COURSE STRUCTURE AND SYLLABI

Master of Computer Application (MCA)

2024-25 Batch

Centurion UNIVERSITY

Shaping Lives...
Empowering Communities...

SCHOOL OF APPLIED SCIENCE
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 Master of Computer Application (MCA) of the School of Applied Science is approved in the 14th Academic Council Meeting held on 22nd November 2024.

Dean
School of Applied Science,
CUTM, Odisha

Centurion University of Technology and Management, Odisha

Course Structure & Syllabus

Master of Computer Application

(MCA)

2-Years Programme

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

School of Engineering & Technology

2024-2025

Centurion University
Shaping Lives... Empowering Communities...

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
CO6	Environment and Sustainability
CO7	Ethics & Team work
CO8	Soft skill

Centurion University
Shaping Lives... Empowering Communities...

Course Structure

Master of Computer Application

SEMESTER - I				
Sl No	**Course Code**	**Name of the Course**	**Credits**	**T+P+J**
1	CUCS1001	Programming in C	6	2+4+0
2	CUCS1006	Network and Protocols for IoT	3	1+2+0
3	CUTM1018	Data Analysis and Visualization using Python	4	0+1+3
4	CUCS1011	Software Engineering and Testing	3	1+2+0
5	CUCS1005	Relational and Distributed Databases	4	2+2+0
6	CUCS1012	Customer Experience Design and Programming	4	1+2+1
7	CUTM1016	Job Readiness	6	0+6+0
		Total	**30**	
SEMESTER – II				
8	CUTM1019	Machine Learning using Python	4	1+2+1
9	CUCS1010	Cloud Practitioner (AWS)	2	1+1+0
10	CUCS1004	Java Programming	6	2+2+2
11	CUCS1002	Data Structures with Competitive Coding	6	2+4+0
12	CUCS1007	Information Security (CISCO)	3	1+1+1
		Total	**21**	
SEMESTER – III & IV is CBCS based (Domains)				
13	CUCS1003	Design and Analysis of Algorithms	6	2+4+0
14	CUCS1015	Cloud Fundamentals (Azure)	2	1+1+0
15	CUCS1013	Android Development with Kotlin	6	1+3+2
16	CUCS1014	Prompt Engineering using ChatGPT	2	1+1+0
17	CUCS1009	System Administrator (RedHat)	3	2+1+0
18	CUTM2474	DotNet Programming	4	0+2+2
Domain Details				
1	**FDCU1000**	**Full-Stack Development with MERN**	**18**	**1+8+9**
	CUFD1001	MongoDB for Developers	3	1+2+0
	CUFD1002	Node.js and Express.js Development	3	0+2+1
	CUFD1003	Front-End Development with React	3	0+2+1
	CUFD1004	Full Stack Integration and Deployment	3	0+2+1
	CUFD1005	Product Development	6	0+0+6
2	**CSCU1100**	**Cyber Security**	**20**	**8+8+4**
	CUCS1101	Linux Server Management and Security	4	2+2+0
	CUCS1102	Offensive Security	4	2+2+0
	CUCS1103	Defensive Security	4	2+2+0
	CUCS1104	Security Analytics	4	2+2+0

	CUCS1105	Project	4	0+0+4
3	**GACU1010**	**Generative AI**	**12**	**0+8+4**
	CUGA1011	Fundamentals of Generative AI	1	0+1+0
	CUGA1012	Advanced Techniques in Generative AI	2	0+2+0
	CUGA1013	Real-Time Generative AI	3	0+1+2
	CUGA1014	Research and Development in Generative AI	2	0+2+0
	CUGA1015	Capstone Project in Generative AI	4	0+2+2
4	**MLCU1020**	**Data Analytics and Machine Learning**	**18 + 4 (Optional)**	**0+6+12**
	CUML1021	Machine Learning for Predictive Analytics	4	0+2+2
	CUML1022	Deep Learning for Image Analytics	4	0+2+2
	CUML1023	Data Analytics using Tableau	4	0+2+2
	CUML1024	ML for Spectral Imaging (Optional)	4	0+2+2
	CUML1025	Project	6	0+0+6
5	**CTCU1030**	**Cloud Technology**	**12**	**0+6+6**
	CUCT1031	Advanced Cloud Architecture and Design	3	0+2+1
	CUCT1032	Cloud Development and DevOps	3	0+2+1
	CUCT1033	Cloud Security and Compliance	3	0+2+1
	CUCT1034	Capstone Project	3	0+0+3
6	**DSCU1040**	**Drone Imaging and Spectral Analysis**	**12**	**0+6+6**
	CUDS1041	Drone Image Processing using Pix4D	3	0+2+1
	CUDS1042	Multispectral Image Analytics for Agriculture	3	0+2+1
	CUDS1043	Drone Imaging Applications	2	0+2+0
	CUDS1044	Domain Project	4	0+0+4
7	**BDCU1090**	**Blockchain Development**	**18**	**0+7+11**
	CUBD1091	INTRODUCTION TO BLOCKCHAIN	2	0+2+0
	CUBD1092	CRYPTOCURRENCIES AND SMART CONTRACTS	3	0+2+1
	CUBD1093	BLOCKCHAIN DEVELOPMENT	3	0+1+2
	CUBD1094	WEB3 AND DECENTRALIZED TECHNOLOGIES	3	0+1+2
	CUBD1095	ADVANCED BLOCKCHAIN CONCEPTS AND DEVELOPMENT	3	0+1+2
	CUBD1096	CAPSTONE PROJECT IN BLOCKCHAIN DEVELOPMENT	4	0+0+4
8	**CBCU1060**	**Computational Biology**	**12**	**2+7+3**
	CUCB1061	Introduction to Computational Biology	3	1+2+0
	CUCB1062	Computational Genomics	3	0+3+0
	CUCB1063	Computational Systems Biology	3	1+2+0
	CUCB1064	Computational Proteomics	3	0+0+3
9	**MACU1070**	**Mobile App Development**	**12**	**1+6+5**

	CUMA1071	Introduction to Mobile App Development	3	1+2+0
	CUMA1072	React Native Development	3	0+2+1
	CUMA1073	Flutter Development	3	0+2+1
	CUMA1074	Advanced Mobile App Development Project	3	0+0+3
10	**GICU1080**	**Gaming and Immersive Learning-AR/VR**	**20**	5+5+10
	CUGI1081	Introduction to Gaming & Simulation	2	1+1+0
	CUGI1082	Game Assets and Objects	3	1+1+1
	CUGI1083	Building Game Environment	3	1+1+1
	CUGI1084	Game Animation, Scripting & UI	3	1+1+1
	CUGI1085	Binary Deployment and Cross-Platform Controls	3	1+1+1
	CUGI1086	Project	6	0+0+6
11	**STCU1050**	**Software Technology**	**18**	**0+6+12**
	CUST1051	Advanced Java	4	0+2+2
	CUST1052	Angular	4	0+2+2
	CUST1053	Spring Boot	4	0+2+2
	CUST1054	Product Development	6	0+0+6

Semester – I

Programming in C (140 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1001	Programming in C	6	2+4+0

Course Description:

This course provides an introduction to programming using the C language, a foundational language that has influenced many other programming languages and is widely used in software development. Students will learn the core concepts of programming including syntax, data types, control structures, functions, arrays, pointers, and dynamic memory management.

Course Objectives:

- To gain a foundational understanding of the syntax and data types in C programming, and achieve proficiency in handling input/output using standard functions.
- Learn to use operators and expressions effectively and implement control structures in C programs.
- To enhance problem-solving abilities by working with functions, recursion, arrays, strings, pointers, structures, and files.

Course Outcomes:

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

- **CO1:** Define and record the fundamental syntax and data types used in the C programming language. (Remembering)
- **CO2:** Describe and illustrate the usage of input/output functions and operators in C programming. (Understanding)
- **CO3:** Demonstrate various operators and control structures to construct functional C programs. (Applying)
- **CO4:** Examine and break down problems appropriate functions and recursion techniques for solution development. (Analysing)
- **CO5:** Design and evaluate sophisticated algorithms and advanced C programs incorporating arrays, strings, pointers, structures, and unions. (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	1	-	1	-	-	-	-	-	1	-	2	1	1

CO2	**3**	**2**	**1**	**-**	**-**	**-**	**-**	**1**	**-**	**-**	**-**	**-**	**2**	**1**	**1**
CO3	**3**	**3**	**2**	**-**	**-**	**2**	**1**	**-**	**-**	**-**	**-**	**-**	**2**	**1**	**1**
CO4	**3**	**3**	**2**	**1**	**-**	**1**	**-**	**-**	**1**	**-**	**-**	**1**	**3**	**2**	**1**
CO5	**3**	**3**	**3**	**2**	**-**	**1**	**-**	**-**	**-**	**1**	**-**	**-**	**3**	**3**	**2**

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

Course Syllabus:

Module 1: Basic Syntax and Control Structures (20 hours)

Theory

- Data Types and Variables
- Basic Data Types: int, char, float, double
- Variable Declaration and Initialization
- Constants and Literals
- Input/Output: printf(), scanf(), gets(), puts()
- Operators and Expressions: Arithmetic, Relational, Logical, Bitwise, Assignment, Conditional (Ternary)

Practice

- Experiment 1.1: Write a C program to demonstrate the use of variables and constants.
- Experiment 1.2: Implement basic arithmetic operations using C.
- Experiment 1.3: Develop a program to showcase the use of relational and logical operators.
- Experiment 1.4: Write a program to perform input and output operations.
- Experiment 1.5: Demonstrate the use of conditional operator in a C program.
- Experiment 1.6: Implement a program using bitwise operators.
- Experiment 1.7: Write a C program to convert temperature from Celsius to Fahrenheit.
- Experiment 1.8: Develop a program to swap two numbers without using a temporary variable.
- Experiment 1.9: Write a program to find the maximum and minimum of three numbers.
- Experiment 1.10: Implement a program to calculate the area of a circle.

Module 2: Control Statements (20 hours)

Theory

- if, if-else, nested if-else
- switch-case
- for loop, while loop, do-while loop
- break, continue, goto, return

Practice

- Experiment 2.1: Write a program using if-else statement to find the largest of three numbers.
- Experiment 2.2: Implement a C program using switch-case to perform arithmetic operations.
- Experiment 2.3: Develop a program to print the first 10 natural numbers using for loop.
- Experiment 2.4: Write a C program to calculate the factorial of a number using while loop.
- Experiment 2.5: Implement a program to reverse a given number using do-while loop.
- Experiment 2.6: Develop a program to check whether a number is prime or not.
- Experiment 2.7: Write a C program to print Fibonacci series up to n terms.
- Experiment 2.8: Implement a program to check whether a number is palindrome or not.

Module 3: Functions and Recursion (30 hours)

Theory

- Function Declaration, Definition, Call
- Parameter Passing: Pass by Value, Pass by Reference
- Scope and Lifetime of Variables
- Inline Functions, Function Pointers
- Recursion: Basics, Examples (Factorial, Fibonacci, Tower of Hanoi)

Practice

- Experiment 3.1: Write a program to demonstrate the use of functions.
- Experiment 3.2: Implement a program using recursion to find the factorial of a number.
- Experiment 3.3: Develop a program to calculate the GCD of two numbers using recursion.
- Experiment 3.4: Write a program to find the nth Fibonacci number using recursion.
- Experiment 3.5: Implement a program to demonstrate pass by value and pass by reference.
- Experiment 3.6: Develop a program using function pointers.
- Experiment 3.7: Write a C program to find the sum of an array using recursion.
- Experiment 3.8: Implement a program to solve the Tower of Hanoi problem

Module 4: Arrays and Strings (20 hours)

Theory

- One-Dimensional Arrays, Multidimensional Arrays
- Array as Function Arguments
- Strings: Declaration, Initialization
- String Manipulation Functions: strcpy(), strcat(), strcmp(), strlen(), etc.
- Array of Strings

Practice

- Experiment 4.1: Write a C program to implement linear search in an array.
- Experiment 4.2: Implement a program to sort an array using bubble sort.
- Experiment 4.3: Develop a program to perform matrix addition.
- Experiment 4.4: Write a program to perform matrix multiplication.
- Experiment 4.5: Implement a program to reverse a string.
- Experiment 4.6: Develop a program to check whether a string is palindrome.
- Experiment 4.7: Write a C program to concatenate two strings without using library functions.
- Experiment 4.8: Implement a program to find the length of a string using a user-defined function.

Module 5: Pointers (20 hours)

Theory

- Pointer Declaration, Initialization
- Pointer Arithmetic
- Pointers and Arrays, Strings
- Advanced Pointers: Pointers to Pointers, Function Pointers
- Dynamic Memory Allocation: malloc(), calloc(), realloc(), free()

Practice

- Experiment 5.1: Write a C program to demonstrate pointer arithmetic.
- Experiment 5.2: Implement a program to swap two numbers using pointers.
- Experiment 5.3: Develop a program to reverse an array using pointers.
- Experiment 5.4: Write a C program to find the length of a string using pointers.
- Experiment 5.5: Implement a program to copy one string to another using pointers.
- Experiment 5.6: Develop a program to dynamically allocate memory for an array.
- Experiment 5.7: Write a C program to demonstrate the use of function pointers.
- Experiment 5.8: Implement a program to demonstrate double pointer.

Module 6: Structures and Unions (10 hours)

Theory

- Defining and Declaring Structures, Accessing Members
- Array of Structures, Passing Structures to Functions, Pointers to Structures
- Defining and Declaring Unions, Difference between Structures and Unions

Practice

- Experiment 6.1: Write a C program to define and use a structure.
- Experiment 6.2: Implement a program to demonstrate array of structures.
- Experiment 6.3: Develop a program to pass structure to a function.
- Experiment 6.4: Write a C program to demonstrate the use of pointers to structures.
- Experiment 6.5: Implement a program to define and use a union.
- Experiment 6.6: Develop a program to demonstrate the difference between structures and unions.
- Experiment 6.7: Write a C program to implement a student record system using structures.

- Experiment 6.8: Implement a program to calculate the size of a structure using sizeof operator.

Module 7: Advanced Topics (20 hours)

Theory

- File I/O: fopen(), fclose(), fread(), fwrite(), fprintf(), fscanf()
- Error Handling in File Operations
- Bitwise Operations and Preprocessor Directives

Practice

- Experiment 7.1: Write a C program to perform file I/O operations.
- Experiment 7.2: Implement a program to demonstrate error handling in file operations.
- Experiment 7.3: Develop a program to perform bitwise operations.
- Experiment 7.4: Write a C program to demonstrate the use of preprocessor directives.
- Experiment 7.5: Implement a program to perform operations on bits.
- Experiment 7.6: Develop a program to demonstrate the use of macros.
- Experiment 7.7: Write a C program to implement a simple calculator using functions.
- Experiment 7.8: Implement a program to demonstrate command-line arguments.

Textbooks:

1. "The C Programming Language" by Brian W. Kernighan and Dennis M. Ritchie.
2. "Programming in ANSI C" by E. Balagurusamy.
3. "Programming in ANSI and TURBO C" by Ashon N. Kamthane.

Reference Books:

1. "C: A Reference Manual" by Samuel P. Harbison and Guy L. Steele
2. "Expert C Programming: Deep C Secrets" by Peter van der Linden.

Network and Protocols for IoT (70 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1006	Network and Protocols for IoT	3	1+2+0

Course Description:

This course covers the essential networking concepts and protocols required for the Internet of Things (IoT). Students will learn about IoT communication models, network architectures, and key protocols for device connectivity, data transmission, and network security. The course involves hands-on experiments to design, implement, and troubleshoot IoT networks, including the use of AWS Greengrass for edge computing.

Course Objectives:

- Understand the fundamental networking concepts and protocols for IoT.
- Gain proficiency in designing and implementing IoT network architectures.
- Learn to configure and troubleshoot IoT communication protocols and security mechanisms.

