 hrs)

Theory

- Introduction to JavaScript: Control Statements, Functions, Arrays, Objects.
- CSS Basics: External, Internal, and Inline Styles.
- Class Selector, div & span tags.
- JavaScript for validation and interaction on web pages.

Practice

3.1 Create a web form for student registration and validate inputs using JavaScript.
3.2 Use CSS to apply styles to a webpage, including fonts, colors, and layouts.
3.3 Write a JavaScript function to manipulate arrays and display results on a webpage.
3.4 Design a webpage with a digital clock using JavaScript.
3.5 Create a webpage that uses JavaScript to validate login form entries (username and password).

Module IV: Document Object Model (DOM) and Dynamic HTML (DHTML) (10 hrs)

Theory

- HTML DOM: Introduction, Manipulating DOM with JavaScript.
- XML: Introduction, Features, Structure of XML documents.
- Dynamic HTML (DHTML): Creating interactive forms with DHTML.

Practice

4.1 Write a JavaScript program to dynamically update webpage content using DOM manipulation.
4.2 Create a simple XML document to represent product details and render it using an XML parser.
4.3 Design a form that dynamically changes based on user input using DHTML.
4.4 Develop a script to retrieve and display XML data in an HTML page.
4.5 Create a basic interactive game using JavaScript and DOM manipulation.

Module V: Advanced XML and XML DOM (10 hrs)

Theory

- Advanced features of XML: Schemas, Validation, Namespaces.
- XML DOM: Structure and traversal of XML documents.
- Integration of XML with HTML and JavaScript.

Practice

5.1 Create an XML schema to validate an XML document containing student information.
5.2 Write a JavaScript program to parse and display an XML file on a webpage.
5.3 Build an application that exchanges data between a server and client using XML.
5.4 Design an XML-based webpage that incorporates XSLT for formatting.
5.5 Create an XML document that stores contact information and validate it using a DTD.

Module VI: CGI/Perl Scripting and Web Programming (10 hrs)

Theory

- Introduction to CGI, Perl scripting for web applications.
- Testing and debugging Perl scripts.
- Use of CGI for server-side form handling.

Practice

6.1 Write a Perl CGI script to display form data submitted by a user.
6.2 Create a web-based calculator using HTML for input and Perl CGI for processing.
6.3 Develop a Perl script to store form data in a server-side file.
6.4 Test and debug a CGI Perl script that handles file uploads.
6.5 Create a dynamic web page using Perl and CGI for database-driven content.

Module VII: Final Projects and Real-World Applications (5 hrs)

Theory

- Best practices for designing and developing web applications.
- Introduction to project planning, debugging, and deployment.
- Real-world applications of web technologies in projects.

Practice

7.1 Create a simple tribute webpage for a famous personality, applying all learned concepts.
7.2 Design a restaurant website with an interactive menu and reservation form.
7.3 Develop a personal portfolio website with responsive design using HTML, CSS, and JavaScript.
7.4 Build an online quiz application using JavaScript and validate answers in real-time.
7.5 Create a small e-commerce website showcasing product listings and contact forms.

Test book

1. Web Technology: Theory and Practice by M. Srinivasan - This book covers key technologies used to create web applications, including HTML, CSS, JavaScript, and server-side scripting.

2. Front-End Developer Handbook 2019 by Cody Lindley - A comprehensive guide to front-end development, including emerging technologies and best practices.

Programming in Python

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1008	**Programming in Python**	1+2+1	NIL

Course Objectives

- **Understand** the fundamental programming features and elements of Python, including variables, data types, operators, and control structures, to effectively write and debug Python code. (Bloom's Level: Understand)
- **Utilize** Python libraries for data analysis and visualization, including Pandas, NumPy, and Matplotlib, to perform data manipulation, visualization, and problem-solving tasks. (Bloom's Level: Apply)
- **Design** and **implement** interactive data visualizations and dashboards using Python libraries to present and interpret complex data effectively. (Bloom's Level: Create)

Course Outcomes

- Read and manipulate data using Python functions and libraries, demonstrating proficiency in data handling and preparation. (Bloom's Level: Apply)
- Visualize data using Python libraries to generate various types of plots and charts, aiding in the interpretation and communication of data insights. (Bloom's Level: Apply)
- Solve problems using Python by applying programming concepts and libraries to real-world scenarios and data analysis tasks. (Bloom's Level: Apply)
- Develop interactive dashboards and visualizations that include layout design, reporting, and live updating components to enhance user interaction and data presentation. (Bloom's Level: Create)
- Employ advanced visualization techniques with Python to represent data in diverse formats and customize visualizations to meet specific analytical needs. (Bloom's Level: Analyze)

Course Outcome to Program Outcome Mapping:

-

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2										2	2	2
CO2	3	3	3	2	2								1	1	1
CO3	2		2										2	2	2
CO4	2	3	3	2									2	3	2
CO5	3	3	3	2	2								2	2	2

*High-3, Medium-2, Low-1

Course Content

MODULE-I: Elements of Python Programming

Theory: Basics of Python programming including variables, data types, operators, and strings.

Practice:

1.1 Write Python programs to declare and use variables of different data types (integer, float, string) and perform basic operations.
1.2 Implement string operations and manipulations including slicing, concatenation, and formatting.
1.3 Create Python scripts to demonstrate the use of various operators (arithmetic, relational, logical).
1.4 Develop small programs to convert data between different types (e.g., string to integer, float to string).
1.5 Validate data types and operators by designing test cases and debugging common errors.

MODULE-II: Control Structures, Python Functions, Arrays

Theory: Control flow, functions, and basic data structures in Python.

Practice:

2.1 Implement conditional statements (if, elif, else) and loops (for, while) in Python scripts to control program flow.
2.2 Design and test Python functions to perform specific tasks, including parameter passing and return values.
2.3 Create and manipulate arrays (lists) in Python, including adding, removing, and accessing elements.
2.4 Develop programs that use nested loops and functions to solve complex problems.
2.5 Write unit tests for functions and array manipulations to ensure correctness and handle edge cases.

MODULE-III: Class and Objects, Modules, File Handling

Theory: Object-oriented programming, module usage, and file operations in Python.

Practice:

3.1 Design and implement Python classes and objects, including constructors, methods, and inheritance.
3.2 Create and import Python modules, demonstrating the use of functions and classes from external files.
3.3 Write programs to handle files (read, write, append) and process different file formats (text, CSV).
3.4 Develop a simple project that uses classes, modules, and file handling to achieve a specific functionality.
3.5 Test file handling operations by simulating various file scenarios and handling exceptions.

MODULE-IV: Plotting

Theory: Introduction to basic plotting techniques in Python.

Practice:

4.1 Generate basic plots using Python's Matplotlib library, including line plots and scatter plots.
4.2 Create histograms and bar plots to visualize distributions and categorical data.
4.3 Implement area plots and pie charts to represent proportions and areas.
4.4 Customize plots with titles, labels, and legends to enhance readability and presentation.
4.5 Compare different types of plots for their suitability in representing various data types.

MODULE-V: Data Reading Using Python Functions

Theory: Data reading and manipulation using Python libraries.

Practice:

5.1 Read data from different sources (HTML, CSV, MS Excel) using Python libraries such as Pandas.
5.2 Scrape data from websites using web scraping techniques and libraries like BeautifulSoup.
5.3 Compile and arrange data from multiple sources, performing data munging to clean and prepare data for analysis.
5.4 Implement data reading functions to handle various data formats and structures.
5.5 Analyze and visualize the processed data to extract meaningful insights using Pandas and other libraries.

MODULE-VI: Data Visualization Using Python Libraries

Theory: Advanced visualization techniques and dashboard basics.

Practice:

6.1 Create scatter plots, line charts, histograms, and bar charts using Python libraries to visualize data.
6.2 Develop bubble charts and heatmaps to represent complex data relationships and patterns.
6.3 Design and implement basic dashboards using Python libraries (e.g., Dash) to present data interactively.
6.4 Customize dashboard components, including layout and interactive elements, to enhance user experience.
6.5 Test live updating features in dashboards to ensure real-time data representation.

MODULE-VII: Interactive Data Visualization

Theory: Advanced interactive visualizations with Python libraries.

Practice:

7.1 Plot simple and advanced scatter plots using glyphs and customize them with different shapes and colors.
7.2 Create and manipulate additional glyphs, lines, and patches for enhanced visualization using libraries like Bokeh.
7.3 Visualize data from NumPy arrays and Pandas dataframes, applying different plot types and customization options.
7.4 Implement color mapping and selection features to highlight and interact with specific data points.
7.5 Design and arrange complex layouts with nested rows and columns to display multiple plots in a cohesive manner.

Assignments

1. **Read and Manipulate Data Using Python Functions:**
    - Implement Python functions to read data from various formats (CSV, Excel) and perform basic data cleaning.
2. **Visualize Data Using Python Libraries:**
    - Create and customize different types of plots (scatter, line, bar) using Matplotlib and Seaborn to represent provided datasets.
3. **Solve a Real-World Problem Using Python:**
    - Develop a Python program to analyze and visualize data from a given problem scenario, applying appropriate libraries and techniques.
4. **Design a Basic Dashboard:**
    - Create an interactive dashboard using Dash or similar libraries, including layout design, reporting, and live updating components.
5. **Advanced Data Visualization:**
    - Implement advanced visualization techniques with Bokeh or Plotly, including interactive plots with glyphs, custom color mapping, and dynamic elements.

Test Book

1. Python Testing with Pytest, Second Edition by Brian Okken:
2. Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing by David Sale:

Operating System Concepts

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1009	Operating System Concepts	2+2+0	Nil

Course Objectives

- **Understand** the structure, components, and services provided by an operating system to manage hardware and software resources effectively.
- **Analyze** process synchronization, scheduling algorithms, and resource management techniques for improved system performance and efficiency.
- **Apply** shell scripting and system-level programming to automate tasks and manage processes, files, and memory in a Linux environment.

Course Outcomes

- **Identify** and **describe** the core concepts of operating systems, including file systems, processes, threads, and memory management.
- **Demonstrate** the use of system calls, Linux commands, and utilities to interact with the OS and manage resources.
- **Develop** shell scripts to perform automated system tasks and manage I/O, decision-making, and looping.
- **Implement** and **simulate** process and thread scheduling algorithms to measure CPU performance metrics like turnaround and waiting time.
- **Evaluate** deadlock scenarios and memory management techniques such as paging and segmentation to ensure resource efficiency.

CO-PO-PSO Mapping:

CO/PO/PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2	-	2	-	-	-	-	-	-	-	3	2	3
CO2	3	3	3	2	3	-	-	-	-	-	-	-	3	3	3
CO3	3	3	3	3	3	-	-	-	-	2	-	2	3	3	3
CO4	3	3	3	2	3	-	-	-	-	2	-	2	3	3	3
CO5	3	3	3	3	3	-	-	-	-	2	-	2	3	3	3

***High-3, Medium-2, Low-1**

Course Content

Module I: Overview and System Call

Theory

- Overview of OS, its structure, services, and types of system calls.
- Major OS services such as process management and concurrent programming.

Practice

1.1 Write a program to copy content from one file to another using unbuffered I/O system calls.
1.2 Implement a program to print the contents of a directory using system calls.
1.3 Create an infinite loop and kill it using the kill command with a SIGINT signal.
1.4 Write a program to open, read, and display file contents using low-level I/O system calls.
1.5 Simulate the creation of a new process using fork() and demonstrate process creation.

Module II: Linux Commands and Utilities

Theory

- Understanding various Linux commands and utilities like su, sudo, ls, ps, top, grep, ssh, and more.
- Basic file management, user management, and process monitoring tools in Linux.

Practice

2.1 Gain root privilege using su and sudo commands.
2.2 Archive and compress files using tar, gzip, and bzip2.
2.3 Create a new user, set up a password, and manage password policies.
2.4 Login to a remote system using SSH and execute basic commands.
2.5 Monitor processes using top and ps and demonstrate process killing.

Module III: File System

Theory

- Understanding Linux's hierarchical file system, directory structure, and file access permissions.
- Learn about symbolic links, hard links, and Access Control Lists (ACLs).

Practice

3.1 Demonstrate the hierarchical file system and directory navigation in Linux.
3.2 Create symbolic and hard links for files and directories.
3.3 Modify and control access to files using Linux file system permissions.
3.4 Manage file permissions using Access Control Lists (ACLs) for finer access control.
3.5 Write a program to demonstrate the working of symbolic links.

Module IV: Vi Editor

Theory

- Introduction to the vim editor, its features, modes, and buffer operations.
- Editing, searching, and substituting text using vim.

Practice

4.1 Create, edit, save, and exit a file using vim.
4.2 Edit a system file using advanced editing features in vim.
4.3 Practice text searching and substitution within files.
4.4 Demonstrate copying, moving, and deleting text within a file.
4.5 Utilize vim buffers to manage multiple files simultaneously.

Module V: Shell Scripting

Theory

- Understanding bash scripting, variables, arrays, decision-making, and loop controls.
- Introduction to input/output redirection and functions in shell scripting.

Practice

5.1 Write a script to demonstrate decision-making with if statements and loop controls.
5.2 Create a shell script that uses arrays and functions to manage data.
5.3 Demonstrate I/O redirection by writing a script that reads input from a file and outputs to another.
5.4 Implement a script that uses for and while loops to automate tasks.
5.5 Write a script that handles command-line arguments and processes them.

Module VI: Process and Thread Management

Theory

- Process concepts, state, context switching, and inter-process communication.
- Multi-threading models and process scheduling algorithms (FCFS, SJF, Priority, RR).

Practice

6.1 Write a program to create a process using fork().
6.2 Simulate the producer-consumer problem using semaphores for synchronization.
6.3 Write a multi-threaded program to perform parallel computation.
6.4 Implement a program to simulate CPU scheduling algorithms: FCFS, SJF, Priority, and Round Robin.
6.5 Analyze and display turnaround time and waiting time for each CPU scheduling algorithm.

Module VII: Resource and Memory Management

Theory

- Deadlock detection, avoidance, and recovery techniques.
- Memory management techniques: swapping, paging, segmentation.

Practice

7.1 Write a program to simulate the Banker's algorithm for deadlock avoidance.
7.2 Implement page replacement algorithms: FIFO, LRU, and LFU.
7.3 Simulate deadlock detection and recovery mechanisms.
7.4 Write a program to demonstrate memory segmentation and paging.
7.5 Simulate a deadlock prevention strategy using resource allocation techniques.

Test Book and Reference Book

- Operating System Concepts by Silberschatz, Gagne, & Galvin:
- Modern Operating Systems by Andrew Stuart Tanenbaum:
- Operating Systems: Principles and Practice:

Dot Net Technology

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1010	Dot Net Technology	1+2+1	Nil

Course Objectives

- **Understand** the core concepts and hardware components of the Internet, including servers, clients, and networking devices like routers and switches.
- **Apply** the principles of web development to design and create functional websites using HTML, CSS, and JavaScript.
- **Analyze** the structure and elements of web pages and implement XML for efficient data management and transmission on the web.

Course Outcomes

- Identify and describe the basic hardware components of the Internet and their interactions (servers, clients, routers, switches, etc.).
- Design and create web pages using HTML and CSS with an emphasis on structure, accessibility, and usability.
- Develop dynamic web pages with JavaScript, incorporating control statements, functions, and form validation.
- Analyze web pages to identify key elements, tags, and attributes and enhance the visual appearance using CSS.
- Create and validate XML documents and schemas to store and transfer structured data.

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2										2	2	2
CO2	3	3	3	2	2								1	1	1
CO3	2		2										2	2	2
CO4	2	3	3	2									2	3	2
CO5	3	3	3	2	2								2	2	2

*High-3, Medium-2, Low-1

Course content

Module I: Introduction to .NET Framework (7 hrs)

Theory

- Vision and goals of .NET.
- Building blocks and evolution of .NET.

- The .NET Framework Architecture: CLR, CTS, CLS.
- Assemblies, JIT Compilation, Intermediate Language (IL), IL Disassembler (ILDASM), and Namespaces.

Practice

1.1 Install and explore the .NET SDK.
1.2 Create a simple "Hello World" C# application.
1.3 Demonstrate the use of IL Disassembler (ILDASM) to explore compiled C# code.
1.4 Write a program to display the Common Language Runtime (CLR) version.
1.5 Create and execute a simple .NET assembly.

Module II: C# Methods and Features (8 hrs)

Theory

- Introduction to methods in C#: Method structure, calling a method.
- Parameter types: Value, Reference, and Output parameters.
- Method overloading, virtual methods, and method overriding.

Practice

2.1 Implement a C# program using value and reference parameters.
2.2 Demonstrate method overloading with different parameter types.
2.3 Write a program with virtual methods and show how overriding works.
2.4 Implement output parameters in methods.
2.5 Write a program to use virtual methods and override them in derived classes.

Module III: C# Classes and Inheritance (10 hrs)

Theory

- Working with classes: Constants, fields, methods, properties, indexers, events.
- Class inheritance, virtual and override methods.
- Abstract classes, sealed classes, boxing and unboxing, namespaces, interfaces.
- Handling exceptions.

Practice

3.1 Implement a C# program using constants, fields, and properties.
3.2 Write a C# program to demonstrate class inheritance and method overriding.
3.3 Implement an interface in C# and demonstrate polymorphism.
3.4 Write a program to handle exceptions in C# using try, catch, and finally.
3.5 Demonstrate boxing and unboxing with C# classes.