Course Outcomes (COs):

- **CO1:** Explain the fundamental networking concepts and protocols used in IoT. (Understand, Remember)
- **CO2:** Demonstrate the ability to design and implement IoT network architectures. (Apply, Analyze)
- **CO3:** Configure and troubleshoot IoT communication protocols. (Apply, Create)
- **CO4:** Implement security mechanisms for IoT networks. (Apply, Evaluate)
- **CO5:** Analyze the performance and scalability of IoT networks. (Analyze, Evaluate)

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	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 Syllabus:

Module 1: Introduction to IoT Networking (10 hours)

- **Theory:**
  - Overview of IoT and its networking requirements
  - IoT communication models: Device-to-Device, Device-to-Cloud, Device-to-Gateway, Back-End Data-Sharing
  - IoT network architectures and topologies
  - Networking fundamentals: OSI and TCP/IP models

- **Practice**
  - Experiment 1.1: Setting up a basic IoT network with Raspberry Pi and sensors
  - Experiment 1.2: Configuring a simple Device-to-Device communication
  - Experiment 1.3: Implementing a Device-to-Cloud communication using MQTT

Module 2: IoT Communication Protocols (10 hours)

- **Theory:**
  - Overview of IoT communication protocols: MQTT, CoAP, HTTP/HTTPS, WebSockets
  - Comparison of IoT protocols: Use cases and performance
  - Protocol stack for IoT: Physical layer, Data link layer, Network layer, Transport layer, Application layer

- **Practice**
  - Experiment 2.1: Implementing MQTT communication between devices
  - Experiment 2.2: Configuring CoAP for constrained devices
  - Experiment 2.3: Setting up HTTP/HTTPS communication for IoT devices
  - Experiment 2.4: Using WebSockets for real-time data transmission

Module 3: Wireless Technologies for IoT (10 hours)

- **Theory:**
  - Overview of wireless technologies: Wi-Fi, Bluetooth, Zigbee, LoRaWAN, NB-IoT
  - Comparative analysis of wireless technologies for IoT
  - Configuring and managing wireless networks for IoT

- **Practice**
  - Experiment 3.1: Configuring Wi-Fi connectivity for IoT devices
  - Experiment 3.2: Implementing Bluetooth communication for IoT
  - Experiment 3.3: Setting up a Zigbee network for IoT sensors

Module 4: IoT Network Security (10 hours)

- **Theory:**
  - Security challenges in IoT networks
  - Key security protocols and mechanisms: SSL/TLS, DTLS, IPSec
  - Implementing access control and authentication in IoT

- **Practice**
  - Experiment 4.1: Configuring SSL/TLS for secure IoT communication
  - Experiment 4.2: Implementing DTLS for constrained devices
  - Experiment 4.3: Setting up IPSec for IoT network security
  - Experiment 4.4: Implementing device authentication using OAuth

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

- **Theory:**
  - Data management in IoT: Data collection, storage, and processing
  - IoT data analytics and visualization
  - Integration with cloud platforms for data analytics

- **Practice**
  - Experiment 5.1: Collecting and storing IoT data in a cloud database
  - Experiment 5.2: Implementing real-time data analytics using Apache Kafka
  - Experiment 5.3: Visualizing IoT data using Grafana

Module 6: Performance and Scalability in IoT Networks (10 hours)

- **Theory:**
  - Performance metrics for IoT networks: Latency, throughput, packet loss, energy consumption
  - Scalability challenges and solutions in IoT networks
  - Load balancing and quality of service (QoS) in IoT
- **Practice:**
  - Experiment 6.1: Measuring performance metrics in an IoT network
  - Experiment 6.2: Implementing load balancing for IoT devices
  - Experiment 6.3: Configuring QoS for IoT applications

Module 7: Advanced Topics in IoT Networking and Edge Computing (10 hours)

- **Theory:**
  - Edge computing and its role in IoT
  - IoT interoperability and standards
  - Future trends in IoT networking

- **Practice:**
  - Experiment 7.1: Setting up AWS Greengrass for edge computing

- Experiment 7.2: Deploying and managing edge computing applications with AWS Greengrass
- Experiment 7.3: Integrating edge devices with the cloud using AWS Greengrass

Text Books

1. "Internet of Things: A Hands-On Approach" by Arshdeep Bahga and Vijay Madisetti.
2. "Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry" by Maciej Kranz.

Reference Books :

1. "IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things" by David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, and Jerome Henry.
2. Getting Started with the Internet of Things: Connecting Sensors and Microcontrollers to the Cloud (Make: Projects) 1st Edition, Kindle Edition.

Data Analysis and Visualization Using Python (70 Hours)

Code	Course Title	Credit	T-P-PJ
CUTM1018	Data Analysis and Visualization Using Python	4	0-1-3

Course Description: This course emphasizes the use of tools and techniques to collect, analyze, and interpret data.

Course Objectives:

- Understand how to read, store and display each data type.
- Get skill to quickly and easily draw plot or visualize the information through visualization technique.
- The ability to develop visualization to tell the story.

Course Outcomes (COs):

- **CO1**: Able to gain knowledge on visualization with good story line and perform job of a data analyst. (Understand)
- **CO2**: Able to analyse and visualize the dataset. (Analyze)
- **CO3**: Ability to design dashboard. (Create)
- **CO4**: Analyze Text data and gain insights. (Analyze)
- **CO5**: Select appropriate data visualization technique for given data. (Understand)

CO-PO-PSO Mapping:

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

Course Syllabus:

Module 1: STORY BOARD DEVELOPMENT (20 hours)

- The objective and flow of the story to be understood through cases

Module 2: DATA READING USING PYTHON FUNCTIONS (23 hours)

- Python libraries: Pandas, NumPy, Plotly, Matplotlib, Seaborn, Dash
- Data collection from online data sources, Web scrap, and data formats such as HTML, CSV, MS Excel, data compilation, arranging and reading data, data munging

Module 3: DATA VISUALSATION USING PYTHON LIBRARIES (27 hours)

- Different graphs such as Scatterplot, Line chart, Histogram, Bar chart, Bubble chart, Heatmaps etc.
- Dashboard Basics – Layout, Reporting, Infographics, Interactive components, live updating

Projects List:

1. COVID 19
2. World Development Indicators
3. ERP dashboarding
4. Details of Social/ Empowerment schemes of Govt. etc.

References:

- https://www.programmer-books.com/wp-content/uploads/2019/04/Python-for-Data-Analysis-2nd-Edition.pdf
- https://towardsdatascience.com/data-visualization/home

Software Engineering and Testing (70 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1011	Software Engineering and Testing	3	1+2+0

Course Description:

This course provides a comprehensive overview of software engineering principles and practices, with a strong emphasis on software testing methodologies. Students will explore the entire software development lifecycle, from initial requirements gathering to design, implementation, and maintenance, with a focus on ensuring software quality through systematic testing.

Course Objectives:

- To learn the foundational principles and methodologies in software engineering, including software design principles.
- To develop expertise in master software testing techniques and quality assurance practices to ensure high-quality software and efficient defect management.
- To gain knowledge about the processes involved in maintaining and evolving software systems over time.

Course Outcomes:

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

- CO1: Define the phases of the software development life cycle (SDLC) and their importance. (Remembering)
- CO2: Clarify and discuss the principles and practices of software engineering. (Understanding)
- CO3: Implement requirement elicitation techniques and object-oriented design principles. (Applying)
- CO4: Evaluate software quality attributes and formulate testing strategies. (Analyzing)
- CO5: Design and construct comprehensive software systems, covering the process from requirements gathering to testing. *(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**	-	-	-	-	-	-	-	-	-	**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 Syllabus:

Module 1: Software Engineering Fundamentals (10 hours)

Theory

- Overview, SDLC Phases, Software Process Models: Waterfall, Agile, Spiral
- Roles, Responsibilities in Software Development Teams

Practice

- Experiment 1.1: Create a project plan for a software development project.
- Experiment 1.2: Develop a requirement specification document.
- Experiment 1.3: Implement a software process model for a project.
- Experiment 1.4: Create a work breakdown structure (WBS) for a project.
- Experiment 1.5: Develop a Gantt chart for project scheduling.
- Experiment 1.6: Conduct a feasibility study for a software project.

Module 2: Requirements Engineering (10 hours)

Theory

- Requirements Elicitation, Analysis
- Functional, Non-functional Requirements
- Use Case Modeling, User Stories
- Requirement Specification, Documentation

Practice

- Experiment 2.1: Perform requirement elicitation using interviews.

- Experiment 2.2: Develop use case diagrams and scenarios.
- Experiment 2.3: Write user stories for a project.
- Experiment 2.4: Develop functional and non-functional requirements.
- Experiment 2.5: Create a requirements traceability matrix (RTM).
- Experiment 2.6: Conduct a requirement review session

Module 3: Software Design Principles (10 hours)

Theory

- Software Design Basics, Modularization, Abstraction
- Design Patterns: Creational, Structural, Behavioral
- Architectural Styles: Layered, Client-Server, MVC

Practice

- Experiment 3.1: Develop a software design using UML diagrams.
- Experiment 3.2: Implement design patterns in a software project.
- Experiment 3.3: Create a class diagram for a project.
- Experiment 3.4: Develop a sequence diagram for a project.
- Experiment 3.5: Implement a state diagram for a project.
- Experiment 3.6: Design a software architecture using layered architecture

Module 4: Object-Oriented Design (10 hours)

Theory

- Classes, Objects, Inheritance, Encapsulation, Polymorphism
- Design Principles: SOLID, GRASP

Practice

- Experiment 4.1: Implement inheritance and polymorphism in a software project.
- Experiment 4.2: Develop an object-oriented design using SOLID principles.
- Experiment 4.3: Create an object-oriented model using CRC cards.
- Experiment 4.4: Implement a design pattern in a project.
- Experiment 4.5: Develop a software component using encapsulation.
- Experiment 4.6: Create an object-oriented model using UML.

Module 5: Software Testing Fundamentals (10 hours)

Theory

- Testing Levels: Unit, Integration, System
- Testing Techniques: Black Box, White Box, Grey Box
- Test Case Design, Testing Strategies: Regression, Smoke, Acceptance
- Test Planning, Execution, Test Automation Frameworks

Practice

- Experiment 5.1: Write and execute unit test cases.
- Experiment 5.2: Develop and execute integration test cases.
- Experiment 5.3: Implement a regression testing strategy.
- Experiment 5.4: Write and execute system test cases.
- Experiment 5.5: Develop and execute acceptance test cases.
- Experiment 5.6: Implement test automation using Selenium or JUnit.

Module 6: Software Quality Assurance (10 hours)

Theory

- Quality Assurance vs. Quality Control
- Software Quality Attributes: Reliability, Usability, Maintainability, Scalability
- Metrics for Software Quality Measurement, Defect Tracking, Management

Practice

- Experiment 6.1: Develop a software quality assurance plan.
- Experiment 6.2: Implement software quality metrics and measurement.
- Experiment 6.3: Conduct a software quality audit.
- Experiment 6.4: Implement defect tracking and management.
- Experiment 6.5: Conduct a peer review and inspection.
- Experiment 6.6: Develop a test plan and test cases.

Module 7: Software Maintenance and Evolution (10 hours)

Theory

- Software Maintenance Activities: Corrective, Adaptive, Perfective

- Impact Analysis, Change Management, Refactoring Techniques
- Legacy System Migration, Modernization

Practice

- Experiment 7.1: Develop a software maintenance plan.
- Experiment 7.2: Implement corrective and adaptive maintenance.
- Experiment 7.3: Conduct impact analysis for software changes.
- Experiment 7.4: Implement software refactoring techniques.
- Experiment 7.5: Develop a strategy for legacy system migration.
- Experiment 7.6: Implement software modernization techniques

Relational and Distributed Databases (90 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1005	Relational and Distributed Databases	4	2+2+0

Course Description:

This course provides an in-depth understanding of both relational and distributed database systems. Students will learn the fundamental principles, architectures, and technologies underpinning relational and distributed databases. The curriculum includes 30 hours of theory and 30 practical experiments to reinforce the theoretical concepts.

Course Objectives:

1. Understand the fundamental concepts of relational databases.
2. Learn advanced SQL and database normalization.
3. Explore practically the principles and architecture of distributed databases.

Course Outcomes (COs):

2. **CO1:** Explain the fundamental concepts and principles of relational databases. (Understand, Remember)
3. **CO2:** Apply advanced SQL queries and database normalization techniques. (Apply, Create)
4. **CO3:** Describe the principles and architecture of distributed databases. (Understand, Remember)
5. **CO4**: Implement and manage relational and distributed database systems. (Apply, Evaluate)
6. **CO5:** Analyze and optimize database performance. (Analyze, Evaluate)

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	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 Syllabus:

Module 1: Database System Concepts and Data Models (14 hours)

- **Theory:**
  - Overview of Database
  - Concept of Different Data Models
  - DBMS Architecture, and Building a DBMS

- **Practice**
  - Collect data from different data sources and identify the type of data source
  - Installation of MySQL Workbench. Import and export the database into MySQL Workbench
  - Create instances of database in MySQL.
  - Viewing all databases
  - Viewing all Tables in a Database
  - Creating Tables (With and Without Constraints)

Module 2: Introduction to Relational Databases (14 hours)

- **Theory:**
  - Overview of Relational Databases
  - Database Models and ER Diagrams
  - Relational Algebra and Calculus

- **Practice**
  - Experiment 1.1: Creating ER Diagrams
  - Experiment 1.2: Translating ER Diagrams to Relational Schemas
  - Experiment 1.3: Implementing Relational Schemas in MySQL
  - Experiment 1.4: Basic SQL Queries (SELECT, INSERT, UPDATE, DELETE)
  - Experiment 1.5: Advanced SQL Queries (JOIN, UNION, INTERSECT)
  - Experiment 1.6: Using Aggregate Functions in SQL

Module 3: Advanced SQL and Database Normalization (14 hours)

- **Theory**:
  - Advanced SQL Concepts (Subqueries, Indexing, Views)
  - Database Normalization (1NF, 2NF, 3NF, BCNF)
  - Transaction Management and Concurrency Control
- **Practice**
  - Experiment 2.1: Writing Subqueries in SQL
  - Experiment 2.2: Creating and Managing Indexes
  - Experiment 2.3: Implementing Views
  - Experiment 2.4: Normalizing a Database to 3NF
  - Experiment 2.5: Creating Transactions in SQL
  - Experiment 2.6: Implementing Concurrency Control Mechanisms

Module 4: Introduction to Distributed Databases (14 hours)

- **Theory:**

- Overview of Distributed Databases
  - Distributed Database Architectures
  - Data Fragmentation, Replication, and Allocation
- **Practice**
  - Experiment 3.1: Setting Up a Distributed Database Environment
  - Experiment 3.2: Implementing Data Fragmentation
  - Experiment 3.3: Configuring Data Replication
  - Experiment 3.4: Managing Data Allocation
  - Experiment 3.5: Connecting and Querying Distributed Databases
  - Experiment 3.6: Ensuring Data Consistency in a Distributed Environment

Module 5: Distributed Database Transactions (14 hours)

- **Theory:**
  - Distributed Transactions and Concurrency Control
  - Two-Phase Commit Protocol
  - Distributed Query Processing and Optimization

- **Practice**
  - Experiment 4.1: Implementing Distributed Transactions
  - Experiment 4.2: Applying the Two-Phase Commit Protocol
  - Experiment 4.3: Distributed Query Processing
  - Experiment 4.4: Optimizing Distributed Queries
  - Experiment 4.5: Handling Failures in Distributed Transactions
  - Experiment 4.6: Monitoring and Troubleshooting Distributed Transactions

Module 6: Database Performance and Tuning (14 hours)

- **Theory:**
  - Database Performance Metrics
  - Indexing and Query Optimization
  - Database Tuning Techniques

- **Practice**
  - Experiment 5.1: Analyzing Database Performance Metrics
  - Experiment 5.2: Implementing Indexes for Performance Optimization
  - Experiment 5.3: Query Optimization Techniques
  - Experiment 5.4: Performing Database Tuning
  - Experiment 5.5: Monitoring and Troubleshooting Database Performance
  - Experiment 5.6: Conducting a Performance Audit

Module 7: Emerging Trends in Databases (14 hours)

- **Theory:**
  - NoSQL Databases: Concepts and Applications
  - NewSQL Databases: Characteristics and Use Cases
  - Big Data and Databases in Cloud Computing
  - Future Directions in Database Technologies

Text Books

- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- "Distributed Systems: Principles and Paradigms" by Andrew S. Tanenbaum and Maarten Van Steen

Reference Books :

- "SQL Performance Explained" by Markus Winand
- "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" by Pramod

J. Sadalage and Martin Fowler

Customer Experience Design and Programming (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1012	Customer Experience Design and Programming	4	1+2+1

Course Description: This course aims to provide students with the skills and knowledge to design and program customer experiences using fundamental web technologies. It covers the principles of customer experience (CX) design, user experience (UX) design, programming for interactive experiences, and project management. The curriculum includes 15 hours of theory, 30 hands-on experiments, and 15 hours devoted to a capstone project.

Course Objectives:

- Understand the principles of customer experience design.
- Create user-centric designs and interfaces.
- Develop programming skills for interactive customer experiences using JavaScript and related web technologies.