Module IV: Windows Applications in .NET (5 hrs)

Theory

- Windows Forms architecture.
- Working with common Windows controls: TextBox, Button, ListBox, and ComboBox.
- Containers, menus, and tool strips.

- Data controls and reporting.

Practice

4.1 Create a simple Windows Forms application with a TextBox and Button.
4.2 Add and use multiple controls like ListBox and ComboBox in a form.
4.3 Design a form with menus and tool strips.
4.4 Implement event handling for Windows controls (e.g., Button click).
4.5 Develop a simple calculator using Windows Forms.

Module V: ADO.NET and Database Programming (10 hrs)

Theory

- Introduction to ADO.NET.
- Working with DataSets, OLEDB, and SQL Server.
- Common database operations: Querying, inserting, updating, and deleting data.
- Data operations with single and multiple rows.
- Hierarchical data operations.

Practice

5.1 Create a connection to a SQL Server database using ADO.NET.
5.2 Implement a program to retrieve data using DataSets.
5.3 Develop a C# program that performs basic SQL operations: Insert, Update, and Delete.
5.4 Implement hierarchical data access using ADO.NET.
5.5 Write a program that retrieves multiple rows from a database and displays them in a GridView.

Module VI: ASP.NET and Web Forms (10 hrs)

Theory

- ASP.NET architecture and differences from ASP.
- Working with ASP.NET Web Forms.
- Code-behind model.
- Validation controls in ASP.NET.
- Data binding, GridView, Data Repeater, and Data List.

Practice

6.1 Create a basic web application in ASP.NET with a Web Form.
6.2 Implement form validation using ASP.NET validation controls.
6.3 Bind data to a GridView from a SQL Server database.
6.4 Design a login page using ASP.NET and validate user credentials.
6.5 Implement a page with a Data Repeater and Data List control for dynamic content.

Module VII: ASP.NET Data Binding and Login Controls (5 hrs)

Theory

- Server controls and advanced data binding.
- ASP.NET login controls.
- Data source controls: SQL Data Source, Object Data Source.
- Advanced GridView and DetailsView controls.

Practice

7.1 Implement advanced data binding in ASP.NET using a SQL Data Source.
7.2 Create a login form using ASP.NET Login control.
7.3 Bind data to a DetailsView control and edit data.
7.4 Develop a registration page using ASP.NET and validate inputs with server controls.
7.5 Create a GridView control with sorting and paging functionality.

Assignments

1. Explain the .NET framework and its components.
2. Differentiate between managed and unmanaged code in .NET.
3. Explain ASP.NET and how it differs from ASP.
4. What are EXE and DLL in .NET?
5. Describe the MVC architecture in .NET.
6. Explain the role of Garbage Collector in .NET.
7. Define CTS, CLR, and JIT.

Textbook:

1. **Jeff Ferguson, Brian Patterson, Jason Beres**, *C# Programming Bible*, Wiley Publishing Inc., Reprint 2006.

Reference Books:

1. **Jeff Prosise**, *Programming .Net*, 2nd Edition, WP Publishers & Distributors Pvt. Ltd, 2009.
2. **Kevin Hoffman & Jeff Gabriel**, *Professional .Net Framework*, 1st Edition, Wrox Press Publishers, 2006.

Assignments:-

1. What is the .Net framework?
2. What is meant by Managed and Unman-aged code?
3. What is ASP.Net?
4. What is an Assembly? What are the different types of Assemblies?
5. What is an EXE and a DLL?
6. What is MVC?
7. What is a Garbage Collector?
8. What is CLR?
9. What is CTS?
10. What is JIT

Fundamentals of Algorithm Design and Analysis

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1011	Fundamentals of Algorithm Design and Analysis	2+2+0	NIL

Course Objectives

- **Understand Algorithm Design and Analysis**: Gain a thorough understanding of the paradigms and approaches for designing and analyzing algorithms, including how to use asymptotic notation to classify and compare algorithms.
- **Implement Fundamental Algorithms**: Learn to implement and evaluate fundamental algorithms such as sorting algorithms, dynamic programming approaches, and graph algorithms, and understand their time and space complexities.
- **Explore Advanced Algorithmic Techniques**: Study advanced algorithmic techniques including greedy methods, flow networks, and NP-completeness to appreciate their impact on practical problem-solving and efficiency.

Course Outcomes (Using Bloom's Taxonomy)

- **Knowledge**: Describe various paradigms of algorithm design, including divide and conquer, dynamic programming, and greedy methods. (Module 1, 2, 3, 4)
- **Comprehension**: Explain the principles of algorithmic analysis, including asymptotic notation, recurrence relations, and the concept of NP-completeness. (Module 1, 7)
- **Application**: Implement and test fundamental algorithms such as merge sort, quicksort, and shortest path algorithms, and report their performance metrics. (Module 2, 5)
- **Analysis**: Analyze the efficiency of different algorithms using their time and space complexities, and compare their performance in solving practical problems. (Module 2, 4, 5)
- **Synthesis**: Design efficient algorithms for complex problems, applying appropriate techniques such as dynamic programming or greedy methods, and understand their practical applications. (Module 3, 4, 6)

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1										2	1	1
CO2	3	2	1	1	1								2	1	1
CO3	3	3	2		2	2							2	1	1
CO4	3	3	2	1									3	2	1
CO5	3	3	3	2									3	3	2

*High-3, Medium-2, Low-1

Course Content

Module I: Introduction to Analysis and Design of Algorithms

Theory: Study the basics of algorithm analysis, including asymptotic notations (Big-O, Big-Ω, Big-Θ), growth of functions, and recurrence relations.

Practice:

1.1: Solve recurrence relations using the substitution method. Verify the solution with examples.
1.2: Use the recursion tree method to solve given recurrences and analyze the time complexity.
1.3: Apply the Master's theorem to solve recurrences and compare the results with other methods.
1.4: Analyze the growth of different functions (e.g., polynomial, exponential) and their impact on algorithm performance.
1.5: Derive and solve recurrence relations for common divide and conquer algorithms.

Module II: Divide and Conquer Algorithms

Theory: Study and implement algorithms that use the divide and conquer approach, including merge sort, quicksort, and heap sort.

Practice:

2.1: Implement and test Insertion Sort, reporting the number of comparisons made.
2.2: Implement and test Merge Sort, reporting the number of comparisons made.
2.3: Implement and test Heap Sort, reporting the number of comparisons made.
2.4: Implement and test Randomized Quicksort, reporting the number of comparisons made.
2.5: Implement Radix Sort, Selection Sort, and Shell Sort, and analyze their performance.

Module III: Dynamic Programming Approach

Theory: Explore dynamic programming techniques with a focus on problems like matrix chain multiplication and longest common subsequence (LCS).

Practice:

3.1: Write a program to solve the Matrix Chain Multiplication problem and analyze its time complexity.
3.2: Write a program to determine the Longest Common Subsequence (LCS) of two sequences.
3.3: Compare the dynamic programming approach to other methods for solving LCS and matrix chain multiplication.
3.4: Implement and analyze variations of dynamic programming problems, such as the Knapsack problem.
3.5: Design a dynamic programming solution for a given problem and discuss its efficiency.

Module IV: Greedy Method

Theory: Study the greedy approach to algorithm design, including problems like the fractional knapsack problem and Huffman coding.

Practice:

4.1: Solve the Fractional Knapsack problem using the greedy method and analyze the results.
4.2: Implement Huffman coding and decode a given encoded message.
4.3: Compare the greedy method with dynamic programming for the 0/1 Knapsack problem.
4.4: Design and analyze algorithms using backtracking and branch & bound techniques.
4.5: Solve and discuss various problems using greedy techniques and compare their effectiveness with other methods.

Module V: Single Source Shortest Paths

Theory: Study graph algorithms for finding the shortest paths from a single source, including Kruskal's, Prim's, Bellman-Ford, and Dijkstra's algorithms.

Practice:

5.1: Implement Breadth-First Search (BFS) in a graph and analyze its performance.
5.2: Implement Depth-First Search (DFS) in a graph and analyze its performance.
5.3: Write a program to determine the minimum spanning tree using Kruskal's algorithm.
5.4: Create a Red-Black Tree and perform operations such as insertion, deletion, and searching.
5.5: Implement and compare Bellman-Ford and Dijkstra's algorithms for finding shortest paths in a graph.

Module VI: Flow Network

Theory: Study network flow algorithms, including Ford-Fulkerson method, Fast Fourier Transform (FFT), and Rabin-Karp string matching algorithm.

Practice:

6.1: Implement the Ford-Fulkerson method for computing maximum flow in a network.
6.2: Implement and analyze the Fast Fourier Transform (FFT) using both DIF and DIT approaches.
6.3: Apply the Rabin-Karp string matching algorithm to solve pattern matching problems.
6.4: Compare the performance of different FFT algorithms and discuss their applications.
6.5: Solve real-world problems using flow network algorithms and string matching techniques.

Module VII: NP-Completeness

Theory: Explore NP-completeness, polynomial time solvability, verification, reducibility, and approximation algorithms.

Practice:

7.1: Explain the concept of NP-completeness and provide examples of NP-complete problems.
7.2: Discuss polynomial time solvability and its significance in computational complexity.
7.3: Implement and analyze approximation algorithms for the Traveling Salesman Problem (TSP).
7.4: Study and discuss various reduction techniques used to prove NP-completeness.
7.5: Evaluate the performance of approximation algorithms and their practical implications.

Test Book

- **"Introduction to Algorithms"** by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
- **"Algorithm Design Manual"** by Steven S. Skiena
- **"Data Structures and Algorithms Made Easy"** by Narasimha Karumanchi

Computer and Communication Networks

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1012	Computer and Communication Networks	2+2+0	NIL

Course Objectives

- **Describe** the general principles of data communication, including network models and data transmission methods, to provide a foundational understanding of how data is communicated across networks. *(Bloom's Level: Understand)*
- **Explain** how computer networks are organized using the layered approach, including the OSI and TCP/IP models, to understand the structure and functioning of network protocols. *(Bloom's Level: Understand)*
- **Implement** a simple Local Area Network (LAN) incorporating hubs, bridges, and switches, to demonstrate practical network configuration and management. *(Bloom's Level: Apply)*

Course Outcomes

- **Explain** the importance of data networks and the Internet in supporting business communications and everyday activities, illustrating their impact on modern life. *(Bloom's Level: Understand)*
- **Describe** how communication works in data networks and the Internet, including the role of protocols and network models in facilitating data exchange. *(Bloom's Level: Understand)*
- **Identify** the devices and services that are used to support communications across a network, and explain their functions and importance. *(Bloom's Level: Understand)*
- **Utilize** network protocol models to explain the layers of communications in data networks, including the OSI and TCP/IP models. *(Bloom's Level: Apply)*
- **Implement** and **analyze** a basic LAN setup with different network devices and protocols to demonstrate practical knowledge of network configurations and performance. *(Bloom's Level: Apply)*

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1										2	1	1
CO2	3	2	1	1	1								2	1	1
CO3	3	3	2		2	2							2	1	1
CO4	3	3	2	1									3	2	1
CO5	3	3	3	2									3	3	2

***High-3, Medium-2, Low-1**

Course Content

MODULE-I: Overview of Computer Networks

Theory: Introduction to network models (OSI, TCP/IP), network types (WAN, LAN, MAN), and historical networks (Novell, Arpanet).

Practice:

1.1 Analyze bus versus star-switch topologies concerning collision rates for a fixed number of nodes in an Ethernet LAN.
1.2 Compare various network topologies (e.g., bus, star, ring) regarding their advantages and disadvantages.

MODULE-II: Physical Layer

Theory: Transmission modes, guided and unguided media, multiplexing techniques (FDM, WDM, TDM), circuit switching.

Practice:

2.1 Analyze the differences between unicast and broadcast transmissions for a fixed number of transmitting nodes.
2.2 Compare various transmission media types, including guided (copper, fiber) and unguided (radio, microwave).

MODULE-III: Data Link Layer

Theory: Design issues, framing, error detection and correction, protocols like CRC, Stop-and-Wait, Sliding Window, HDLC, ATM.

Practice:

3.1 Verify the Stop-and-Wait protocol through simulation.
3.2 Test error detection and correction techniques (e.g., CRC) using provided data sets.
3.3 Implement and analyze the Sliding Window protocol to handle multiple data frames.
3.4 Simulate data link layer operations using HDLC and ATM protocols.
3.5 Compare the performance of different data link layer protocols.

MODULE-IV: Point-to-Point Access

Theory: PPP, multiple access protocols (Random Access, Controlled Access, Channelization), Ethernet, IEEE 802.11, Bluetooth, virtual circuits.

Practice:

4.1 Compare CSMA/CD and CSMA/CA protocols for a fixed number of transmitting nodes.
4.2 Verify the Selective Repeat protocol and analyze its efficiency.
4.3 Simulate distance vector and link state routing algorithms and evaluate their performance.
4.4 Test the Stop-and-Wait protocol and analyze its effectiveness in a controlled environment.

4.5 Implement and evaluate various multiple access protocols and their impact on network performance.

MODULE-V: Network Layer

Theory: Addressing, ARP, IPv4, ICMP, IPv6, ICMPv6, broadcast, multicast, congestion control algorithms, internetworking.

Practice:

5.1 Verify the Stop-and-Wait protocol and assess its functionality in different network scenarios.
5.2 Implement and test network layer protocols like ARP and ICMP in simulated environments.
5.3 Analyze congestion control algorithms and their application in network traffic management.
5.4 Compare the network layer in the Internet with that in ATM networks.
5.5 Implement IPv4 and IPv6 addressing schemes and evaluate their effectiveness.

MODULE-VI: Transport Layer

Theory: Process-to-process delivery, UDP, TCP, congestion control.

Practice:

6.1 Verify the Go-Back-N protocol and analyze its performance.
6.2 Test UDP protocol functionality and compare it with TCP in terms of reliability and performance.
6.3 Analyze TCP congestion control mechanisms and their impact on network throughput.
6.4 Implement and evaluate transport layer protocols in different scenarios.
6.5 Compare UDP and TCP in terms of their use cases and performance characteristics.

MODULE-VII: Application Layer

Theory: Client-server model, DNS, SMTP, FTP, HTTP, WWW.

Practice:

7.1 Implement a simple client-server application to understand the basics of application layer protocols.
7.2 Simulate DNS queries and analyze the resolution process for domain names.
7.3 Test the functionality of SMTP and FTP protocols in sending and receiving emails and files.
7.4 Analyze HTTP and its role in web communication, including methods and status codes.
7.5 Develop a basic web application and use HTTP to interact with web servers.

Assignments

1. **Define computer networks and discuss various network topologies:**
    - Define computer networks and describe topologies such as bus, star, ring, and mesh. Discuss their advantages and disadvantages.
2. **Applications of Computer Networks:**

- Explain the various applications of computer networks in different domains, including business and personal use.

3. **OSI Model Explanation:**
   - Describe the OSI model, including the functions, protocols, and services of each layer.
4. **LAN, WAN, MAN, ARPANET:**
   - Explain LAN, WAN, MAN, and the historical context of ARPANET.
5. **TCP/IP Model and Comparison with OSI:**
   - Describe the TCP/IP model, its layers, and compare it with the OSI model.
6. **Transmission Media in Computer Networks:**
   - Explain different types of transmission media, including guided and unguided media.
7. **Error-Correcting Techniques:**
   - Discuss various error-correcting techniques used in computer networks.
8. **Sliding Window Protocols:**
   - Explain the Sliding Window protocol and its application in data transmission.
9. **Elementary Data Link Layer Protocols:**
   - Describe basic data link layer protocols and their functionalities.
10. **ALOHA Protocols Comparison:**
    - Explain Pure ALOHA and Slotted ALOHA, and compare their performance at low load conditions.

Text Books:-

1. Data Communications and Networking: Behrouz A. Forouzan, Tata McGraw-Hill, 4thEd
2. Computer Networks: A. S. Tannenbum, D. Wetherall, Prentice Hall, Imprint of Pearson 5thEd

Reference Book: -

1. Computer Networks: A system Approach: Larry L, Peterson and Bruce S. Davie,Elsevier, 4thEd
2. Computer Networks: Natalia Olifer, Victor Olifer, Willey India
3. An Engineering Approach to Computer Networks-S.Keshav, 2nd Edition, Pearson Education
4. Computer Networking: A Top-Down Approach Featuring the Internet, James F. Kurose and Keith W. Ross , 2nd Edition, Pearson Education, 2002.

Introduction to Software Engineering

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1013	Introduction to Software Engineering	2+2+0	NIL

Course Objectives

- **Understand** the software development lifecycle models and their applications in real-world scenarios.
- **Apply** requirement engineering techniques to gather, document, and analyze user requirements effectively.
- **Develop** software using structured design methodologies, software metrics, and project estimation techniques to enhance software quality and maintainability.

Course Outcomes (Using Bloom's Taxonomy)

- **Explain** and **classify** various software process models and their relevance to specific project scenarios.
- **Apply** requirement engineering processes to collect, analyze, and document user requirements for software projects.
- **Design** and **develop** data flow diagrams (DFDs), entity-relationship (ER) diagrams, and structured charts for system analysis.
- **Evaluate** software testing techniques (e.g., black-box, white-box) and perform unit, integration, and system testing.
- **Estimate** project costs and efforts using project estimation models like COCOMO and apply software and design metrics for software quality improvement.