Course Outcomes (COs):

- **CO1:** Explain fundamental principles of customer experience design. (Understand, Remember)
- **CO2:** Apply UX design principles to create user-friendly interfaces. (Apply, Create)
- **CO3:** Develop interactive customer experiences using JavaScript, HTML, CSS, Bootstrap, and jQuery. (Apply, Create)
- **CO4:** Manage and execute CX design projects from conception to completion. (Apply, Evaluate)
- **CO5:** Evaluate the effectiveness of CX designs through user feedback and analytics. (Evaluate, Analyze)

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 Syllabus:

Module 1: Introduction to Customer Experience Design (8 hours)

Theory:

- Overview of Customer Experience (CX) Design
- Importance of CX in Modern Business
- Key Principles and Concepts in CX Design

Practice:

- **Experiment 1.1:** Analyzing Good and Bad CX Design Examples
- **Experiment 1.2:** Mapping Customer Journeys
- **Experiment 1.3:** Identifying Customer Pain Points
- **Experiment 1.4:** Designing Customer Journey Maps with Miro
- **Experiment 1.5:** Creating Personas Using Canva

Module 2: User Experience (UX) Design Fundamentals (8 hours)

Theory:

- Introduction to UX Design
- Principles of User-Centered Design
- Basics of Wireframing and Prototyping

Practice:

- **Experiment 2.1:** Creating Wireframes for a Website Using Figma
- **Experiment 2.2:** Prototyping a Simple App Interface with Pencil Project
- **Experiment 2.3:** Conducting Usability Testing with Maze
- **Experiment 2.4:** Developing Mockups Using Balsamiq
- **Experiment 2.5:** Creating User Flows with Lucidchart

Module 3: Programming for Interactive Experiences (12 hours)

Theory:

- Basics of Interactive Programming
- Introduction to HTML, CSS, and Bootstrap
- Introduction to JavaScript for Web Interactivity

Practice:

- **Experiment 3.1:** Creating a Simple Interactive Web Page Using HTML and CSS
- **Experiment 3.2:** Implementing Responsive Design with Bootstrap
- **Experiment 3.3:** Adding Basic Interactive Elements with JavaScript
- **Experiment 3.4:** Building Interactive Forms Using Bootstrap and JavaScript
- **Experiment 3.5:** Developing a Basic Single Page Layout with Bootstrap and JavaScript

Module 4: Advanced JavaScript Techniques (14 hours)

Theory:

- Advanced JavaScript Concepts (Functions, Objects, ES6+)
- JavaScript DOM Manipulation
- Basics of Asynchronous JavaScript (AJAX, Fetch API)

Practice:

- **Experiment 4.1:** Using JavaScript for Dynamic Content Creation
- **Experiment 4.2:** Manipulating the DOM with Vanilla JavaScript
- **Experiment 4.3:** Creating Dynamic Content with JavaScript and JSON
- **Experiment 4.4:** Building a Simple Interactive Gallery Using JavaScript
- **Experiment 4.5:** Implementing Basic AJAX Requests

Module 5: jQuery for Interactive Web Development (12 hours)

Theory:

- Introduction to jQuery
- jQuery Selectors and Events
- jQuery Effects and Animations

- jQuery and AJAX

Practice:

- **Experiment 5.1:** Manipulating the DOM with jQuery
- **Experiment 5.2:** Creating Animations and Effects Using jQuery
- **Experiment 5.3:** Building a Dynamic Web Page with jQuery
- **Experiment 5.4:** Implementing AJAX with jQuery
- **Experiment 5.5:** Developing a Simple Interactive Web Application Using jQuery

Module 6: Project Management for CX Design (6 hours)

Theory:

- Basics of Project Management
- Introduction to Agile Methodology
- Managing a CX Design Project

Practice:

- **Experiment 6.1:** Creating a Project Plan for a CX Design Project Using Google Sheets
- **Experiment 6.2:** Using Trello for Task Management and Collaboration
- **Experiment 6.3:** Conducting Sprint Planning and Reviews
- **Experiment 6.4:** Tracking Progress and Deadlines with Notion

Module 7: Evaluation and Feedback (6 hours)

Theory:

- Gathering and Analyzing User Feedback
- Using Basic Analytics Tools
- Iterative Design and Continuous Improvement

Practice:

- **Experiment 7.1:** Designing User Surveys with Google Forms
- **Experiment 7.2:** Analyzing User Feedback Data with Google Analytics
- **Experiment 7.3:** Conducting Simple A/B Testing with Google Optimize
- **Experiment 7.4:** Using Hotjar for Heatmaps and User Recordings

Capstone Project:

- **Project Planning and Proposal:**
  - Identifying a CX Design Problem or Opportunity
  - Defining Project Objectives and Scope
  - Creating a Detailed Project Plan
- **Project** Development:
  - Designing and Prototyping the CX Solution
  - Implementing the Solution Using JavaScript, jQuery, and Web Technologies
  - Testing and Iterating Based on User Feedback
- **Project** Presentation:
  - Preparing a Final Report and Presentation
  - Demonstrating the CX Solution
  - Gathering and Analyzing Final Feedback

Textbooks and References:

- *"Eloquent JavaScript: A Modern Introduction to Programming"* by Marijn Haverbeke
- *"JavaScript: The Good Parts"* by Douglas Crockford
- *"Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics"* by Jennifer Niederst Robbins

"jQuery in Action" by Bear Bibeault and Yehuda Katz

Job Readiness

Code	Course Title	Credit	T-P-PJ
CUTM1016	**Job Readiness**	6	0-6-0

Course Objectives

- Develop competent level of English proficiency, i.e., 6.5 band on the CELTS Test
- Enhance proficiency in verbal, quantitative aptitude and logical reasoning skills aiming for Level 4 mastery in MyPerfectice
- Strengthen students' employability skills through communication, analytical and problem-solving abilities

Course Outcomes

After completion of the course students will be able to:

- CO1.Achieve competency in English language (6.5 band on the CELTS Test)
- CO2. Apply English proficiency in real-world scenarios such as professional communication and presentations
- CO3. Exhibit verbal ability, strong quantitative aptitude and advanced logical reasoning (Level 4 mastery in MyPerfectice)
- CO4.Develop and apply enhanced employability skills, including effective communication, sharp analytical thinking and problem- solving abilities in various professional contexts.
- CO5. Commitment to lifelong learning fostering a mindset of continuous improvement in English proficiency and employability skills.

Note: A student will be awarded the credits and grades as outlined in the attached presentation: https://drive.google.com/file/d/1Wst-jdAJuHHVtYC4F-p3SKuw1PHWOI1U/view?usp=sharing

Course Outcome to Program Outcome Mapping:

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

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

COURSE SYLLABUS

Course I: CELTS - Listening, Speaking, Reading and Writing

Module I: CELTS Listening

- Notes/ Form/Table completion
- Label the Map/Passage, Multiple Choice Questions
- Complete the Sentences, listening to Find Information
- Assessment on Listening Skills

Module II: CELTS Speaking

- Speaking about self, family, hobbies, interests
- Introduction & Interview
- Topic Discussion (e.g, Environment, Post Covid 19, Job)
- Assessment on Speaking Skills

Module III: CELTS Reading

- Skimming and Scanning
- Sentence Completion
- Choose the Correct options (A, B, C, D)
- Locating the Specific Information
- Assessment on Reading Skill

Module IV: CELTS Writing

- Summarising the chart, table or graph
- Comparing and contrasting graphs and tables
- Describing maps & diagrams
- Agreeing & disagreeing
- Expressing a personal view & opinion
- Assessment on Writing Skill
- Job Application: CV & Cover Letter (2nd year)
- Letter Writing
- Email Writing (2nd year)
- Getting Started –writing an introduction

Course II: CELTS

Verbal Module I:

Grammar (4 Hrs)

- Articles
- Prepositions
- Subject-Verb
- Spotting Errors
- Sentence Correction

Module II: Vocabulary (5 Hrs)

- Synonyms
- Antonyms
- Contextual Vocabulary

Module III: Reading Comprehension (3 Hrs)

- Paragraph/ Sentence Completion
- Jumbled Sentences/ Jumbled Paragraph
- Reading Comprehension

Module IV: Verbal Analogies (3 Hrs)

References Recommended:

Books

- The Official Cambridge Guide to Ielts Student's Book With Answers with DVD Rom
- Simone Braverman's Target Band 7
- Focus on IELTS Foundation Coursebook.

Semester – II
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

Cloud Practitioner (AWS) (42 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1010	Cloud Practitioner (AWS)	2	1+1+0

Course Description

This course is designed to provide students with a foundational understanding of AWS cloud services and prepare them for the AWS Certified Cloud Practitioner exam. It covers basic cloud concepts, AWS core services, security, architecture, pricing, and support.

Course Objectives:

1. Understand the basic concepts of cloud computing and AWS.
2. Learn about core AWS services and their uses.
3. Gain knowledge on AWS security measures, compliance, and architectural best practices.

Course Outcomes (COs):

1. Explain the basic concepts of cloud computing and AWS. (Understand)
2. Describe core AWS services and their use cases. (Remember)
3. Identify AWS security measures and compliance practices. (Understand)
4. Explain AWS architecture and best practices. (Understand)
5. Describe AWS pricing models and support plans. (Remember)

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 Syllabus:

Module 1: Introduction to Cloud Computing and AWS (8 hours)

Theory:

- Overview of cloud computing.
- AWS global infrastructure.
- Key AWS services and their benefits.

Practice

- Experiment 1.1: Set up an AWS Free Tier account.
- Experiment 1.2: Navigate the AWS Management Console.
- Experiment 1.3: Explore AWS Global Infrastructure using the AWS Console.
- Experiment 1.4: Create and manage IAM users and groups.
- Experiment 1.5: Implement multi-factor authentication (MFA) for IAM users.
- Experiment 1.6: Use the AWS Pricing Calculator to estimate costs.

Module 2: Core AWS Services (9 hours)

Theory:

- Compute services (EC2, Lambda).
- Storage services (S3, EBS).
- Database services (RDS, DynamoDB).
- Networking services (VPC, Route 53).

Practice:

- . Experiment 2.1: Launch and configure an EC2 instance
- Experiment 2.2: Set up and configure AWS Lambda functions.
- Experiment 2.3: Create and manage S3 buckets.
- Experiment 2.4: Implement versioning and lifecycle policies in S3.
- Experiment 2.5: Create and configure an RDS instance.
- Experiment 2.6: Set up and use DynamoDB tables.
- Experiment 2.7: Create and configure a VPC.
- Experiment 2.8: Use Route 53 to configure DNS settings.

Module 3: AWS Security and Compliance (7 hours)

Theory:

- AWS shared responsibility model.
- AWS Identity and Access Management (IAM).

- AWS security services (KMS, CloudTrail).

Practice:

- Experiment 3.1: Configure IAM roles and policies.
- Experiment 3.2: Enable and use AWS CloudTrail for auditing.
- Experiment 3.3: Set up AWS KMS for key management.
- Experiment 3.4: Implement security groups and network ACLs.
- Experiment 3.5: Create and configure AWS Config for compliance monitoring.
- Experiment 3.6: Set up AWS Inspector for security assessments.

Module 4: AWS Architecture and Best Practices (8 hours)

Theory:

- AWS Well-Architected Framework.
- Architectural best practices for high availability and fault tolerance.
- Cost optimization strategies.

Practice:

- Experiment 4.1: Use the Well-Architected Tool to review an architecture.
- Experiment 4.2: Design a highly available and fault-tolerant architecture.
- Experiment 4.3: Implement cost optimization techniques using AWS Trusted Advisor.
- Experiment 4.4: Monitor and optimize resource usage with CloudWatch.

Module 5: AWS Pricing and Support (8 hours)

Theory:

- AWS pricing models (pay-as-you-go, reserved instances, spot instances).
- Cost management tools (AWS Budgets, Cost Explorer).
- AWS support plans and their features.

Practice:

- Experiment 5.1: Use AWS Budgets to set up budget alerts.
- Experiment 5.2: Explore cost reports with AWS Cost Explorer.
- Experiment 5.3: Create a report using the AWS Pricing Calculator.
- Experiment 5.4: Simulate a support case using AWS Support Center.
- Experiment 5.5: Analyze and optimize costs using the AWS Free Tier dashboard.
- Experiment 5.6: Review AWS Trusted Advisor recommendations.

Module 6: Networking and Storage(8 hours)

Theory:

- Networking Basics
- Amazon Virtual Private Cloud,
- VPC Security,
- Amazon Route53
- AWS Elastic Block Store(EBS)
- AWS Simple Storage Service(S3)

Practice:

- Experiment 6.1: Create your own VPC.
- Experiment 6.2:Implement VPC peering.
- Experiment 6.3:Create private and public subnet in a VPC.
- Experiment 6.4:Create a bucket and upload Object to it.
- Experiment 6.5:Host a static website using S3.
- Experiment 6.6:Create an EBS volume and attach it to an EC2 Instance.

Module 7: Compute and databases Service (8 hours)

Theory:

- Overview of Compute Services,
- Amazon Elastic Compute Cloud,
- Amazon EC2 versus Managed Services,
- Amazon EC2 Cost Optimization,
- Container Services,
- AWS Lambda

Practice:

- Experiment 7.1: Create a Windows and Linux Virtual Machine.
- Experiment 7.2: Install mySQL using AWS EC2 Instance.
- Experiment 7.3: Create a RDS Instance using EC2 Instance.
- Experiment 7.4: Create a database table using AWS DynamoDB.

Textbooks and References:

- "AWS Certified Cloud Practitioner Study Guide" by Ben Piper.
- "AWS Certified Cloud Practitioner Exam Guide" by AWS.
- AWS online documentation and whitepapers.
- AWS training and certification resources.

Java Programming (112 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1004	Java Programming	6	2+2+2

Course Description:

This course provides an in-depth introduction to Java programming, focusing on both fundamental concepts and advanced features of the language. Java is a versatile, object-oriented language widely used in industry, making it an excellent choice for learning programming and software development principles.

Course Objectives:

- Introduce the Java programming language, 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:

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

- **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	-	-	-	1	-	-	-	1	-	2	2	2
CO2	3	3	3	2	2	-	-	-	-	-	-	-	1	1	1
CO3	2	-	2	-	-	-	-	-	1	-	-	1	2	2	2
CO4	2	3	3	2	-	-	1	-	-	1	-	-	2	3	2
CO5	3	3	3	2	2	1	-	1	-	1	-	-	2	2	2

***High-3, Medium-2, Low-1**
Course Syllabus:

Module 1: Introduction to Java (15 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.
- Experiment 1.6: Develop a program to perform string operations in Java.
- Experiment 1.7: Write a program to demonstrate the use of arrays in Java.
- Experiment 1.8: Implement a program to perform matrix operations in Java.

Module 2: Object-Oriented Programming in Java (15 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.
- Experiment 2.6: Develop a program to demonstrate the use of abstract classes in Java.
- Experiment 2.7: Write a program to demonstrate the use of interfaces in Java.
- Experiment 2.8: Implement a program to demonstrate the use of inner classes in Java.

Module 3: Java Collections Framework (15 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.
- Experiment 3.6: Develop a program to demonstrate the use of TreeMap in Java.
- Experiment 3.7: Write a program to demonstrate the use of Iterator in Java.
- Experiment 3.8: Implement a program to demonstrate the use of Comparator and Comparable in Java.

Module 4: Exception Handling and I/O (20 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.
- Experiment 4.6: Develop a program to read and write data to a file in Java.
- Experiment 4.7: Write a program to demonstrate the use of BufferedReader and BufferedWriter in Java.
- Experiment 4.8: Implement a program to demonstrate the use of FileInputStream and FileOutputStream in Java.

Module 5: Multithreading and Concurrency (15 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 interfacein 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.
- Experiment 5.6: Develop a program to demonstrate thread pooling in Java.
- Experiment 5.7: Write a program to demonstrate the use of Executors framework in Java.
- Experiment 5.8: Implement a program to demonstrate the use of Callable and Future in Java

Module 6: GUI Programming and Advanced Java Basics (15 hours)

Theory

- AWT: Container
- Components
- Layout Managers
- Event Handling
- Introduction to Swing
- Generics, Lambda Expressions, Stream API Annotations, Reflection
- Java Database Connectivity (JDBC)

Practice

- Experiment 6.1: Write a program to demonstrate the use of generics in Java.
- Experiment 6.2: Implement a program to demonstrate the use of lambda expressions in Java.
- Experiment 6.3: Develop a program to demonstrate the use of Stream API in Java.
- Experiment 6.4: Write a program to demonstrate the use of annotations in Java.
- Experiment 6.5: Implement a program to demonstrate the use of reflection in Java.
- Experiment 6.6: Develop a program to connect to a database using JDBC in Java.
- Experiment 6.7: Write a program to perform CRUD operations using JDBC in Java.
- Experiment 6.8: Implement a program to demonstrate the use of Prepared Statement in Java.

Module 7: Networking and Advanced(15 hours)

Theory

- Networking Fundamental
- Client-Server Communication
- Remote Method Invocation (RMI)

Practice

- Experiment 7.1:Write a program for one way communication.
- Experiment 7.2:Write a program for two way communication.
- Experiment 7.3:Write a program to implement RMI.
- Experiment 7.4:Write a program to implement TCP/IP protocal.

Projects (22 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.

Data Structures with Competitive Coding (140 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1002	Data Structures with Competitive Coding	6	2+4+0

Course Description:

This course combines the study of fundamental data structures with practical competitive coding techniques. Students will explore essential data ststructures such as arrays, linked lists, stacks, queues, trees, and graphs, and learn how to implement and utilize these structures effectively. The course also delves into advanced topics including hash tables, heaps, and balanced trees.

Course Objectives:

- To train students in algorithm analysis, recursion, and selecting suitable data structures for problem-solving, focusing on algorithm correctness.
- To implement dynamic data structures (linked lists, binary trees) and sub-quadratic sorting algorithms (quick sort, merge sort, heap sort) to solve data structure problems.
- To able to get jobs in different IT firms as a developer with core and competitive coding skills.

Course Outcomes:

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

- **CO1:** Recall and relate algorithmic techniques and recursive methods to efficiently solve complex problems, demonstrating proficiency in analyzing and validating algorithm correctness.
- **CO2:** Describe and discuss dynamic data structures, optimizing data manipulation and storage solutions.
- **CO3:** Apply and demonstrate sorting and searching algorithms, achieving sub-quadratic performance in data processing tasks.
- **CO4:** Analyse and test coding problems using advanced data structures and algorithms under timed conditions.
- **CO5:** Formulate strategies and show readiness for software development roles by enhancing coding skills and practical data structure knowledge.

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	-	-	-	-	-	1	-	-	1	2	-	1

CO2	2	3	2	2	-	2	-	-	-	-	-	-	2	2	1
CO3	2	3	3	2	-	2	-	-	-	1	-	-	3	3	-
CO4	3	3	3	2	-	2	1	-	-	-	1	-	3	3	-
CO5	3	3	3	2	2	-	-	1	-	-	-	-	2	-	1

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

Course Syllabus:

Module 1: Data Structures Basics (10 hours)

Theory

- Definition, Importance, Algorithm, and Pseudocode
- Types of Data Structures, Abstract Data Types (ADTs)
- Complexity Analysis & Asymptotic Notations

Module 2: Arrays and Linked Lists (20 hours)

Theory

- Array Operations
- Linked Lists: Singly, Doubly, Circular
- Operations on Linked Lists: Insert, Delete, Search, Sort, Reverse, Merge

Practice

- Experiment 2.1: Write a program to demonstrate the use of arrays.
- Experiment 2.2: Implement a program to perform insertion and deletion operations on an array.
- Experiment 2.3: Develop a program to implement a singly linked list.
- Experiment 2.4: Write a program to perform insertion and deletion operations on a singly linked list.
- Experiment 2.5: Implement a program to implement a doubly linked list.
- Experiment 2.6: Develop a program to perform insertion and deletion operations on a doubly linked list.
- Experiment 2.7: Write a program to implement a circular linked list.
- Experiment 2.8: Implement a program to perform insertion and deletion operations on a circular linked list.
- Experiment 2.9: Develop a program to reverse a linked list.
- Experiment 2.10: Write a program to merge two sorted linked lists.

Module 3: Stacks and Queues (20 hours)

Theory

- Stack Operations and Applications
- Queue Operations and Applications
- Variants: Circular Queue, Priority Queue, Deque

Practice

- Experiment 3.1: Write a program to implement a stack using an array.
- Experiment 3.2: Implement a program to perform push and pop operations on a stack.
- Experiment 3.3: Develop a program to implement a queue using an array.
- Experiment 3.4: Write a program to perform enqueue and dequeue operations on a queue.
- Experiment 3.5: Implement a program to implement a circular queue.
- Experiment 3.6: Develop a program to perform insertion and deletion operations on a circular queue.
- Experiment 3.7: Write a program to implement a priority queue.
- Experiment 3.8: Implement a program to perform insertion and deletion operations on a priority queue.

Module 4: Trees (30 hours)

Theory

- Binary Trees: Traversals, Insertion, Deletion
- Binary Search Trees (BST)
- Balanced Trees: AVL, Red-Black Trees

Practice

- Experiment 4.1: Write a program to implement a binary tree.
- Experiment 4.2: Implement a program to perform inorder, preorder, and postorder traversal on a binary tree.
- Experiment 4.3: Develop a program to implement a binary search tree (BST).
- Experiment 4.4: Write a program to perform insertion and deletion operations on a BST.
- Experiment 4.5: Implement a program to find the height of a binary tree.
- Experiment 4.6: Develop a program to implement an AVL tree.
- Experiment 4.7: Write a program to perform insertion and deletion operations on an AVL tree.
- Experiment 4.8: Implement a program to implement a red-black tree.

Module 5: Graphs (20 hours)

Theory

- Representation of Graphs
- Graph Traversal Algorithms: BFS, DFS
- Shortest Path Algorithms: Dijkstra, Floyd-Warshall

Practice

- Experiment 5.1: Write a program to represent a graph using adjacency matrix.
- Experiment 5.2: Implement a program to represent a graph using adjacency list.
- Experiment 5.3: Develop a program to perform BFS traversal on a graph.
- Experiment 5.4: Write a program to perform DFS traversal on a graph.
- Experiment 5.5: Implement a program to find the shortest path in a graph using Dijkstra's algorithm.
- Experiment 5.6: Develop a program to find the shortest path in a graph using Floyd-Warshall algorithm.
- Experiment 5.7: Write a program to find the minimum spanning tree in a graph using Kruskal's algorithm.
- Experiment 5.8: Implement a program to find the minimum spanning tree in a graph using Prim's algorithm.

Module 6: Hashing and Heaps (20 hours)

Theory

- Hash Tables: Hash Functions, Collision Handling
- Heaps: Min-Heap, Max-Heap, Operations

Practice

- Experiment 6.1: Write a program to implement a hash table.
- Experiment 6.2: Implement a program to handle collisions using chaining.
- Experiment 6.3: Develop a program to handle collisions using open addressing.
- Experiment 6.4: Write a program to implement a min-heap.
- Experiment 6.5: Implement a program to perform insertion and deletion operations on a min- heap
- Experiment 6.6: Develop a program to implement a max-heap.
- Experiment 6.7: Write a program to perform insertion and deletion operations on a max-heap.
- Experiment 6.8: Implement a program to heap sort an array.

Module 7: Sorting and Searching (20 hours)

Theory

- Linear Search, Binary Search
- Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Heap Sort, Radix Sort.

Practice

- Experiment 7.1: Write a program to implement linear search.
- Experiment 7.2: Implement a program to implement binary search.
- Experiment 7.3: Develop a program to implement bubble sort.
- Experiment 7.4: Write a program to implement selection sort.
- Experiment 7.5: Implement a program to implement insertion sort.
- Experiment 7.6: Develop a program to implement quick sort.
- Experiment 7.7: Write a program to implement merge sort.

- Experiment 7.8: Implement a program to implement heap sort.

Textbooks:

1. "Data Structures" by S. Lipschutz and G.A. Pai.
2. "Data Structures Using C" by Reema Thareja.

Reference Books:

1. " Data Structures Using C" by Amiya Kumar Rath, and Alok Kumar Jagadev.
2. "Expert Data Structure with C" by R.B. Patel.

"Competitive Programming" by Steven Halim, Felix Halim, and Suhendry Effendy.

Information Security (CISCO) (56 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1007	Information Security (CISCO)	3	1+1+1

Course Description:

This course covers the foundational principles of information security and the practices involved in protecting data and systems. The curriculum is designed to provide students with the knowledge and skills necessary to implement and manage security measures using Cisco technologies. Students will gain practical experience through hands-on labs and exercises.

Course Objectives:

- Understand the fundamentals of information security.
- Learn to configure and manage Cisco security devices and technologies.
- Develop skills to implement security policies and measures to protect data and systems.

Course Outcomes (COs):

- **CO1:** Explain the basic principles of information security and their importance. (Understand, Remember)
- **CO2:** Configure and manage Cisco security devices and technologies. (Apply, Analyze)
- **CO3:** Implement security policies and access controls. (Apply, Create)
- **CO4:** Conduct risk assessments and vulnerability analysis. (Evaluate, Analyze)
- **CO5:** Develop and implement strategies for incident response and disaster recovery. (Apply, Evaluate)

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	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 Syllabus:

Module 1: Introduction to Information Security (8 hours)

- **Theory:**
  - Overview of Information Security
  - Key Concepts and Terminology
  - Security Goals: Confidentiality, Integrity, and Availability
  - Threats, Vulnerabilities, and Attacks

- **Practice:**
  - Experiment 1.1: Exploring Cisco Security Products and Solutions
  - Experiment 1.2: Setting Up a Basic Security Lab Environment
  - Experiment 1.3: Configuring Basic Security Settings on Cisco Routers and Switches
  - Experiment 1.4: Implementing Basic Firewall Rules

Module 2: Network Security Fundamentals (8 hours)

- **Theory:**
  - Network Security Principles and Concepts
  - Introduction to Firewalls, VPNs, and IDS/IPS
  - Secure Network Design and Architecture

- **Practice**
  - Experiment 2.1: Configuring Cisco ASA Firewall
  - Experiment 2.2: Implementing VPNs with Cisco Routers and ASA
  - Experiment 2.3: Setting Up and Managing Cisco IDS/IPS
  - Experiment 2.4: Network Segmentation and VLAN Configuration

Module 3: Security Policies and Access Control (8 hours)

- **Theory:**
  - Importance of Security Policies
  - Access Control Models: DAC, MAC, and RBAC
  - Implementing Access Control Policies
  - User Authentication and Authorization

- **Practice**
  - Experiment 3.1: Configuring AAA (Authentication, Authorization, Accounting) on Cisco Devices
  - Experiment 3.2: Implementing Role-Based Access Control (RBAC)
  - Experiment 3.3: Setting Up Network Access Control (NAC)
  - Experiment 3.4: Managing User Accounts and Permissions

Module 4: Risk Management and Vulnerability Assessment (8 hours)

- **Theory:**
  - Introduction to Risk Management
  - Risk Assessment and Analysis
  - Vulnerability Assessment and Penetration Testing
  - Security Audits and Compliance

- **Practice**
  - Experiment 4.1: Conducting a Risk Assessment for a Network
  - Experiment 4.2: Using Cisco Security Tools for Vulnerability Scanning
  - Experiment 4.3: Performing Penetration Testing with Cisco Secure Network Analytics
  - Experiment 4.4: Implementing Security Measures Based on Assessment Results

Module 5: Incident Response and Disaster Recovery (8 hours)

- **Theory:**
  - Incident Response Planning and Procedures
  - Disaster Recovery Strategies and Planning
  - Business Continuity Planning
  - Legal and Ethical Issues in Information Security

- **Practice**
  - Experiment 5.1: Developing an Incident Response Plan
  - Experiment 5.2: Setting Up a Disaster Recovery Environment
  - Experiment 5.3: Simulating Security Incidents and Response
  - Experiment 5.4: Testing and Evaluating Business Continuity Plans

- **Project (16 hours)**

Text Books

- "Cisco ASA, PIX, and FWSM Firewall Handbook" by David Hucaby
- "CCNA Cyber Ops SECOPS – Certification Guide" by Omar Santos and Joseph Muniz

Reference Books

- "Cisco Next-Generation Security Solutions: All-in-one Cisco ASA FirePOWER Services, NGIPS and AMP" by Omar Santos
- Fundamentals Of Information Security Fundamentals of Information Security: A Complete Go-to Guide for Beginners to Understand All the Aspects of Information Security (English Edition

Semester – III & IV

Design and Analysis of Algorithms (140 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1003	Design and Analysis of Algorithms	6	2+4+0

Course Description:

This course provides a comprehensive exploration of algorithm design and analysis techniques. Students will study various strategies for developing efficient algorithms and analyzing their performance. Emphasis is placed on understanding the theoretical foundations of algorithms as well as practical techniques for solving complex computational problems.

Course Objectives:

- To understand algorithms' characteristics and design techniques, Implement and analyze divide-and-conquer, greedy, and dynamic programming algorithms.
- To comprehend graph algorithms and optimization techniques and explore NP-completeness and approximation algorithms.
- To prepare for competitive coding and placement exams.

Course Outcomes:

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

- **CO1**: Recall the definition and characteristics of algorithms. *(Remembering)*
- **CO2**: Explain various algorithmic strategies. *(Understanding)*
- **CO3**: Implement divide-and-conquer and greedy algorithms. *(Applying)*
- **CO4**: Analyze the efficiency of different algorithms. *(Analyzing)*
- **CO5**: Develop advanced algorithms for solving complex problems. *(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	3	2	-	-	-	-	1	-	-	-	1	2	-	1
CO2	2	3	3	2	-	2	1	-	-	1	-	-	2	2	1
CO3	2	3	3	2	-	2	-	-	-	1	-	1	3	3	-
CO4	2	3	3	2	-	2	-	-	1	-	1	-	3	3	-

CO5	**2**	**2**	**2**	**-**	**-**	**1**	**-**	**1**	**1**	**-**	**-**	**1**	**2**	**-**	**2**

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

Course Syllabus:

Module 1: Algorithm

Basics (20 hours) Theory

- Definition, Characteristics, Algorithm Design Techniques
- Recurrence Relations: Substitution, Recursion Tree, Master Method

Practice

- Experiment 1.1: Write a program to solve a problem using brute force approach.
- Experiment 1.2: Implement a program to solve a problem using divide and conquer approach.
- Experiment 1.3: Develop a program to solve a problem using dynamic programming.
- Experiment 1.4: Write a program to solve a problem using greedy approach.
- Experiment 1.5: Implement a program to solve a problem using backtracking.
- Experiment 1.6: Develop a program to solve a problem using branch and bound.
- Experiment 1.7: Write a program to solve a problem using graph traversal.
- Experiment 1.8: Implement a program to solve a problem using tree traversal.
- Experiment 1.9: Develop a program to solve a problem using matrix operations.
- Experiment 1.10: Write a program to solve a problem using number theory.

Module 2: Divide and Conquer

(10 hours) Theory

- Merge Sort, Quick Sort, Binary Search

Practice

- Experiment 2.1: Write a program to implement merge sort.
- Experiment 2.2: Implement a program to implement quick sort.
- Experiment 2.3: Develop a program to find the maximum and minimum elements in an array using divide and conquer.
- Experiment 2.4: Write a program to find the closest pair of points using divide and conquer.
- Experiment 2.5: Implement a program to find the median of two sorted arrays using divide and conquer.
- Experiment 2.6: Develop a program to perform binary search using divide and conquer.
- Experiment 2.7: Write a program to solve the matrix multiplication problem using divide and conquer.
- Experiment 2.8: Implement a program to solve the convex hull problem using divide and conquer.

Module 3: Greedy Algorithms

(20 hours) Theory

- Activity Selection Problem, Huffman Coding
- Kruskal's and Prim's Algorithms

Practice

- Experiment 3.1: Write a program to solve the activity selection problem using greedy algorithm.
- Experiment 3.2: Implement a program to solve the fractional knapsack problem using greedy algorithm.
- Experiment 3.3: Develop a program to solve the job sequencing problem using greedy algorithm.
- Experiment 3.4: Write a program to solve the coin change problem using greedy algorithm.
- Experiment 3.5: Implement a program to solve the Huffman coding problem using greedy algorithm.
- Experiment 3.6: Develop a program to solve the minimum spanning tree problem using Kruskal's algorithm.
- Experiment 3.7: Write a program to solve the minimum spanning tree problem using Prim's algorithm.
- Experiment 3.8: Implement a program to solve the single source shortest path problem using Dijkstra's algorithm.