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2												
CO2	3	3	3	2	2								2	2	2
CO3	2		2										1	1	1
CO4	2	3	3	2									2	2	2
CO5	3	3	3	2	2								2	3	2

*High-3, Medium-2, Low-1

Course Content

Module I: Introduction to Software Engineering (8 hrs)

Theory

- Overview of software engineering.
- Software Development Life Cycle Models: Waterfall, V-Model, Prototyping, Spiral, and Iterative Waterfall Models.

Practice

1.1 Write an introduction report on software engineering principles.
1.2 Create a presentation comparing different software process models.
1.3 Demonstrate how to apply the Waterfall Model in a simple project example.
1.4 Conduct a group discussion on choosing the best process model for different project scenarios.
1.5 Prepare a project plan using the Iterative Waterfall Model for a case study.

Module II: Requirement Engineering (8 hrs)

Theory

Software Requirement Engineering: Requirement Engineering Process, Inception, Stakeholders, Requirement Elaboration.

Focus on gathering user requirements and analyzing system specifications.

Practice

2.1 Create a requirement elicitation document for a Library Management System.
2.2 Develop a list of stakeholders for an e-commerce website.
2.3 Prepare a user requirement document based on stakeholder feedback.
2.4 Write use cases to describe user interactions with an Online Hotel Reservation System.
2.5 Draft a Software Requirements Specification (SRS) for an inventory management system.

Module III: Structured Analysis & Design (10 hrs)

Theory

Introduction to Structured Analysis, Data Flow Diagrams (DFD), Process Specification, Entity Relationship (ER) Model.

Practice

3.1 Develop a Data Flow Diagram (DFD) for an Online Hotel Reservation System.
3.2 Design a personal library management system using DFD.
3.3 Create an ER diagram for an Online Hotel Reservation System.
3.4 Prepare a structured chart for an Online Hotel Reservation System.
3.5 Design an ER diagram and structured chart for a Library Management System.

Module IV: Structured Design Methodologies (8 hrs)

Theory

Structured Design Methodologies: Coupling, Cohesion, Software Testing (Black Box, White Box, Unit Testing, System Testing, Usability Testing).

Practice

4.1 Draw an ER Diagram for a Hospital Management System.
4.2 Write test cases for unit testing in an inventory management system.
4.3 Conduct black-box testing on a login module.
4.4 Perform white-box testing on a calculator application.
4.5 Create a verification and validation report for a school management system.

Module V: Software Metrics (8 hrs)

Theory

- Introduction to Software Metrics and its Classification: LOC Metrics, Function Point Metrics, Feature Point Metrics, Process Metrics.

Practice

5.1 Measure software size using LOC (Lines of Code) metrics for a small program.
5.2 Calculate function points for a simple e-commerce system.
5.3 Conduct process metrics analysis for a project workflow.
5.4 Prepare a report comparing LOC and Function Point metrics for a case study.
5.5 Present a study on how metrics improve software quality during the software development lifecycle.

Module VI: Design Metrics (8 hrs)

Theory

Design Metrics: High-Level Design Metrics, Component-Level Design Metrics, Object-Oriented Metrics (CK Metrics Suite, MOOD Metrics).

Practice

6.1 Draw a class diagram for an Online Hotel Reservation System.
6.2 Design an activity diagram for an ATM System.
6.3 Create a use case diagram for a Library Management System.
6.4 Develop an object diagram for an ATM System.
6.5 Calculate CK Metrics for a Library Management System to assess design quality.

Module VII: Project Estimation Technique (8 hrs)

Theory

COCOMO Model: Basic, Intermediate, and Complete COCOMO Models for project cost estimation.

Practice

7.1 Implement the Basic COCOMO Model for project cost estimation in a case study.
7.2 Use the Intermediate COCOMO Model for project estimation of a software project.
7.3 Apply the Complete COCOMO Model for cost estimation of a large-scale software project.
7.4 Prepare a comparative study on project estimation techniques.
7.5 Create a project cost estimation plan using COCOMO for a library management system project.

Test Book

- **"Software Engineering: A Practitioner's Approach"** by Roger Pressman
- **"Software Engineering"** by Ian Sommerville
- **"Introduction to Algorithms"** by Thomas H. Cormen

Computer System Architecture

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1014	**Computer System Architecture**	2+2+0	NIL

Course Objectives

- **Analyze** the impact of various instruction set architectures (ISAs) on processor performance and design, using high-level language constructs to explain how they are implemented at the hardware level. (Bloom's Level: Analyze)
- **Evaluate** the effectiveness of different memory organization strategies and input-output techniques in improving overall system performance and efficiency. (Bloom's Level: Evaluate)
- **Design** and **implement** simulations of fundamental computer arithmetic and digital logic circuits to demonstrate their behavior and validate theoretical concepts using design and simulation tools. (Bloom's Level: Create)

Course Outcomes

- **Calculate** the performance metrics of a digital computer system by applying parameters such as processor speed and cycles per instruction to real-world scenarios. (Bloom's Level: Apply)
- **Identify** and **describe** the components and organization of modern processor instruction sets, and explain how these components interact to execute high-level language instructions. (Bloom's Level: Understand)
- **Construct** and **formulate** assembly-level instructions from high-level imperative language constructs and translate them into the corresponding machine code. (Bloom's Level: Apply)
- **Compare** and **contrast** various types of memory hierarchies and input-output mechanisms to assess their impact on computer system performance. (Bloom's Level: Analyze)
- **Simulate** and **synthesize** basic digital logic circuits and computer arithmetic algorithms using simulation tools to validate theoretical principles and practical applications. (Bloom's Level: Apply)

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1										2	1	1
CO2	3	2	1										2	1	1
CO3	3	3	2			2							2	1	1
CO4	3	3	2	1		1							3	2	1
CO5	3	3	3	2		1							3	3	2

***High-3, Medium-2, Low-1**

Module I: Data Representation and Basic Computer Arithmetic

Theory: Understanding number systems, binary codes, fixed and floating-point representations, and basic arithmetic operations.

Practice:

1.1 Convert decimal numbers to binary, octal, and hexadecimal systems and vice versa.
1.2 Perform binary addition and subtraction using both manual methods and simulations.
1.3 Implement integer multiplication algorithms (e.g., long multiplication) in C/MATLAB and validate results.
1.4 Simulate integer division algorithms (e.g., long division) in C/MATLAB and analyze accuracy.
1.5 Represent floating-point numbers in IEEE 754 single and double precision formats and verify correctness with sample values.

Module II: Digital Logic and Circuits

Theory: Study of Boolean algebra, logic gates, circuit simplifications, and combinational and sequential circuits.

Practice:

2.1 Verify truth tables for basic logic gates (AND, OR, NOT) using simulation tools.
2.2 Design and simulate basic combinational circuits such as adders, decoders, and multiplexers using Xilinx.
2.3 Simulate and analyze the operation of flip-flops and sequential circuits like registers and counters using Xilinx.
2.4 Implement and test universal gates (NAND, NOR) using Xilinx and demonstrate their universality.
2.5 Design and synthesize a half adder and a full adder in Xilinx and verify their functionality.

Module III: Basic Computer Organization and Design

Theory: Understanding computer registers, system bus, instruction sets, and basic instruction cycles.

Practice:

3.1 Draw and label a block diagram of a basic computer system including registers, bus, and CPU.
3.2 Simulate the instruction cycle using flowcharts and validate the basic cycle operations.
3.3 Model and analyze system bus interactions in a simulation environment to understand data transfer.
3.4 Implement a basic instruction cycle flowchart and describe its steps in detail.
3.5 Simulate the behavior of different instruction sets in a controlled environment to observe their effect on system performance.

Module IV: Central Processing Unit

Theory: Study of CPU components, register organization, micro-operations, RISC vs. CISC, and pipeline and parallel architectures.

Practice:

4.1 Design and simulate a CPU block diagram, including register organization and ALU.

4.2 Analyze the micro-operations for arithmetic and logical operations and simulate their execution.
4.3 Compare RISC and CISC architectures by simulating their instruction execution and performance.
4.4 Implement a simple pipeline architecture in a simulation tool and observe its effect on instruction throughput.
4.5 Explore Flynn's classification of parallel architectures through simulation and case studies.

Module V: Instruction Design

Theory: Examination of instruction formats, addressing modes, and instruction codes.

Practice:

5.1 Design various instruction formats and addressing modes and simulate their execution.
5.2 Translate high-level imperative language instructions into assembly language and machine code.
5.3 Simulate the execution of basic computer instructions and analyze their effects on CPU performance.
5.4 Verify instruction codes for different formats and addressing modes through practical examples.
5.5 Create a set of sample instructions for different addressing modes and evaluate their efficiency.

Module VI: Memory Organization

Theory: Classification of memory types, cache memory, and memory hierarchy.