Module 4: Dynamic

Programming (30 hours)

Theory

- Principles of Dynamic Programming
- Knapsack Problem, Longest Common Subsequence (LCS), Matrix Chain Multiplication

Practice

- Experiment 4.1: Write a program to solve the 0/1 knapsack problem using dynamic programming.
- Experiment 4.2: Implement a program to solve the longest common subsequence problem using dynamic programming.
- Experiment 4.3: Develop a program to solve the matrix chain multiplication problem using dynamic programming.
- Experiment 4.4: Write a program to solve the rod cutting problem using dynamic programming.
- Experiment 4.5: Implement a program to solve the coin change problem using dynamic programming.
- Experiment 4.6: Develop a program to solve the subset sum problem using dynamic programming.

- Experiment 4.7: Write a program to solve the traveling salesman problem using dynamic programming.
- Experiment 4.8: Implement a program to solve the edit distance problem using dynamic programming.

Module 5: Graph Algorithms

(30 hours) Theory

- Shortest Path Algorithms: Dijkstra, Bellman-Ford
- Minimum Spanning Trees: Kruskal, Prim
- Network Flow: Ford-Fulkerson

Practice

- Experiment 5.1: Write a program to solve the 0/1 knapsack problem using dynamic programming.
- Experiment 5.2: Implement a program to solve the longest common subsequence problem using dynamic programming.
- Experiment 5.3: Develop a program to solve the matrix chain multiplication problem using dynamic programming.
- Experiment 5.4: Write a program to solve the rod cutting problem using dynamic programming.
- Experiment 5.5: Implement a program to solve the coin change problem using dynamic programming.
- Experiment 5.6: Develop a program to solve the subset sum problem using dynamic programming.
- Experiment 5.7: Write a program to solve the traveling salesman problem using dynamic programming.
- Experiment 5.8: Implement a program to solve the edit distance problem using dynamic programming.

Module 6: NP-Completeness and Approximation

Algorithms (20 hours) Theory

- Concepts of NP-Completeness
- Approximation Algorithms

Practice

- Experiment 6.1: Write a program to solve the vertex cover problem using approximation algorithm.
- Experiment 6.2: Implement a program to solve the set cover problem using approximation algorithm.
- Experiment 6.3: Develop a program to solve the traveling salesman problem using approximation algorithm.
- Experiment 6.4: Write a program to solve the knapsack problem using approximation algorithm.

- Experiment 6.5: Implement a program to solve the bin packing problem using approximation algorithm.
- Experiment 6.6: Develop a program to solve the maximum satisfiability problem using approximation algorithm.
- Experiment 6.7: Write a program to solve the minimum dominating set problem using approximation algorithm.
- Experiment 6.8: Implement a program to solve the minimum coloring problem using approximation algorithm.

Module 7: Optimization Techniques (10 hours)

- Amortized Analysis, Cache-Oblivious Algorithms, Parallel Algorithms.

Textbooks:

1. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest,

   and Clifford Stein.
2. "Introduction to the Design and Analysis of Algorithms" by Anany Levitin.
3. "Algorithms Design and Analysis" by Udit Agarwal.

Reference Books:

1. "Algorithm Design" by Jon Kleinberg and Éva Tardos.
2. "Algorithms" by Robert Sedgewick and Kevin Wayne.

"Competitive Programming" by Steven Halim, Felix Halim, and Suhendry Effendy

Cloud Fundamentals (Azure) (42 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1015	Cloud Fundamentals (Azure)	2	1+1+0

Course Description:

This course provides fundamental knowledge and practical skills in Microsoft Azure cloud technology. Students will learn about Azure services, cloud computing concepts, security, architecture, and pricing models. The course will involve hands-on experiments to design, deploy, and manage applications on Azure.

Course Objectives:

- Understand the core concepts of cloud computing and Azure services.
- Gain proficiency in deploying and managing applications on Azure.
- Learn to design scalable, resilient, and cost-effective cloud architectures.

Course Outcomes (COs):

- CO1: Explain the fundamental concepts and benefits of cloud computing and Azure. (Understand, Remember)
- CO2: Demonstrate the ability to use Azure services for deploying and managing applications. (Apply, Analyze)
- CO3: Design and implement scalable and resilient cloud architectures using Azure best practices. (Apply, Create)
- CO4: Implement security and compliance measures on Azure. (Apply, Evaluate)
- CO5: Optimize Azure resource usage and understand Azure pricing models. (Analyze, Evaluate)

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 Syllabus:

Module 1: Introduction to Cloud Computing and Azure (6 hours)

Theory:

- Overview of Cloud Computing
- Introduction to Azure
- Azure Global Infrastructure
- Azure Management Portal and CLI
- Azure Pricing Models and Billing

Practice

- Experiment 1.1: Create and configure an Azure account.
- Experiment 1.2: Navigate the Azure Management Portal.
- Experiment 1.3: Explore and use the Azure CLI.
- Experiment 1.4: Explore Azure pricing models using the Pricing Calculator.

Module 2: Azure Core Services (6 hours)

Theory:

- Azure Virtual Machines (VMs)
- Azure Blob Storage
- Azure SQL Database
- Azure Virtual Network (VNet)
- Azure Functions: Serverless Computing

Practice

- Experiment 2.1: Launch and configure an Azure VM.
- Experiment 2.2: Create and manage Blob Storage.
- Experiment 2.3: Set up a SQL Database in Azure.
- Experiment 2.4: Configure a VNet with subnets and security groups.

Module 3: Networking and Content Delivery (6 hours)

Theory:

- Azure DNS: Domain Name System
- Azure Content Delivery Network (CDN)
- Azure Load Balancer
- Azure ExpressRoute

Practice

- Experiment 3.1: Configure a custom domain using Azure DNS.

- Experiment 3.2: Set up a CDN in Azure.
- Experiment 3.3: Implement Azure Load Balancer for a web application.
- Experiment 3.4: Explore Azure ExpressRoute options.

Module 4: Security, Identity, and Compliance (6 hours)

Theory:

- Azure Active Directory (AD)
- Azure Key Vault
- Azure DDoS Protection and Firewall
- Azure Compliance Programs

Practice

- Experiment 4.1: Create and manage Azure AD users and groups.
- Experiment 4.2: Implement Azure AD roles and policies.
- Experiment 4.3: Enable multi-factor authentication (MFA) in Azure AD.
- Experiment 4.4: Configure Azure Firewall for a web application.

Module 5: Azure Architecture and Design (6 hours)

Theory:

- Azure Well-Architected Framework
- High Availability and Fault Tolerance
- Azure Scale Sets and Load Balancer
- Cost Management and Optimization

Practice

- Experiment 5.1: Design a high-availability architecture using Azure Scale Sets.
- Experiment 5.2: Implement Load Balancer and Scale Sets for an application.
- Experiment 5.3: Explore cost management tools like Azure Cost Management.
- Experiment 5.4: Optimize resource usage with cost explorer.

Module 6: Monitoring and Management (6 hours)

Theory:

- Azure Monitor
- Azure Security Center
- Azure Log Analytics
- Azure Advisor

Practice

- Experiment 6.1: Set up monitoring for an application using Azure Monitor.
- Experiment 6.2: Configure Azure Security Center for auditing.
- Experiment 6.3: Use Azure Log Analytics to monitor resource configurations.
- Experiment 6.4: Utilize Azure Advisor for best practice recommendations.

Module 7: Deployment and Automation (6 hours)

Theory:

- Azure Resource Manager (ARM)
- Azure App Service
- Azure DevOps
- Azure Automation

Practice

- Experiment 7.1: Automate infrastructure deployment using ARM templates.
- Experiment 7.2: Deploy a web application with Azure App Service.
- Experiment 7.3: Set up a CI/CD pipeline using Azure DevOps.
- Experiment 7.4: Implement automated deployment with Azure Automation.
- Experiment 7.5: Deploy a serverless application using Azure Functions and Logic Apps.
- Experiment 7.6: Manage and monitor deployment versions in Azure DevOps.

Textbooks and References:

- "Exam Ref AZ-900 Microsoft Azure Fundamentals" by Jim Cheshire
- "Azure for Architects" by Ritesh Modi
- Microsoft Azure Documentation
- Online Tutorials and Courses on Azure

Android Development with Kotlin(126 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1013	Android Development with Kotlin	6	1+3+2

Course Description:

This course offers a comprehensive introduction to Android development using Kotlin, the modern programming language officially supported for Android development. Students will learn to design, build, and deploy Android applications with a focus on Kotlin's features and best practices.

Course Objectives:

- Understand the basics of Android development, Kotlin programming, 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 and Kotlin programming.(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

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

Course Syllabus:

Module 1: Introduction to Android Development (16 hours) Theory

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

Practice

- Experiment 1.1: Set up Android Studio and create a new project.
- Experiment 1.2: Write a Kotlin program to print "Hello, World!".
- Experiment 1.3: Demonstrate the use of Kotlin data types and variables.
- Experiment 1.4: Implement basic arithmetic operations in Kotlin.
- Experiment 1.5: Implement a conditional expressions & loop in Kotlin.
- Experiment 1.6: Implement function concept in Kotlin
- Experiment 1.7: Implement collection and arrays in Kotlin
- Experiment 1.8: Implement object Oriented concept in Kotlin

Module 2: Android Activities and Layouts (16 hours)

Theory

- Activity & its type
- Layouts: Linear, Relative, Constraint, Frame,Table
- UI Components: TextView, EditText, Button, ImageView, RecyclerView
- Material Design Principles, Activity Life Cycle

Practice

- Experiment 2.1: Create a simple Android app with a single activity
- Experiment 2.2: Design a user interface using Layout and user input controls (TextView, EditText, Button)
- Experiment 2.3: Design a user interface using Constraint Layout
- Experiment 2.4: Design a user interface using Relative Layout .
- Experiment 2.5: Implement a simple calculator app in Android(Use themes & Styles)
- Experiment 2.6: Implement material design components in Android.
- Experiment 2.7: Implement an activity lifecycle in Android.

Module 3: Navigation & data Passing (16 hours)

Theory

- Intents, Intent Filter
- Services
- Broadcast Receivers

Practice

- Experiment 3.1: Use intents to navigate between activities.
- Experiment 3.2: Receive data from the user by Edit Text and pass the data to another activity using intent.

- Experiment 3.3Create and use fragments in an Android app.
- Experiment 3.4: Implement a service to run in the background.
- Experiment 3.5: Use broadcast receivers to receive system events.
- Experiment 3.6: Retrieve the device's battery info. And show in a project

Module 4: Data Storage (16 hours)

Theory

- Shared Preferences, Internal, External Storage
- SQLite Database, Room Persistence Library
- Content Providers

Practice

- Experiment 4.1: Use shared preferences for simple data storage.
- Experiment 4.2: Implement internal storage for file management.
- Experiment 4.3: Implement external storage for file management.
- Experiment 4.4: Implement SQLite database in an Android app(CRUD Operation).
- Experiment 4.5: Implement data binding in an Android app.
- Experiment 4.6: Implement content providers for data sharing.
- Experiment 4.7: Use Room Persistence Library for data storage.

Module 5: List, Adapters, and Permission (14 Hrs)

Theory

- Android Permissions
- List, Adapter, Spinner,
- Alert Dialog

Practice

- Experiment 5.1: Implement arraylist & adapter to populate data to listview.
- Experiment 5.2: Usage of different types of Android permissions in App.
- Experiment 5.3: Use of Spinner in Android App
- Experiment 5.4: Create alert Dialog in android app.
- Experiment 5.5: Retrieve data from a given URL and arrange them in a recycler view

Module 6: Network Communication (16 hours)

Theory

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

Practice

- Experiment 6.1: Implement HTTP networking using Retrofit.
- Experiment 6.2: Implement HTTP networking using Volley.
- Experiment 6.3: Parse JSON data from a web API.

- Experiment 6.4: Use OkHttp for network communication.
- Experiment 6.5: Use Firebase Realtime Database for data synchronization.

Module 7: Advanced Topics (18 hours)

Theory

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

Practice

- Experiment 7.1: Implement Firebase Authentication in an Android app.
- Experiment 7.2: Use Google Maps API for location-based services.
- Experiment 7.3: Implement push notifications using Firebase Cloud Messaging.
- Experiment 7.4: Integrate social media login (Facebook, Google) in an Android app.
- Experiment 7.5: Implement data synchronization with Firebase.
- Experiment 7.6: Use ML Kit for image recognition.
- Experiment 7.7: Implement in-app purchases in an Android app.
- Experiment 7.8: Use Google Play Services for analytics.

Project (28 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:

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

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

Reference Books:

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

Prompt Engineering using ChatGPT (42 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1014	Prompt Engineering using ChatGPT	2	1+1+0

Course Description:

This course offers a deep dive into the art and science of prompt engineering, focusing on how to effectively interact with advanced AI models like ChatGPT to generate high-quality outputs. Students will learn the techniques for crafting precise and effective prompts to maximize the performance of language models, ensuring that responses are relevant, accurate, and aligned with user needs.

Course Objectives:

- Understand the basics of prompt engineering, the capabilities of ChatGPT, and the ethical and societal implications of AI.
- Master the techniques of crafting effective prompts and implementing advanced features of ChatGPT in real-world scenarios.
- Design and develop applications using ChatGPT.

Course Outcomes:

- CO1: Recall the fundamental concepts of prompt engineering and ChatGPT. (Remembering)
- CO2: Explain the techniques of crafting effective prompts for different use cases. (Understanding)
- CO3: Apply prompt engineering techniques to develop applications using ChatGPT. (Applying)
- CO4: Analyze the performance and limitations of ChatGPT in various scenarios. (Analyzing)
- CO5: Develop innovative applications using advanced features of ChatGPT. (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	2	1	-	-	-	-	-	-	-	-	-	2	2	1
CO2	2	1	2	1	3	-	-	-	-	-	-	-	3	2	2
CO3	-	1	2	1	2	-	-	-	-	-	-	-	2	2	3
CO4	-	3	-	-	1	-	-	-	-	-	-	-	3	3	2
CO5	1	1	2	1	-	-	-	-	-	-	-	-	2	3	3

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

Course Syllabus:

Module 1: Introduction to Prompt Engineering (7 hours)

Theory

- Basics, Importance of Prompt Engineering
- Overview of ChatGPT and its Capabilities
- Setting up Development Environment

Practice

- Experiment 1.1: Set up development environment for ChatGPT.
- Experiment 1.2: Create basic prompts for text generation.
- Experiment 1.3: Develop prompts for Q&A applications.
- Experiment 1.4: Create prompts for summarization tasks.
- Experiment 1.5: Implement prompts for translation tasks.
- Experiment 1.6: Optimize prompts for different applications.

Module 2: Crafting Effective Prompts (7 hours)

Theory

- Techniques for Effective Prompt Creation
- Prompts for Different Applications: Text Generation, Q&A, Summarization, Translation
- Prompt Tuning and Optimization

Practice

- Experiment 2.1: Design prompts for chatbots.
- Experiment 2.2: Create prompts for virtual assistants.
- Experiment 2.3: Develop prompts for content generation.
- Experiment 2.4: Create prompts for creative writing tasks.
- Experiment 2.5: Implement prompts for educational tools.
- Experiment 2.6: Optimize prompts for personalized learning.

Module 3: Developing Applications with ChatGPT (7 hours)

Theory

- Chatbots, Virtual Assistants
- Content Generation, Creative Writing
- Educational Tools, Personalized Learning

Practice

- Experiment 3.1: Implement a chatbot using ChatGPT.
- Experiment 3.2: Develop a virtual assistant using ChatGPT.
- Experiment 3.3: Create a content generation tool using ChatGPT.
- Experiment 3.4: Implement a personalized learning tool using ChatGPT.
- Experiment 3.5: Integrate ChatGPT with external APIs.
- Experiment 3.6: Develop a creative writing assistant using ChatGPT.

Module 4: Advanced Features of ChatGPT (7 hours)

Theory

- Customizing Responses, Controlling Output
- Handling Ambiguity, Ensuring Consistency
- Integrating External APIs, Data Sources

Practice

- Experiment 4.1: Customize ChatGPT responses.
- Experiment 4.2: Control ChatGPT output format.
- Experiment 4.3: Handle ambiguity in ChatGPT responses.
- Experiment 4.4: Ensure consistency in ChatGPT output.
- Experiment 4.5: Integrate ChatGPT with external data sources.
- Experiment 4.6: Implement advanced features for specific use cases.

Module 5: Performance Analysis and Optimization (7 hours)

Theory

- Evaluating ChatGPT Performance
- Fine-tuning, Model Selection
- Optimizing for Speed, Accuracy

Practice

- Experiment 5.1: Evaluate ChatGPT performance for different tasks.
- Experiment 5.2: Fine-tune ChatGPT for specific applications.
- Experiment 5.3: Optimize ChatGPT for speed and accuracy.
- Experiment 5.4: Implement error handling for ChatGPT responses.
- Experiment 5.5: Analyze ChatGPT limitations and workarounds.
- Experiment 5.6: Implement best practices for prompt engineering.

Module 6: Ethical and Societal Implications (7 hours)

- Ethical Considerations in AI
- Bias, Fairness, Transparency
- Societal Impact, Responsible AI Development

Textbooks and References:

- "The Art of Prompt Engineering with ChatGPT" by Nathan Hunter
- "Prompt Engineering for AI: Writing Effective Prompts for NLP Models" by Ethan Williams
- "Deep Learning for Natural Language Processing" by Palash Goyal, Sumit Pandey, and Karan Jain
- "Generative AI with Python and TensorFlow 2" by Joseph Babcock

System Administrator (RedHat) (56 hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1009	System Administrator (RedHat)	3	2+1+0

Course Description:

This course is designed to prepare students for the Red Hat Certified System Administrator (RHCSA) certification. It covers essential skills for managing and administering Red Hat Enterprise Linux (RHEL) systems, including system configuration, network services, and security management.

Course Objectives:

- Understand the fundamentals of Red Hat Enterprise Linux system administration.
- Learn to manage and configure RHEL systems.
- Gain practical experience in administering network services and security.

Course Outcomes (COs):

1. Explain the basic concepts of Red Hat Enterprise Linux and its administration. (Understand)
2. Configure and manage RHEL systems effectively. (Apply)
3. Implement and manage network services and security measures in RHEL. (Apply)
4. Troubleshoot and maintain RHEL systems. (Analyze)
5. Demonstrate proficiency in tasks required for the RHCSA certification. (Create)

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	-	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 Syllabus:

Module 1: Introduction to Red Hat Enterprise Linux (8 hours)

Theory:

- Overview of RHEL and Linux distributions.

- Installation and initial setup of RHEL.
- Basic command-line usage and file system navigation

Practice

- Experiment 1.1: Install Red Hat Enterprise Linux.
- Experiment 1.2: Navigate the file system using basic commands.
- Experiment 1.3: Use the man pages to find command usage.
- Experiment 1.4: Create, move, copy, and delete files and directories.
- Experiment 1.5: Use the vi text editor to edit configuration files.

Module 2: User and Group Management (8 hours)

Theory:

- Managing user accounts and groups.
- Understanding file permissions and ownership.
- Configuring sudo privileges.

Practice

- Experiment 2.1: Create and manage user accounts.
- Experiment 2.2: Create and manage groups.
- Experiment 2.3: Change file permissions and ownership.
- Experiment 2.4: Configure and test sudo access.
- Experiment 2.5: Use ACLs to provide additional permissions.

Module 3: File System Management (8 hours)

Theory:

- Understanding and managing disk partitions.
- Mounting and unmounting file systems.
- Managing Logical Volume Management (LVM).

Practice

- Experiment 3.1: Create and manage disk partitions using fdisk.
- Experiment 3.2: Format and mount file systems.
- Experiment 3.3: Create and manage LVM volumes.
- Experiment 3.4: Extend and reduce LVM volumes.
- Experiment 3.5: Configure and use swap space.

Module 4: Network Configuration (8 hours)

Theory:

- Configuring network interfaces.
- Managing network services and firewall.
- Understanding and using SSH for remote management.

Practice

- Experiment 4.1: Configure static and dynamic IP addresses.
- Experiment 4.2: Configure and manage network services (e.g., httpd, sshd).
- Experiment 4.3: Set up and manage firewall rules using firewalld.
- Experiment 4.4: Use SSH for remote access and management.
- Experiment 4.5: Configure and manage network time synchronization.

Module 5: Service Management and System Monitoring (8 hours)

Theory:

- Managing system services using systemctl.
- Monitoring system performance and logs.
- Scheduling tasks with cron and at.

Practice

- Experiment 5.1: Start, stop, and manage services with systemctl.
- Experiment 5.2: Monitor system performance using tools like top, htop, and vmstat.
- Experiment 5.3: View and manage system logs using journalctl.
- Experiment 5.4: Schedule recurring tasks with cron.
- Experiment 5.5: Schedule one-time tasks with at.

Module 6: Security and SELinux (8 hours)

Theory:

- Understanding SELinux and its configuration.
- Managing firewall rules and security policies.
- Implementing basic system security measures

Practice

- Experiment 6.1: Configure and manage SELinux policies.
- Experiment 6.2: Manage and troubleshoot firewall rules.
- Experiment 6.3: user and group security policies.
- Experiment 6.4: Secure services using SELinux.
- Experiment 6.5: and use iptables for additional security.

Module 7: Troubleshooting and Maintenance (8 hours)

Theory:

- Basic system troubleshooting techniques.
- Understanding and managing boot processes.
- Regular system maintenance tasks

Practice

- Experiment 7.1: Diagnose and resolve common boot issues.
- Experiment 7.2: Troubleshoot network connectivity problems.
- Experiment 7.3: Use rescue mode for system recovery.
- Experiment 7.4: Perform regular system updates and patches.
- Experiment 7.5: Backup and restore system data.

Textbooks and References:

- "RHCSA/RHCE Red Hat Linux Certification Study Guide" by Michael Jang and Alessandro Orsaria
- "Red Hat Enterprise Linux 8 Essentials" by Neil Smyth
- Red Hat official documentation and resources

DotNET Programming

Course Title	Course Code	Credit	T-P-PJ
DotNET Programming	CUTM2474	4	0-2-2

Course Objectives

- Introduce the C# programming language, its features, and implement object-oriented programming concepts.
- Utilize the Collections Framework, manage exceptions, and perform file I/O operations.Develop multithreaded applications and connect to databases.
- Learn to develop web applications

Course Outcomes

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

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

Course Syllabus:

Module-1: The .NET Technology(15 hours)

Introduction to .NET Framework, Architecture of .NET framework– BCL (Base Class Library), CLR(Common Language Runtime), . NET Languages – introduction, Types of applications supported by .NET Technology, Managed code, compilation to intermediate language, Just-In-Time compilation,garbage collection, assemblies and the GAC

Module-2: C# Basics(15 hours)

Data Types, Variables & Constants, Operators in C#, Arithmetic Operators, Prefix and Postfix notation, Assignment Operators, Relational Operators, Other Operators, Operators precedence, Flow Control and Conditional Statements if-else statement, switch statement, Loops in C#, for loop, do-while loop, Array in C#, foreach Loop.

Module-3: Object and Classes(15 hours)

Concept of a class, Objects, Fields, Methods, Instantiating the class, Accessing the members of a class, Access modifiers, Properties, Static members of the class, Constructors, Destructors, Overloading Constructors, Value types (out & ref keywords) ,sealed keyword, Inheritance,

Module-4: Object and Classes -II(15 hours)

Polymorphism, Overriding the methods, the new keywords, Type casting, is and as keywords, Boxingand Un-boxing, Structures, Enumeration, Nested Classes, Abstract classes, Exception handling, Delegates & Events

Module-5: ASP.NET - I(15 hours)

Overview of ASP.NET framework, ASP.NET Page – layout, lifecycle, Stages in Web Forms Processing, Introduction to Server Controls, HTML Controls, Validation Controls, User control, Data Binding Controls, Configuration, Session State, Adding controls to a web form, Buttons, Text Box , Labels, Checkbox, Radio Buttons, List Box, etc, Master Pages, themes and skins

Module-6: ASP.NET – 2(15 hours)

Overview of ADO.NET, Benefits of ADO.NET, ADD.NET compared to classic ADO, ADO.NET

architecture (Connected and Disconnected), State Management, Using Database connection.

Working with DataSets, Managed Providers, Data Binding, Typed DataSets, Working with Data

Reader, Transactions

Module-7: Packaging and Deploying(15 hours)

Reporting, Web Services – overview, creation and calling, Packaging and Deploying ASP.NET Applications

Reference Books

1. Andrew Troelsen, Philip Japikse, : C# 6.0 and the .NET 4.6 Framework, Apress, 2017.
2. Black Book: .NET 4.5 Programming (6-in-1) covers .NET 4.5 Framework, Visual Studio 2012, C#2012, ASP.NET 4.5, VB 2012, and F# 3.0, Dreamtech Press, 2013.
3. Matthew MacDonald: Beginning ASP.NET 4.5 in C#, Apress, 2013.
4. G. Andrew Duthie, “ASP.NET programming with Microsoft Visual C#.NET Step by Step”, version 2003, Prentice-Hall of India.

Course Code	Course Title	Credit	Type (T+P+Pj)
FDCU1000	**Full-Stack Development with MERN**	**18**	**1+8+9**
CUFD1001	MongoDB for Developers	3	1+2+0
CUFD1002	Node.js and Express.js Development	3	0+2+1
CUFD1003	Front-End Development with React	3	0+2+1
CUFD1004	Full Stack Integration and Deployment	3	0+2+1
CUFD1005	Product Development	6	0+0+6
GACU1010	**Generative AI**	**12**	**0+8+4**
CUGA1011	Fundamentals of Generative AI	1	0+1+0
CUGA1012	Advanced Techniques in Generative AI	2	0+2+0
CUGA1013	Real-Time Generative AI	3	0+1+2
CUGA1014	Research and Development in Generative AI	2	0+2+0
CUGA1015	Capstone Project in Generative AI	4	0+2+2
MLCU1020	**Data Analytics and Machine Learning**	**18 + 4 (Optional)**	**0+6+12**
CUML1021	Machine Learning for Predictive Analytics	4	0+2+2

CUML1022	Deep Learning for Image Analytics	4	0+2+2
CUML1023	Data Analytics using Tableau	4	0+2+2
CUML1024	ML for Spectral Imaging (Optional)	4	0+2+2
CUML1025	Project	6	0+0+6
CTCU1030	**Cloud Technology**	**12**	**0+6+6**
CUCT1031	Advanced Cloud Architecture and Design	3	0+2+1
CUCT1032	Cloud Development and DevOps	3	0+2+1
CUCT1033	Cloud Security and Compliance	3	0+2+1
CUCT1034	Capstone Project	3	0+0+3
DSCU1040	**Drone Imaging and Spectral Analysis**	**12**	**0+6+6**
CUDS1041	Drone Image Processing using Pix4D	3	0+2+1
CUDS1042	Multispectral Image Analytics for Agriculture	3	0+2+1
CUDS1043	Drone Imaging Applications	2	0+2+0
CUDS1044	Domain Project	4	0+0+4
STCU1050	**Software Technology**	**18**	**0+6+12**
CUST1051	Advanced Java	4	0+2+2

CUST1052	Angular	4	0+2+2
CUST1053	Spring Boot	4	0+2+2
CUST1054	Product Development	6	0+0+6
MACU1070	**Mobile App Development**	**12**	**1+6+5**
CUMA1071	Introduction to Mobile App Development	3	1+2+0
CUMA1072	React Native Development	3	0+2+1
CUMA1073	Flutter Development	3	0+2+1
CUMA1074	Advanced Mobile App Development Project	3	0+0+3
GICU1080	**Gaming and Immersive Learning-AR/VR**	**20**	**5+5+10**
CUGI1081	Introduction to Gaming & Simulation	2	1+1+0
CUGI1082	Game Assets and Objects	3	1+1+1
CUGI1083	Building Game Environment	3	1+1+1
CUGI1084	Game Animation, Scripting & UI	3	1+1+1
CUGI1085	Binary Deployment and Cross-Platform Controls	3	1+1+1
CUGI1086	Project	6	0+0+6
BDCU1090	**Blockchain Development**	**18**	**0+7+11**

CUBD1091	INTRODUCTION TO BLOCKCHAIN	2	0+2+0
CUBD1092	CRYPTOCURRENCIES AND SMART CONTRACTS	3	0+2+1
CUBD1093	BLOCKCHAIN DEVELOPMENT	3	0+1+2
CUBD1094	WEB3 AND DECENTRALIZED TECHNOLOGIES	3	0+1+2
CUBD1095	ADVANCED BLOCKCHAIN CONCEPTS AND DEVELOPMENT	3	0+1+2
CUBD1096	CAPSTONE PROJECT IN BLOCKCHAIN DEVELOPMENT	4	0+0+4
CSCU1100	**Cyber Security**	**20**	**8+8+4**
CUCS1101	Linux Server Management and Security	4	2+2+0
CUCS1102	Offensive Security	4	2+2+0
CUCS1103	Defensive Security	4	2+2+0
CUCS1104	Security Analytics	4	2+2+0
CUCS1105	Project	4	0+0+4

MINI AND MICRO DOMAINS PROPOSAL

Domain Track: Full-Stack Development with MERN (18 Credits) (1+8+9)

MongoDB for Developers (70 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUFD1001	**MongoDB for Developers**	03	1+2+0

Course Description: This course covers the fundamental and advanced aspects of MongoDB, focusing on how to utilize this NoSQL database for full-stack development within the MERN stack.

Course Objectives:

- To understand the basics of MongoDB and NoSQL databases.
- To perform CRUD operations, complex queries, and manage databases using MongoDB Atlas.
- To implement data modelling and schema design.

Course Outcomes:

- **CO1**: Understand the fundamentals of NoSQL databases, MongoDB architecture, and perform CRUD operations using MongoDB tools.
- **CO2**: Apply advanced MongoDB features, including indexing, aggregation, geospatial queries, and text search for optimized data management.
- **CO3**: Design effective data models and schemas, utilizing best practices in schema design, embedding vs. referencing, and data modeling.
- **CO4**: Implement replication and sharding techniques to ensure data distribution, redundancy, and scalability in MongoDB applications.
- **CO5**: Utilize MongoDB Atlas for cloud deployment, ensuring security, backup, and restore strategies for robust and secure data management.

CO-PO Mapping

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

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

Course Syllabus:

Module 1: Introduction to MongoDB (10 hours)

- Overview of NoSQL databases.
- MongoDB architecture and concepts.
- Setting up MongoDB locally and on the cloud.

Experiments:

1. **Experiment 1.1**: Set up MongoDB locally on your machine.
2. **Experiment 1.2**: Set up MongoDB on a cloud service like MongoDB Atlas.
3. **Experiment 1.3**: Compare the performance of MongoDB with a traditional SQL database.
4. **Experiment 1.4**: Create a sample database and collection in MongoDB.
5. **Experiment 1.5**: Insert multiple documents into a collection.
6. **Experiment 1.6**: Explore the MongoDB architecture and identify key components.
7. **Experiment 1.7**: Connect to MongoDB using a client (e.g., MongoDB Compass).
8. **Experiment 1.8**: Perform basic queries to retrieve documents from a collection.
9. **Experiment 1.9**: Experiment with different data types supported by MongoDB.
10. **Experiment 1.10**: Set up and configure MongoDB authentication and authorization.

Module 2: CRUD Operations (10 hours)

- Create, Read, Update, Delete (CRUD) operations.
- MongoDB Shell and Compass.
- Data validation and schema design.

Experiments:

1. **Experiment 2.1**: Create documents in a MongoDB collection using MongoDB Shell.
2. **Experiment 2.2**: Read documents from a MongoDB collection using various query operators.
3. **Experiment 2.3**: Update documents in a collection using different update operators.
4. **Experiment 2.4**: Delete documents from a collection using MongoDB Shell.
5. **Experiment 2.5**: Use MongoDB Compass to perform CRUD operations.
6. **Experiment 2.6**: Implement data validation rules in a MongoDB collection.
7. **Experiment 2.7**: Design a schema for a sample application and implement it in MongoDB.
8. **Experiment 2.8**: Experiment with different data validation techniques in MongoDB.
9. **Experiment 2.9**: Create an index on a collection and observe its effect on query performance.
10. **Experiment 2.10**: Use MongoDB transactions to perform multiple operations atomically.

Module 3: Advanced MongoDB Features (10 hours)

- Indexing and performance optimization.
- Aggregation framework.
- Geospatial queries and text search.

Experiments:

1. **Experiment 3.1**: Create and use indexes to optimize query performance.
2. **Experiment 3.2**: Experiment with different types of indexes (e.g., single field, compound).
3. **Experiment 3.3**: Use the aggregation framework to perform data analysis.
4. **Experiment 3.4**: Implement and query geospatial data in MongoDB.
5. **Experiment 3.5**: Perform text search queries on a collection.
6. **Experiment 3.6**: Compare the performance of indexed vs. non-indexed queries.
7. **Experiment 3.7**: Create a complex aggregation pipeline to process data.
8. **Experiment 3.8**: Use map-reduce operations for large-scale data processing.
9. **Experiment 3.9**: Experiment with TTL indexes to automatically delete documents.
10. **Experiment 3.10**: Monitor and analyze query performance using MongoDB tools.