Practice:

6.1 Simulate various types of memory (primary, secondary, cache) and analyze their performance characteristics.
6.2 Implement and test different memory mapping techniques (direct, associative, set associative).
6.3 Design a memory hierarchy model and simulate its impact on system performance.
6.4 Analyze cache memory strategies and their effectiveness in reducing access times.
6.5 Evaluate the role of different types of memory in overall system design and performance.

Module VII: Input-Output Organization

Theory: Understanding I/O devices, modules, and I/O techniques including DMA.

Practice:

7.1 Study the functioning of various external I/O devices and simulate their interactions with the computer system.
7.2 Implement and test programmed I/O and interrupt-driven I/O operations in a simulation environment.
7.3 Simulate Direct Memory Access (DMA) operations and analyze their impact on system performance.

7.4 Design and implement an I/O processor and simulate its interactions with the CPU.
7.5 Describe and validate DMA transfer operations through block diagram simulations and practical examples.

Assignments

1. **Represent 525.5 in IEEE 754 Single Precision and Double Precision Formats:**
    - Perform conversion manually and verify results using a calculator.
2. **Perform 7×3 Multiplication Using Booth Algorithm:**
    - Implement Booth's algorithm in C/MATLAB and verify multiplication results.
3. **Explain 8421, 2421, 5421 BCD with Examples:**
    - Provide a detailed explanation and example for each BCD code.
4. **Draw the K-map and Simplify the Boolean Expression:**
    - Solve the given Boolean expressions using Karnaugh maps and provide simplified expressions.
5. **Prove that NAND Gate and NOR Gate are Universal Gates:**
    - Demonstrate how NAND and NOR gates can be used to construct other logic gates.
6. **Design a Mod-6 Synchronous Counter:**
    - Create a schematic for a Mod-6 synchronous counter and simulate its operation.
7. **Describe Flowchart for Instruction Cycle:**
    - Provide a detailed flowchart and description of the instruction cycle.
8. **Describe 16-Logical Microoperation and the Logic Circuit:**
    - Explain and illustrate 16 types of logical micro-operations and their associated logic circuits.
9. **Describe Flynn's Classification of Parallel Architecture:**
    - Provide an overview of Flynn’s classification and examples of each category.
10. **Describe DMA Transfer Operation with Block Diagram:**
    - Illustrate and explain DMA transfer operations with a detailed block diagram.

Text Books

1. M.Mano, Computer System Architecture, Pearson Education, 1992
2. Carl Hamacher, Computer Organization, McGrawHill, 2012.
3. William Stalling,Computer Organization and Architecture Designing for performance, Prentice Hall of India, 2009

Android App Development

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1015	Android App Development	1+2+1	Nil

Course Objectives

- Understand the basics of Android development and the Android Studio environment.
- Design and develop user interfaces, implement data storage, and manage network communication for Android applications.
- Deploy and manage Android applications

Course Outcomes

- CO1: Recall the fundamental concepts of Android development. (Remembering)
- CO2: Explain the Android app components and lifecycle. (Understanding)
- CO3: Apply user interface design principles to develop Android applications. (Applying)
- CO4: Analyze data storage and network communication techniques. (Analyzing)
- CO5: Develop and deploy comprehensive Android applications. (Creating)

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	1	1										2		1
CO2	2	2	3	1	3								2	2	1
CO3	2	3	3	2	3								3	3	
CO4	2	3	3	2	3								3	3	
CO5	3	3	3	2	3								3	3	2

*High-3, Medium-2, Low-1

Course Content

Module 1: Introduction to Android Development (10 hours)

Theory

- Overview, Android Architecture
- Setting up Android Studio, Kotlin Basics
- Creating First Android Project

Practice

1.1: Set up Android Studio and create a new project.
1.2: Write a Kotlin program to print "Hello, World!".
1.3: Implement basic arithmetic operations in Kotlin.
1.4: Create a simple Android app with a single activity.
1.5: Implement a simple calculator app in Android.

Module 2: Android App Components (12 hours)

Theory

- Activities, Fragments, Services
- Broadcast Receivers, Content Providers
- Intents, Activity Lifecycle

Practice

2.1: Implement an activity lifecycle in Android.
2.2: Create and use fragments in an Android app.
2.3: Implement a service to run in the background.
2.4: Use broadcast receivers to receive system events.
2.5: Implement content providers to share data between apps.

Module 3: User Interface Design (10 hours)

Theory

- Layouts: Linear, Relative, Constraint, Frame
- UI Components: TextView, EditText, Button, ImageView, RecyclerView
- Material Design Principles

Practice

3.1: Design a user interface using ConstraintLayout.
3.2: Implement a RecyclerView to display a list of items.
3.3: Create custom views and view groups.
3.4: Implement user input controls (TextView, EditText, Button).
3.5: Implement a navigation drawer for app navigation.

Module 4: Data Storage (10 hours)

Theory

- SharedPreferences, Internal, External Storage

- SQLite Database, Room Persistence Library
- Content Providers

Practice

4.1: Implement SQLite database in an Android app.
4.2: Use shared preferences for simple data storage.
4.3: Implement internal storage for file management.
4.4: Implement external storage for file management.
4.5: Implement data binding in an Android app.

Module 5: Network Communication (10 hours)

Theory

- HTTP Networking, RESTful APIs
- Retrofit, Volley Libraries
- JSON Parsing, Background Tasks with WorkManager

Practice

5.1: Implement HTTP networking using Retrofit.
5.2: Implement HTTP networking using Volley.
5.3: Parse JSON data from a web API.
5.4: Use OkHttp for network communication.
5.5: Implement RESTful API communication.

Module 6: Advanced Topics (12 hours)

Theory

- Firebase Integration: Authentication, Realtime Database
- Google Play Services, Location APIs
- App Distribution, Google Play Store, App Signing

Practice

6.1: Implement Firebase Authentication in an Android app.
6.2: Use Google Maps API for location-based services.
6.3: Implement push notifications using Firebase Cloud Messaging.
6.4: Integrate social media login (Facebook, Google) in an Android app.
6.5: Use Google Play Services for analytics.

Project (10 hours)

- Project 1: Develop a personal finance management app.
- Project 2: Create a location-based reminder app.
- Project 3: Develop a social media integration app.

Text Books:

1. "Android Programming with Kotlin: A Hands-on Guide to Building Android Apps" by Ashok L. Ahuja

2. "Head First Android Development" by Dawn Griffiths & David Griffiths

Reference Books:

1. "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips, Chris Stewart, & Kristin Marsicano
2. "Android Application Development with Kotlin" by Anmol Gupta

Information Security

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1016	Information Security	1+2+1	Nil

Course Objectives

- Understand the foundational principles of information security, including key terminologies, historical developments, and the need for legal and ethical measures.
- Apply risk management and cryptography techniques to design secure systems and protect sensitive data in compliance with industry standards.
- Analyze various security models, protocols, and tools to develop and evaluate comprehensive solutions for real-world security challenges.

Course Outcomes

- Define and explain basic concepts and terminologies in information security, including the importance of safeguarding information (Knowledge, Understanding).
- Apply risk management frameworks and standards (such as TEMPEST, Firewalls, VPNs) to ensure system security (Application).
- Implement cryptographic algorithms (like DES, AES, public-key cryptosystems) in real-world scenarios to secure information (Application, Analysis).
- Evaluate different security models, such as Kerberos and distributed authentication systems, to maintain integrity and confidentiality (Evaluation).
- Develop secure software systems using secure programming languages and protocols to address network vulnerabilities and protect data (Synthesis, Evaluation).

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	1	1										2		1
CO2	2	2	3	2									2	2	1
CO3	2	3	3	3	1								3	3	
CO4	2	3	3	3	2								3	3	
CO5	3	3	3	2	1								3	3	2

*H*High-3, Medium-2, Low-1

Course Content

Module 1: (10 hours)

Theory

Introduction, Definition of security, Assessing security, Security terminology, Historical developments, Structure of security, Introduction to Information Security, The Need for Security , Legal, Ethical, and Professional Issues in Information Security

Practice

1. 1Analyze historical security breaches.
1. 2Compare legal and ethical issues in security.
1. 3Conduct a vulnerability assessment on a sample system.
1. 4Discuss the importance of security in modern-day organizations.
1. 5Review and critique security terminologies used in academic and professional settings.

Module 2:(12 hours)

Theory

Risk Management and Special requirements such as Emanation Security/TEMPEST Standards, Planning for Security, Rainbow Series Reports for DOD; FIPS for all Federal Govt; DHS and CNSS guidance: Firewalls & VPNs

Practice

2.1.· Simulate risk assessment on different organizational scenarios.
2.2.· Explore the application of DOD's Rainbow Series Reports.
2.3.· Configure firewalls and VPNs in a virtual environment.
2.4.· Compare FIPS and DHS security guidance.
2.5.· Analyze case studies on firewalls in real-world organizations.