Module 4: Data Modeling and Schema Design (10 hours)

- Schema design patterns.
- Embedding vs. referencing.
- Data modeling best practices.

Experiments:

1. **Experiment 4.1**: Design a schema for a blog application using embedding.
2. **Experiment 4.2**: Design a schema for an e-commerce application using referencing.
3. **Experiment 4.3**: Compare the performance of embedded vs. referenced data models.
4. **Experiment 4.4**: Implement one-to-many and many-to-many relationships in MongoDB.
5. **Experiment 4.5**: Design a schema for a social media application.
6. **Experiment 4.6**: Implement data modeling best practices in a sample application.
7. **Experiment 4.7**: Optimize a data model for read-heavy workloads.
8. **Experiment 4.8**: Optimize a data model for write-heavy workloads.
9. **Experiment 4.9**: Experiment with different schema design patterns (e.g., bucket pattern).
10. **Experiment 4.10**: Use MongoDB Atlas to visualize and analyze data models.

Module 5: Replication (10 hours)

- Understanding replication.
- Setting up replica sets.

Experiments:

1. **Experiment 5.1**: Set up a replica set locally.
2. **Experiment 5.2**: Add members to a replica set and configure their roles.
3. **Experiment 5.3**: Perform read and write operations on a replica set.
4. **Experiment 5.4**: Simulate a primary node failure and observe the failover process.
5. **Experiment 5.5**: Monitor replica set status and performance using MongoDB tools.
6. **Experiment 5.6**: Configure read preferences to read from secondary nodes.
7. **Experiment 5.7**: Implement write concern and read concern settings in a replica set.
8. **Experiment 5.8**: Perform a backup and restore operation on a replica set.
9. **Experiment 5.9**: Experiment with delayed replica set members.
10. **Experiment 5.10**: Implement and test a hidden replica set member.

Module 6: Sharding (10 hours)

- Introduction to sharding and partitioning data.

Experiments:

1. **Experiment 6.1**: Set up a sharded cluster locally.
2. **Experiment 6.2**: Add shards to a sharded cluster and configure shard keys.
3. **Experiment 6.3**: Perform data distribution across shards using different shard keys.
4. **Experiment 6.4**: Monitor and analyze sharded cluster performance.
5. **Experiment 6.5**: Perform read and write operations on a sharded cluster.
6. **Experiment 6.6**: Experiment with different shard key selection strategies.
7. **Experiment 6.7**: Implement zone sharding to control data distribution.
8. **Experiment 6.8**: Migrate chunks between shards manually.
9. **Experiment 6.9**: Handle a shard failure and observe the recovery process.
10. **Experiment 6.10**: Experiment with balancing data across shards.

Module 7: MongoDB Atlas and Cloud Deployment (10 hours)

- Using MongoDB Atlas.
- Backup and restore strategies.
- Security best practices.

Experiments:

1. **Experiment 7.1**: Set up a MongoDB Atlas cluster.
2. **Experiment 7.2**: Configure backup and restore strategies in MongoDB Atlas.
3. **Experiment 7.3**: Implement security best practices in MongoDB Atlas (e.g., IP whitelisting, encryption).

4. **Experiment 7.4**: Monitor MongoDB Atlas performance and usage metrics.
5. **Experiment 7.5**: Scale a MongoDB Atlas cluster vertically and horizontally.
6. **Experiment 7.6**: Perform a data migration to MongoDB Atlas from a local instance.
7. **Experiment 7.7**: Implement automated backups and test data recovery.
8. **Experiment 7.8**: Configure and use MongoDB Atlas triggers.
9. **Experiment 7.9**: Integrate MongoDB Atlas with a serverless platform (e.g., AWS Lambda).
10. **Experiment 7.10**: Implement role-based access control in MongoDB Atlas.

Assignments and Projects:

- Implement a database for a sample application.
- Perform complex queries and aggregations.

Node.js and Express.js Development (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUFD1002	**Node.js and Express.js Development**	**03**	**0+2+1**

Course Description: This course delves into backend development using Node.js and Express.js, equipping students with the skills to create robust and scalable server-side applications.

Course Objectives:

- Understand the Node.js runtime and asynchronous programming.
- Develop RESTful APIs using Express.js.
- Implement middleware and error handling to build secure and scalable backend applications.

Course Outcomes (COs):

- **CO1**: Explain the architecture of Node.js and set up Node.js projects. (Understand, Apply)
- **CO2**: Utilize asynchronous programming concepts such as callbacks, promises, and async/await in Node.js. (Apply, Analyze)
- **CO3**: Develop RESTful APIs using Express.js, including route setup and middleware implementation. (Create, Evaluate)
- **CO4**: Implement authentication and authorization mechanisms using JWT and session-based strategies. (Apply, Analyze)

- **CO5**: Build and deploy secure and scalable Node.js applications, incorporating testing and real-time features. (Create, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	3	2	2	3	-	-	-	-	2	-	2	3	2	3
CO2	3	2	3	2	3	-	-	-	-	2	-	2	3	2	3
CO3	3	2	3	2	3	-	-	-	2	2	1	2	3	3	3
CO4	3	2	3	2	3	-	-	-	2	2	2	2	3	2	3
CO5	3	2	3	2	3	-	-	-	2	2	2	2	3		

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

Course Syllabus:

Module 1: Introduction to Node.js (12 hours)

- Node.js architecture.
- Setting up a Node.js project.
- Node.js modules and npm.

Experiments:

1. **Experiment 1.1**: Install and configure Node.js and npm.
2. **Experiment 1.2**: Create a simple Node.js application.
3. **Experiment 1.3**: Explore Node.js modules and their usage.
4. **Experiment 1.4**: Use npm to manage project dependencies.
5. **Experiment 1.5**: Build a basic server using Node.js.
6. **Experiment 1.6**: Implement environment variables in Node.js.
7. **Experiment 1.7**: Use Nodemon to automate server restarts.
8. **Experiment 1.8**: Create custom Node.js modules.
9. **Experiment 1.9**: Handle errors in a Node.js application.
10. **Experiment 1.10**: Explore the Node.js file system module.

Module 2: Asynchronous Programming (12 hours)

- Callbacks, Promises, and Async/Await.
- Event-driven programming.
- Streams and buffers.

Experiments:

1. **Experiment 2.1**: Implement callbacks in Node.js.
2. **Experiment 2.2**: Convert callback-based code to Promises.
3. **Experiment 2.3**: Use async/await for asynchronous operations.
4. **Experiment 2.4**: Handle multiple asynchronous operations with Promises.
5. **Experiment 2.5**: Create and manage events using the EventEmitter.
6. **Experiment 2.6**: Implement streams for reading and writing files.
7. **Experiment 2.7**: Use streams to process large data sets.
8. **Experiment 2.8**: Create a custom readable and writable stream.
9. **Experiment 2.9**: Handle backpressure in streams.
10. **Experiment 2.10**: Explore buffers and their usage in Node.js.

Module 3: Introduction to Express.js (12 hours)

- Basics of Express.js.
- Setting up routes and middleware.
- Handling requests and responses.

Experiments:

1. **Experiment 3.1**: Install and configure Express.js.
2. **Experiment 3.2**: Create a basic Express.js application.
3. **Experiment 3.3**: Set up routes in an Express.js application.
4. **Experiment 3.4**: Implement middleware functions in Express.js.
5. **Experiment 3.5**: Handle GET and POST requests in Express.js.
6. **Experiment 3.6**: Create a RESTful API with Express.js.
7. **Experiment 3.7**: Implement error handling in Express.js.
8. **Experiment 3.8**: Use query parameters and URL parameters.
9. **Experiment 3.9**: Serve static files with Express.js.
10. **Experiment 3.10**: Implement request logging using middleware.

Module 4: Building RESTful APIs (12 hours)

- REST principles and API design.
- CRUD operations with Express.js.
- Data validation and error handling.

Experiments:

1. **Experiment 4.1**: Design and implement a RESTful API.
2. **Experiment 4.2**: Create endpoints for CRUD operations.
3. **Experiment 4.3**: Validate incoming data using middleware.
4. **Experiment 4.4**: Handle errors in API responses.
5. **Experiment 4.5**: Implement pagination and filtering in APIs.
6. **Experiment 4.6**: Use Postman to test API endpoints.
7. **Experiment 4.7**: Implement versioning in APIs.

8. **Experiment 4.8**: Add documentation to APIs using Swagger.
9. **Experiment 4.9**: Secure APIs with rate limiting.
10. **Experiment 4.10**: Implement CORS in Express.js.

Module 5: Authentication and Authorization (12 hours)

- JWT and session-based authentication.
- Role-based access control.
- Security best practices.

Experiments:

1. **Experiment 5.1**: Implement JWT authentication in an Express.js app.
2. **Experiment 5.2**: Create protected routes using JWT.
3. **Experiment 5.3**: Implement session-based authentication.
4. **Experiment 5.4**: Set up role-based access control.
5. **Experiment 5.5**: Encrypt passwords using bcrypt.
6. **Experiment 5.6**: Secure API endpoints with middleware.
7. **Experiment 5.7**: Implement refresh tokens for JWT.
8. **Experiment 5.8**: Prevent common security vulnerabilities (e.g., XSS, CSRF).
9. **Experiment 5.9**: Monitor authentication attempts and lockout.
10. **Experiment 5.10**: Log security events and audit logs.

Module 6: Advanced Topics in Express.js (12 hours)

- Building real-time applications with WebSockets.
- Testing with Mocha and Chai.
- Deployment and scaling Node.js applications.

Experiments:

1. **Experiment 6.1**: Implement WebSockets for real-time communication.
2. **Experiment 6.2**: Build a chat application using WebSockets.
3. **Experiment 6.3**: Write unit tests for Express.js routes using Mocha.
4. **Experiment 6.4**: Write integration tests using Chai and Supertest.
5. **Experiment 6.5**: Set up continuous integration with GitHub Actions.
6. **Experiment 6.6**: Deploy a Node.js application to Heroku.
7. **Experiment 6.7**: Deploy a Node.js application to AWS.
8. **Experiment 6.8**: Use Docker to containerize a Node.js application.
9. **Experiment 6.9**: Scale a Node.js application using Docker Swarm.
10. **Experiment 6.10**: Monitor and optimize performance of a Node.js app.

Module 7: Capstone Project Development (12 hours)

- Develop a RESTful API for a web application.
- Implement user authentication and authorization.
- Peer reviews and feedback sessions.

Course Code	Course Title	Credits	Type (T+P+Pj)
CUFD1003	**Front-End Development with React**	**03**	**0+2+1**

- Final project presentations.

Experiments:

1. **Experiment 7.1**: Plan and design a full-stack application.
2. **Experiment 7.2**: Develop the backend API using Node.js and Express.js.
3. **Experiment 7.3**: Implement authentication and authorization.
4. **Experiment 7.4**: Integrate the backend with a frontend application.
5. **Experiment 7.5**: Perform unit and integration testing.
6. **Experiment 7.6**: Deploy the application to a cloud service.
7. **Experiment 7.7**: Conduct a peer review of the project.
8. **Experiment 7.8**: Incorporate feedback and optimize the application.
9. **Experiment 7.9**: Prepare a presentation of the project.
10. **Experiment 7.10**: Demonstrate the final application.

Front-End Development with React (84 Hours)

Course Description: This course provides an in-depth exploration of React, focusing on building dynamic, responsive, and efficient front-end applications.

Course Objectives:

1. Connect front-end and back-end components.
2. Implement real-time communication and GraphQL.
3. Deploy applications using modern CI/CD pipelines.

Course Outcomes (COs):

1. **CO1**: Explain the basics of React and JSX, and set up React projects using Create React App. (Understand, Apply)
2. **CO2**: Utilize state and props to manage data and handle events in React components. (Apply, Analyze)
3. **CO3**: Implement React Router for navigation and dynamic routing in React applications. (Apply, Create)

4. **CO4**: Apply state management techniques using Context API and Redux in React applications. (Analyze, Evaluate)
5. **CO5**: Optimize React applications for performance and integrate them with back-end services. (Create, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	2	2	3	-	-	-	-	2	-	2	3	2	3
CO2	3	2	3	2	3	-	-	-	-	2	-	2	3	2	3
CO3	3	2	3	2	3	-	-	-	2	2	1	2	3	3	3
CO4	3	2	3	2	3	-	-	-	2	2	2	2	3	2	3
CO5	3	2	3	2	3	-	-	-	2	2	2	2			

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

Syllabus:

Module 1: Introduction to React (10 hours)

- Basics of React and JSX.
- Setting up a React project with Create React App.
- Functional vs. class components.

Experiments:

1. **Experiment 1.1**: Set up a React project using Create React App.
2. **Experiment 1.2**: Create and render a simple React component.
3. **Experiment 1.3**: Compare functional and class components.
4. **Experiment 1.4**: Use JSX to create React elements.
5. **Experiment 1.5**: Create nested components in React.
6. **Experiment 1.6**: Implement conditional rendering in React.
7. **Experiment 1.7**: Pass data between components using props.
8. **Experiment 1.8**: Use props to customize component behavior.
9. **Experiment 1.9**: Manage component state using useState.
10. **Experiment 1.10**: Handle events in React components.

Module 2: State and Props (12 hours)

- Understanding state and props.
- Handling events in React.
- Lifting state up and state management patterns.

Experiments:

1. **Experiment 2.1**: Create a stateful component using useState.
2. **Experiment 2.2**: Pass state as props to child components.
3. **Experiment 2.3**: Handle form inputs using state.
4. **Experiment 2.4**: Lift state up to a parent component.
5. **Experiment 2.5**: Implement a controlled component.
6. **Experiment 2.6**: Use callback functions to handle events.
7. **Experiment 2.7**: Implement state management patterns.
8. **Experiment 2.8**: Manage multiple state variables.
9. **Experiment 2.9**: Use useReducer for complex state management.
10. **Experiment 2.10**: Create a simple state management library.

Module 3: React Lifecycle Methods and Hooks (14 hours)

- Component lifecycle methods.
- Introduction to React hooks (useState, useEffect).
- Custom hooks.

Experiments:

1. **Experiment 3.1**: Implement component lifecycle methods in class components.
2. **Experiment 3.2**: Use useEffect to manage side effects.
3. **Experiment 3.3**: Create a custom hook for data fetching.
4. **Experiment 3.4**: Use useState and useEffect together.
5. **Experiment 3.5**: Implement cleanup in useEffect.
6. **Experiment 3.6**: Manage component updates with useEffect.
7. **Experiment 3.7**: Create a custom hook for form handling.
8. **Experiment 3.8**: Implement a custom hook for authentication.
9. **Experiment 3.9**: Use useRef to manage component references.
10. **Experiment 3.10**: Optimize performance with useMemo and useCallback.

Module 4: React Router (12 hours)

- Setting up React Router.
- Navigation and dynamic routing.
- Protected routes.

Experiments:

1. **Experiment 4.1**: Set up React Router in a React project.
2. **Experiment 4.2**: Create routes for different components.
3. **Experiment 4.3**: Implement dynamic routing using URL parameters.
4. **Experiment 4.4**: Create a navigation menu with React Router.
5. **Experiment 4.5**: Implement protected routes using authentication.

6. **Experiment 4.6**: Redirect users based on authentication status.
7. **Experiment 4.7**: Use React Router hooks for navigation.
8. **Experiment 4.8**: Implement nested routing in React Router.
9. **Experiment 4.9**: Handle 404 pages with React Router.
10. **Experiment 4.10**: Optimize route loading with React.lazy.

Module 5: State Management with Context API and Redux (10 hours)

- Context API basics.
- Introduction to Redux.
- Connecting Redux with React components.

Experiments:

1. **Experiment 5.1**: Implement global state management using Context API.
2. **Experiment 5.2**: Create a context provider and consumer.
3. **Experiment 5.3**: Use useContext to access context values.
4. **Experiment 5.4**: Set up Redux in a React project.
5. **Experiment 5.5**: Create Redux actions and reducers.
6. **Experiment 5.6**: Connect Redux state to React components using useSelector.
7. **Experiment 5.7**: Dispatch actions using useDispatch.
8. **Experiment 5.8**: Use Redux middleware for asynchronous actions.
9. **Experiment 5.9**: Integrate Redux DevTools for debugging.
10. **Experiment 5.10**: Compare Context API and Redux for state management.

Module 6: Form Handling and Validation (10 hours)

- Controlled vs. uncontrolled components.
- Form validation with Formik and Yup.
- Handling form submissions.