Module 3: (10 hours)

Theory

cryptography: Applications of cryptography, Terminology, Evolution of cryptography, Caesar ciphers, one-time pads, Operation of DES, AES ,Public-key cryptosystems, Topics in Information Systems Security, Minimum privilege ,Compartmentalization , Dual controls ,Security perimeters, Trustworthy software, proof of design correctness, Single-points-of-failure, Covert channels, Inference

Practice

3.1.· Demonstrate Caesar ciphers and one-time pads using coding exercises.
3.2.· Apply DES and AES encryption on datasets.
3.3.· Explore the evolution of cryptography through research papers.
3.4.· Simulate public-key cryptosystems in software.
3.5.· Analyze security perimeters in compartmentalized systems.

Module 4:(10 hours)

Theory

technology: IDS and Access Control Cryptography, Physical Security including Emanations Security , (EMSEC)/TEMPEST/CFR 32 Marking, Handling, labeling and destruction of Sensitive information) Implementing Security, Security and Personnel, InfoSec Maintenance

Practice

4.1.· Simulate an IDS setup in a virtual network.
4.2.· Explore access control mechanisms in secure systems.
4.3.· Conduct a physical security audit using EMSEC guidelines.
4.4.· Examine cryptography handling practices in real-time.
4.5.· Research security measures for personnel in a corporate setting.

Module 5: (10 hours)

Theory

Security models: Requirements ;Types, State-machine models, Mandatory/Discretionary controls, Information-flow models, Informal models, Kerberos Authentication, Authentication in centralized systems, Distributed Authentication

Practice

5.1.· Implement a state-machine security model in a controlled lab environment.
5.2.· Analyze discretionary vs. mandatory controls through role-playing exercises.
5.3.· Research and present on Kerberos authentication in centralized systems.
5.4.· Simulate distributed authentication in a cloud environment.
5.5.· Critique various information-flow models used in security.

Module 6: (12 hours)

Theory

Denial of Service attacks, Security vs. ATM, IP, wireless mobile networks, QoS, Traffic modeling, Network topology , Security Protocols, Zero-knowledge proofs, Subliminal channels,

Oblivious transfer, Digital signature schemes, Bit commitment, Digital cash, Secure contract signing, Secure voting, Digital certified mail, Anonymous message broadcast TEMPEST and related topics

Practice

6.1.· Simulate a Denial of Service attack and analyze its impact.
6.2.· Compare security protocols across different network environments (ATM, IP, wireless).
6.3.· Implement zero-knowledge proofs in a cryptography lab.
6.4.· Research subliminal channels and their implications in security.
6.5.· Explore digital signature schemes and simulate secure contract signing.

Module 7: (10 Hours)

Secure programming languages- concepts structured multiprogramming, shared classes, cooperating sequential processes, structure of te multiprogramming system RC-4000 software.

Practice

7.1.· Develop basic programs using secure programming languages.
7.2.· Explore structured multiprogramming systems and shared classes.
7.3.· Research RC-4000 software and its application in secure programming.
7.4.· Implement concepts of cooperating sequential processes in software development.
7.5.· Conduct a peer review of secure programming practices in a team project.

Test Book

1. **Information Security: Principles and Practice** by Mark Stamp
2. **Computer and Information Security Handbook**, Third Edition
3. **Network Security: Books** (Amazon)

Cloud Computing

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1017	Cloud Computing	1+2+1	Nil

Course Objectives:

- To Learn the core concepts and benefits of cloud computing, including knowledge about cloud service providers and virtualization.
- To get started with cloud services and manage cloud resources and explore cloud storage, networking, and security.
- To develop and deploy applications on the cloud.

Course Outcomes:

- **CO1**: Identify and remember the essential concepts and benefits of cloud computing. (Remembering)
- **CO2**: Describe and explain different types of cloud storage and the basics of cloud networking. (Understanding)
- **CO3**: Utilize and implement cloud services to deploy applications on cloud platforms. (Applying)
- **CO4**: Examine and evaluate strategies for optimizing cloud resource usage and managing costs. (Analyzing)
- **CO5**: Design and build scalable applications for cloud environments. (Creating)

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2												
CO2	3	3	3	2	2								2	2	2
CO3	2	2	2										1		1
CO4	2	3	3	2	2								2	2	2
CO5	3	3	3	2	2								2	3	3

*H***High-3, Medium-2, Low-1**

Course Syllabus:

Module 1: Foundations of Cloud Computing

- Overview, Characteristics, Advantages
- Service Models: IaaS, PaaS, SaaS
- Deployment Models: Public, Private, Hybrid, Community Cloud

Module 2: Cloud Service Providers and Virtualization

- Major Cloud Service Providers: AWS, Azure, Google Cloud
- Virtual Machines vs. Containers, Benefits of Virtualization
- Hypervisors: Type 1, Type 2

Module 3: Getting Started with Cloud Services

- Setting up Cloud Accounts, Accessing Cloud Console/Portal
- Creating Virtual Machines, Storage Buckets
- Managing Permissions, Security Groups

Module 4: Cloud Storage and Networking

- Cloud Storage Services, Creating, Managing Storage Buckets
- Object Storage vs. Block Storage
- Networking Basics: VPC, Subnets, Security Groups

Module 5: Deploying Applications on Cloud

- Deployment Options: Virtual Machines, Containers, Serverless
- Containerization with Docker, Serverless Computing with AWS Lambda, Google Cloud Functions

Module 6: Managing Cloud Resources

- Monitoring Resources, Scaling Automatically
- Backing Up Data in the Cloud
- Cost Management: Billing Metrics, Cost Allocation Tags

Module 7: Cloud Security and Compliance

- Shared Responsibility Model, Best Practices for Securing Cloud Resources
- Identity and Access Management (IAM)
- Compliance Standards, Regulations in the Cloud

Practicals:

- Lab 1: Setting up Cloud Accounts and Creating VMs
- Lab 2: Cloud Storage and File Uploads
- Lab 3: Deploying Web Applications on VMs
- Lab 4: Docker Containerization
- Lab 5: Serverless Computing with AWS Lambda
- Lab 6: Cloud Resource Monitoring and Scaling
- Lab 7: Implementing Cloud Security Policies

Projects:

- Project 1: Scalable Web Application Deployment on Cloud
- Project 2: Cloud-Based Backup Strategy for Small Business
- Project 3: Implementing Secure Cloud Environment for E-Commerce Platform

Test Book

1. **Cloud Computing: Concepts, Technology & Architecture** by Zaigham Mahmood, Ricardo Puttini, and Thomas Erl:
2. The Basics of Cloud Computing: Understanding the Fundamentals of Cloud Computing in Theory and Practice by Derrick Rountree and Ileana Castrillo

Internet of Things

Code	Course Title	T-P-Pj (Credit)	Prerequisite
CUBC1018	Cloud Computing	1+2+1	Nil

Prerequisites: Basic knowledge of computer networks and programming

Course Objectives

- Understand the core concepts of the Internet of Things, its architecture, protocols, and applications across different domains.
- Develop skills to design IoT solutions using sensors, actuators, and microcontrollers such as Raspberry Pi and Arduino.
- Apply IoT frameworks and communication technologies to solve real-world problems and build connected systems.

Course Outcomes

- CO1: Recall the fundamental concepts and architecture of IoT (Remembering).
- CO2: Explain the protocols, sensors, and communication technologies used in IoT systems (Understanding).
- CO3: Design and implement IoT solutions using microcontrollers and sensors (Applying).
- CO: Analyze IoT systems for connectivity, scalability, and security (Analyzing).
- CO5: Evaluate and optimize the performance of IoT applications for specific use cases (Evaluating)

Course Outcome to Program Outcome Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	**2**	**1**	**1**										**2**		**1**
CO2	**2**	**2**	**3**	**2**									**2**	**2**	**1**
CO3	**2**	**3**	**3**	**3**	**1**								**3**	**3**	
CO4	**2**	**3**	**3**	**3**	**2**								**3**	**3**	
CO5	**3**	**3**	**3**	**2**	**1**								**3**	**3**	**2**

*H***High-3, Medium-2, Low-1**

Course Content

Module 1: Introduction to IoT (10 hours)

Theory

- Overview of IoT, applications in various domains (healthcare, agriculture, smart cities, etc.).
- IoT architecture: Edge, Fog, and Cloud Computing.
- Characteristics of IoT devices.
- Types of IoT communication models: Device-to-Device (D2D), Device-to-Cloud, etc.
- IoT enabling technologies: RFID, WSN, Big Data, AI, Cloud.

Practice

1.1. Explore real-world IoT applications in smart homes and smart cities.
1.2. Set up a basic IoT system using Arduino to blink an LED.
1.3. Implement Device-to-Device communication between two IoT devices using Bluetooth.
1.4. Create a cloud-based system to log sensor data from a temperature sensor.
1.5. Set up a Raspberry Pi and connect it to a cloud platform like AWS IoT.

Module 2: IoT Communication Protocols (12 hours)

Theory

- IoT network protocols: MQTT, CoAP, HTTP, AMQP.
- Comparison between protocols for different use cases.
- Wireless communication: Zigbee, LoRa, NB-IoT, BLE, Wi-Fi.
- Data formats: JSON, XML.
- Message security protocols in IoT: DTLS, TLS, IPsec.

Practice

2.1. Implement MQTT communication between IoT devices.
2.2. Compare CoAP and HTTP for a sensor-based IoT system.
2.3. Set up a Wi-Fi-based sensor system using ESP8266 or ESP32.
2.4. Implement LoRa-based long-range communication between devices.
2.5. Secure MQTT communication using TLS in a Raspberry Pi environment.

Module 3: IoT Hardware Platforms (10 hours)

Theory

- Overview of IoT hardware platforms: Arduino, Raspberry Pi, ESP8266/ESP32.

- Sensor and actuator integration with microcontrollers.
- GPIO programming for hardware control.
- Power management in IoT devices.
- IoT development environments (Arduino IDE, Python for Raspberry Pi)

Practice

3.1. Set up an Arduino board and interface a DHT11 temperature and humidity sensor.
3.2. Control a servo motor using GPIO pins on a Raspberry Pi.
3.3. Build a basic home automation system using a PIR motion sensor.
3.4. Implement IoT projects using ESP32 and Wi-Fi connectivity.
3.5. Monitor battery life and optimize power consumption for IoT devices

Module 4: IoT Sensors and Actuators (10 hours)

Theory

- Types of sensors (temperature, humidity, proximity, accelerometers).
- Actuators (motors, relays, LEDs) in IoT systems.
- Sensor interfacing techniques (analog, digital).
- Real-time data acquisition from sensors.
- Calibration and filtering techniques for sensor data.

Practice

4.1. Interface a temperature sensor with an Arduino and display the readings on an LCD.
4.2. Connect multiple sensors to a single Raspberry Pi using I2C protocol.
4.3. Control home appliances using relays and an IoT app.
4.4. Build a weather monitoring station using sensors for temperature, humidity, and pressure.
4.5. Implement filtering techniques for noise reduction in sensor data.

Module 5: IoT Data Management and Analytics (10 hours)

Theory

- IoT data management: Data collection, storage, and analysis.
- Data streaming and real-time processing.
- IoT analytics: Predictive analytics, decision-making, anomaly detection.
- Cloud platforms for IoT (AWS IoT, Google Cloud IoT, Microsoft Azure).
- Data visualization tools for IoT: Grafana, Tableau.

Practice

5.1. Collect and visualize real-time data from a sensor using Grafana.
5.2. Store IoT sensor data in a cloud database and retrieve it for analysis.
5.3. Implement real-time IoT data streaming and processing using Apache Kafka.
5.4. Perform predictive analytics on historical IoT data using Python.
5.5. Integrate an IoT device with AWS IoT and visualize the data on the AWS dashboard

Module 6: Security in IoT Systems (12 hours)

Theory

- Security challenges in IoT systems.
- Authentication and authorization mechanisms.
- Encryption techniques: Symmetric and asymmetric encryption.
- Threat modeling in IoT networks.
- Secure IoT communication protocols (SSL/TLS, DTLS).

Practice

6.1. Implement basic encryption techniques to secure sensor data transmission.
6.2. Set up an SSL/TLS-based communication channel between IoT devices.
6.3. Develop an authentication system for IoT devices using OAuth2.
6.4. Perform a security vulnerability assessment of an IoT network.
6.5. Simulate a man-in-the-middle attack on an IoT system and develop mitigation strategies

Project (10 hours)

- Smart home automation.
- IoT in healthcare (wearable devices, remote monitoring).
- Industrial IoT (IIoT) for automation and predictive maintenance.
- Smart agriculture using IoT sensors.

Test Book

1. **Internet of Things: A Beginner's Guide** by Adrian McEwen
2. **Getting Started with Internet of Things** by Cuno Pfister

Machine Learning using Python (98 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUTM1019	Machine Learning using Python	4	1-2-1

Course Description: This course deals with various machine learning algorithms, strategies for model generation and evaluations are covered as per the industry requirement.

Course Objectives:

- Understand the meaning, purpose, scope, stages, applications, and effects of ML.
- Explore important packages of python, such as numpy, scipy, OpenCV and scikit-learn.
- To apply and design ML algorithms on given data and interpret the results obtained

Course Outcomes (COs):

- **CO1**: Develop a good understanding of fundamental principles of machine learning (Understand, Create)
- **CO2**: Invention of a Machine Learning problem. (Create)
- **CO3**: Develop a model using supervised/unsupervised machine learning algorithms for classification/prediction/clustering. (Design)
- **CO4**: Design and Concrete implementations of various machine learning algorithms to solve a given problem using languages such as Python. (Create)
- **CO5**: Evaluate performance of various machine learning algorithms on various datasets of a domain. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	1	1		-	-	2	2	2	3	3	3
CO2	3	3	2	2	1	-	-	-	2	2	3	3	3
CO3	3	3	3	2	2	-	-	2	-	2	3	3	3
CO4	3	3	3	3	3	-	-	2	2	2	3	3	3
CO5	3	3	3	3	1	-	-	2	2	3	3	3	3

***High-3, Medium-2, Low-1**

Course Syllabus:

Module 1: Application and Environmental-setup (24 hrs)

- Applications of Machine Learning In different fields (Medical science, Agriculture, Automobile, mining and many more).
- Supervised vs Unsupervised Learning based on problem Definition.
- Understanding the problem and its possible solutions using IRIS datasets.
- Python libraries suitable for Machine Learning(numpy, scipy, scikit-learn, opencv)
- Environmental setup and Installation of important libraries.

Module 2: Regression (24 hrs)

- Linear Regression
- Non-linear Regression
- Model Evaluation in Regression
- Evaluation Metrics in Regression Models
- Multiple Linear Regression
- Feature Reduction using PCA
- Implementation of regression model on IRIS datasets.

Module 3: Classification (26 hrs)

- Defining Classification Problem with IRIS datasets.
- Mathematical formulation of K-Nearest Neighbour Algorithm for binary classification.
- Implementation of K-Nearest Neighbour Algorithm using sci-kit learn.
- Classification using Decision tree.
- Construction of decision trees based on entropy.
- Implementation of Decision Trees for Iris datasets .
- Classification using Support Vector Machines.
- SVM for Binary classification
- Regulating different functional parameters of SVM using sci-kit learn.
- SVM for multi class classification.
- Implementation of SVM using Iris datasets .
- Implementation of Model Evaluation Metrics using sci-kit learn and IRIS datasets.

Module 4 - Unsupervised Learning (24 hrs)

- Defining clustering and its application in ML .
- Mathematical formulation of K-Means Clustering.
- Defining K value and its importance in K-Means Clustering.
- Finding appropriate K value using elbow technique for a particular problem.
- Implementation of K-Means clustering for IRIS datasets

Projects

- To be defined based on respective study area of student.

References:

Text Book:

1. EthemAlpaydin, Introduction to Machine Learning, Second Edition, http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=12012.

Web Resource:

https://towardsdatascience.com/beginners-guide-to-machine-learning-with-python-b9ff35bc9c51

Major Project

Code	Course Title	Credit	T-P-PJ
CUBC1019	**Major Project**	4	1-2-1

- To equip students with a comprehensive understanding of project concepts, technologies, and applications.
- To enable students to design, develop, and implement technological solutions that address real-world problems.
- To foster critical thinking, problem-solving, and teamwork skills in the context of projects.

Course Outcomes

- **Knowledge and Understanding:** Students will be able to define and explain key concepts of project
- **Application:** Students will be able to apply project principles to design and develop innovative solutions for various domains, such as smart homes, agriculture, and healthcare.
- **Analysis:** Students will be able to analyze system requirements, identify potential challenges, and evaluate the feasibility of projects.
- **Synthesis:** Students will be able to integrate various components and technologies to create functional and efficient systems.
- **Evaluation:** Students will be able to critically assess the performance of systems, identify areas for improvement, and propose solutions to optimize system performance.

CO-PO-PSO Mapping:

CO/PO/PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3	2	3	-	-	-	2	3	2	2	3	3	3
CO2	3	2	3	3	3	-	-	-	2	3	2	2	3	3	3
CO3	3	2	3	3	3	-	-	-	2	2	2	2	3	3	3
CO4	3	2	3	3	3	-	-	-	3	2	2	2	3	3	3
CO5	3	2	3	3	3	-	-	-	3	3	3	3	3	3	3

***High-3, Medium-2, Low-1**