Experiments:

1. **Experiment 6.1**: Create controlled form components.
2. **Experiment 6.2**: Implement uncontrolled form components.
3. **Experiment 6.3**: Validate form inputs using Formik.
4. **Experiment 6.4**: Use Yup for schema-based form validation.
5. **Experiment 6.5**: Handle form submission events.
6. **Experiment 6.6**: Implement dynamic form validation.
7. **Experiment 6.7**: Create reusable form components with Formik.
8. **Experiment 6.8**: Use Formik hooks for form handling.
9. **Experiment 6.9**: Manage form state with useReducer.
10. **Experiment 6.10**: Integrate form validation with backend APIs.

Module 7: Performance Optimization and Best Practices (12 hours)

- Code splitting and lazy loading.
- Memoization and useMemo/useCallback.
- Debugging and profiling.

Experiments:

1. **Experiment 7.1**: Implement code splitting with React.lazy.
2. **Experiment 7.2**: Use React Suspense for lazy loading components.
3. **Experiment 7.3**: Optimize component rendering with React.memo.
4. **Experiment 7.4**: Use useMemo to memoize expensive calculations.
5. **Experiment 7.5**: Implement useCallback to memoize functions.
6. **Experiment 7.6**: Use React Profiler to measure component performance.
7. **Experiment 7.7**: Debug React applications with React Developer Tools.
8. **Experiment 7.8**: Optimize performance with shouldComponentUpdate.
9. **Experiment 7.9**: Implement lazy loading for images and other assets.
10. **Experiment 7.10**: Apply best practices for React project structure.

Assignments and Projects

1. **Assignment 1**: Build a dynamic front-end application using React.
2. **Assignment 2**: Implement state management with Context API or Redux.
3. **Assignment 3**: Create a multi-page application using React Router.
4. **Assignment 4**: Develop a form with validation using Formik and Yup.
5. **Project**: Integrate React front-end with an Express.js backend to create a full-stack application.

Full Stack Integration and Deployment (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUFD1004	**Full Stack Integration and Deployment**	**03**	**0+2+1**

Course Description: This course focuses on integrating all components of the MERN stack and deploying full-stack applications to production environments.

Course Objectives:

- Connect front-end and back-end components.
- Implement real-time communication and GraphQL.
- Deploy applications using modern CI/CD pipelines.

Course Outcomes (COs):

1. **CO1**: Explain the full-stack architecture and configure CORS and security for connecting React with Express.js API. (Understand, Apply)
2. **CO2**: Implement real-time communication features in MERN applications using WebSockets and Socket.io. (Apply, Analyze)
3. **CO3**: Set up and integrate GraphQL with React and Express.js. (Apply, Create)
4. **CO4**: Perform unit, integration, and end-to-end testing on full-stack applications using Jest and Cypress. (Apply, Evaluate)
5. **CO5**: Deploy MERN applications to production environments using Heroku, AWS, Docker, and CI/CD pipelines. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	2	2	3	-	-	-	-	2	-	2	3	2	3
CO2	3	2	3	2	3	-	-	-	-	2	-	2	3	2	3
CO3	3	2	3	2	3	-	-	-	2	2	1	2	3	3	3
CO4	3	2	3	2	3	-	-	-	2	2	2	2	3	2	3
CO5	3	2	3	2	3	-	-	-	2	2	2	2	3	3	3

Course Syllabus:

Module 1: Connecting MERN Components (10 hours)

- Overview of full-stack architecture.
- Setting up CORS and security configurations.
- Connecting React with Express.js API.

Experiments:

1. **Experiment 1.1**: Set up a full-stack MERN application.
2. **Experiment 1.2**: Configure CORS in an Express.js application.
3. **Experiment 1.3**: Secure API endpoints with JWT.
4. **Experiment 1.4**: Connect a React front-end to an Express.js back-end.
5. **Experiment 1.5**: Implement basic CRUD operations in a MERN application.
6. **Experiment 1.6**: Set up and use environment variables.
7. **Experiment 1.7**: Use Axios to make API requests from React.
8. **Experiment 1.8**: Handle authentication and authorization in a MERN app.
9. **Experiment 1.9**: Secure the MERN application against common vulnerabilities.
10. **Experiment 1.10**: Monitor API performance and log requests.

Module 2: Real-Time Communication with WebSockets (12 hours)

- Introduction to WebSockets.

- Implementing WebSockets with Socket.io.
- Real-time features in MERN applications.

Experiments:

1. **Experiment 2.1**: Set up a WebSocket server using Socket.io.
2. **Experiment 2.2**: Create a real-time chat application.
3. **Experiment 2.3**: Implement real-time notifications in a MERN app.
4. **Experiment 2.4**: Handle real-time data updates with WebSockets.
5. **Experiment 2.5**: Use Socket.io with React for real-time communication.
6. **Experiment 2.6**: Implement WebSocket authentication.
7. **Experiment 2.7**: Integrate real-time features with a database.
8. **Experiment 2.8**: Monitor WebSocket connections and traffic.
9. **Experiment 2.9**: Handle disconnections and reconnections in Socket.io.
10. **Experiment 2.10**: Scale WebSocket applications with multiple servers.

Module 3: Introduction to GraphQL (14 hours)

- Basics of GraphQL.
- Setting up a GraphQL server with Express.js.
- Integrating GraphQL with React.

Experiments:

1. **Experiment 3.1**: Set up a GraphQL server with Express.js.
2. **Experiment 3.2**: Create GraphQL schemas and resolvers.
3. **Experiment 3.3**: Implement queries and mutations in GraphQL.
4. **Experiment 3.4**: Use Apollo Client to connect React with GraphQL.
5. **Experiment 3.5**: Implement authentication in a GraphQL server.
6. **Experiment 3.6**: Use GraphQL to fetch data in a React component.
7. **Experiment 3.7**: Handle errors and validations in GraphQL.
8. **Experiment 3.8**: Integrate GraphQL with MongoDB.
9. **Experiment 3.9**: Optimize GraphQL queries for performance.
10. **Experiment 3.10**: Monitor and debug GraphQL requests.

Module 4: Testing Full Stack Applications (12 hours)

- Unit testing with Jest.
- Integration testing.
- End-to-end testing with Cypress.

Experiments:

1. **Experiment 4.1**: Set up Jest for unit testing in a MERN application.
2. **Experiment 4.2**: Write unit tests for React components.

3. **Experiment 4.3**: Test Express.js routes and controllers with Jest.
4. **Experiment 4.4**: Implement integration tests for the MERN stack.
5. **Experiment 4.5**: Set up Cypress for end-to-end testing.
6. **Experiment 4.6**: Write end-to-end tests for a MERN application with Cypress.
7. **Experiment 4.7**: Mock API requests in tests.
8. **Experiment 4.8**: Use testing libraries to simulate user interactions.
9. **Experiment 4.9**: Measure test coverage and improve test quality.
10. **Experiment 4.10**: Automate tests with CI/CD pipelines.

Module 5: Preparing for Production (10 hours)

- Environment variables and configuration management.
- Logging and monitoring.
- Performance tuning and optimization.

Experiments:

1. **Experiment 5.1**: Manage environment variables in a MERN application.
2. **Experiment 5.2**: Implement logging with Winston in Express.js.
3. **Experiment 5.3**: Monitor application performance with PM2.
4. **Experiment 5.4**: Optimize database queries for performance.
5. **Experiment 5.5**: Use New Relic for monitoring and performance tuning.
6. **Experiment 5.6**: Implement application-level caching.
7. **Experiment 5.7**: Optimize front-end performance with code splitting.
8. **Experiment 5.8**: Set up and monitor health checks for the application.
9. **Experiment 5.9**: Use a performance monitoring tool like Lighthouse.
10. **Experiment 5.10**: Conduct load testing and stress testing.

Module 6: Deployment Strategies (12 hours)

- Deploying with Heroku and AWS.
- Dockerizing MERN applications.
- Setting up CI/CD pipelines with GitHub Actions.

Experiments:

1. **Experiment 6.1**: Deploy a MERN application to Heroku.
2. **Experiment 6.2**: Deploy a MERN application to AWS.
3. **Experiment 6.3**: Set up Docker for a MERN application.
4. **Experiment 6.4**: Create Docker images and containers for deployment.
5. **Experiment 6.5**: Use Docker Compose to manage multi-container applications.
6. **Experiment 6.6**: Set up CI/CD pipelines with GitHub Actions.
7. **Experiment 6.7**: Automate deployments with CI/CD pipelines.

8. **Experiment 6.8**: Monitor deployments and rollbacks.
9. **Experiment 6.9**: Secure Docker containers and images.
10. **Experiment 6.10**: Scale MERN applications using Docker Swarm or Kubernetes.

Product Development (168 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUFD1005	**Product Development**	**06**	**0+0+6**

Course Description: This course focuses on the end-to-end process of product development, from conceptualization to deployment. Students will work on a capstone project integrating all aspects of the MERN stack and deploy their application to a production environment.

Course Objectives:

1. Develop a comprehensive understanding of product development lifecycle.
2. Integrate and apply knowledge from the MERN stack to build a complete application.
3. Deploy applications to a production environment and manage them effectively.

Course Outcomes (COs):

1. **CO1**: Plan and design a comprehensive product using the MERN stack. (Create, Evaluate)
2. **CO2**: Implement front-end and back-end components and integrate them seamlessly. (Apply, Analyze)
3. **CO3**: Incorporate real-time features and GraphQL in the application. (Apply, Create)
4. **CO4**: Perform thorough testing and debugging of the product. (Apply, Evaluate)
5. **CO5**: Deploy the application to a production environment and manage its performance and security. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
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**

Course Syllabus:

Module 1: Product Conceptualization and Planning (20 hours)

- Understanding the product development lifecycle.
- Requirements gathering and analysis.
- Designing the product architecture and UI/UX.

Module 2: Front-End Development with React (30 hours)

- Building the user interface with React.
- State management with Context API and Redux.
- Implementing routing and form handling.

Module 3: Back-End Development with Node.js and Express.js (30 hours)

- Setting up the server with Node.js and Express.js.
- Building RESTful APIs and GraphQL endpoints.
- Implementing authentication and authorization.

Module 4: Integration and Testing (30 hours)

- Integrating front-end and back-end components.
- Unit, integration, and end-to-end testing.
- Debugging and performance optimization.

Module 5: Deployment and Production Management (20 hours)

- Preparing the application for deployment.
- Setting up CI/CD pipelines.
- Managing production environment and security.

Assignments and Projects

1. **Assignment 1**: Develop a capstone project integrating all aspects of the MERN stack.
2. **Assignment 2**: Deploy the application to a production environment.

Domain Track: Generative AI (12 Credits) (0+8+4)

Fundamentals of Generative AI (28 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGA1011	Fundamentals of Generative AI	01	0+1+0

Course Description: This course introduces the fundamental concepts and techniques in generative AI, providing a solid foundation for understanding and building generative models.

Course Objectives:

1. Understand the basic concepts and history of generative AI.
2. Learn about various generative models and their applications.
3. Gain proficiency in basic machine learning concepts relevant to generative AI.

Course Outcomes (COs):

1. **CO1**: Explain the basic concepts and history of generative AI. (Understand, Remember)
2. **CO2**: Distinguish between different types of machine learning, including supervised and unsupervised learning. (Understand, Analyze)
3. **CO3**: Describe the architecture and functioning of various generative models. (Understand, Apply)
4. **CO4**: Understand the basic concepts and challenges of training Generative Adversarial Networks (GANs). (Understand, Analyze)
5. **CO5**: Evaluate and compare different generative models based on their performance and applications. (Evaluate, Analyze)

CO-PO-PSO Mapping:

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

Module 1: Introduction to Generative AI (2 hours)

- Overview of AI and machine learning.
- History and evolution of generative AI.
- Applications of generative AI in various fields.

Module 2: Basic Machine Learning Concepts (2 hours)

- Supervised vs. unsupervised learning.
- Introduction to neural networks and deep learning.
- Basic optimization techniques.

Module 3: Generative Models (2 hours)

- Introduction to generative models.
- Gaussian Mixture Models (GMMs).
- Hidden Markov Models (HMMs).

Module 4: Generative Adversarial Networks (GANs) (2 hours)

- Basic concepts of GANs.
- Architecture of GANs: Generator and Discriminator.
- Training GANs and common challenges.

Module 5: Variational Autoencoders (VAEs) (2 hours)

- Introduction to VAEs.
- VAE architecture and loss functions.
- Applications of VAEs.

Module 6: Other Generative Models (2 hours)

- Restricted Boltzmann Machines (RBMs).
- Normalizing Flows.
- Autoregressive models.

Module 7: Applications and Future of Generative AI (2 hours)

- Current applications of generative AI in various fields.
- Ethical considerations and challenges in generative AI.
- Future directions and advancements in generative AI.

Assignments and Projects:

- **Assignment 1**: Implement basic generative models using Python.
- **Assignment 2**: Compare and contrast different generative models based on their performance and applications.

Relevant Job Roles:

- Machine Learning Engineer: Understand and implement foundational generative models.
- Data Scientist: Analyze and apply generative AI techniques to data.
- AI Research Scientist: Study and develop new generative AI methods.
- AI Developer: Build and integrate basic generative AI applications.

Advanced Techniques in Generative AI (54 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGA1012	Advanced Techniques in Generative AI	02	0+2+0

Course Description: This course delves into advanced generative AI techniques, exploring state-of-the-art models and their applications in various domains.

Course Objectives:

- Master advanced techniques in generative AI.
- Understand and implement state-of-the-art generative models.
- Explore applications of generative AI in creative and scientific fields.

Course Outcomes (COs):

- **CO1**: Explain the architecture and functionality of advanced generative models such as DCGANs, cGANs, StyleGAN, and BigGAN. (Understand, Analyze)
- **CO2**: Implement sequence generation models using RNNs and LSTMs, and apply GANs and VAEs for sequence generation. (Apply, Create)
- **CO3**: Understand and implement attention mechanisms and transformer architectures, including GPT. (Understand, Apply)
- **CO4**: Describe diffusion models and their applications in image and text generation. (Understand, Analyze)
- **CO5**: Evaluate and apply advanced training techniques for GANs and VAEs, and understand ethical considerations in generative AI. (Evaluate, Apply)

CO-PO-PSO Mapping:

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

Module 1: Deep Generative Models (9 hours)

- Deep Convolutional GANs (DCGANs).
- Conditional GANs (cGANs).
- StyleGAN and BigGAN.

Module 2: Sequence Generation Models (9 hours)

- Recurrent Neural Networks (RNNs).
- Long Short-Term Memory (LSTM) networks.
- Generating sequences with GANs and VAEs.

Module 3: Attention Mechanisms and Transformers (9 hours)

- Introduction to attention mechanisms.
- Transformer architecture.
- Generative Pre-trained Transformers (GPT).

Module 4: Diffusion Models (9 hours)

- Introduction to diffusion models.
- Training and inference processes.
- Applications in image and text generation.

Module 5: Advanced Topics in GANs and VAEs (9 hours)

- CycleGAN and Pix2Pix.
- Denoising Autoencoders.
- Advanced training techniques and loss functions.

Module 6: Applications and Ethics (9 hours)

- Applications in art, music, and design.
- Ethical considerations and biases in generative AI.

- Fairness, accountability, and transparency.

Assignments and Projects:

- **Assignment 1**: Implement advanced generative models.
- **Assignment 2**: Explore creative applications of generative AI.

Relevant Job Roles:

- **Machine Learning Engineer**: Develop and optimize advanced generative models.
- **Data Scientist**: Apply advanced generative techniques to complex datasets.
- **AI Research Scientist**: Conduct cutting-edge research in generative AI.
- **AI Developer**: Create sophisticated generative AI applications for various industries.

Real-Time Generative AI (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGA1013	Real-Time Generative AI	03	0+1+2

Course Description: This course focuses on the practical aspects of implementing generative AI models, including programming, optimization, and deployment.

Course Objectives:

- Gain hands-on experience with generative AI tools and libraries.
- Learn to optimize and fine-tune generative models.
- Deploy generative AI models in real-world applications.

Course Outcomes (COs):

1. **CO1**: Implement basic neural networks using TensorFlow and PyTorch, and utilize pre-trained models. (Apply, Create)
2. **CO2**: Apply training techniques and hyperparameter tuning to optimize generative models. (Apply, Analyze)
3. **CO3**: Implement transfer learning and fine-tune pre-trained generative models for specific tasks. (Apply, Create)
4. **CO4**: Evaluate generative models using common metrics and interpret the results. (Evaluate, Analyze)
5. **CO5**: Deploy generative AI models using cloud platforms and build APIs for real-world applications. (Apply, Create)

CO-PO-PSO Mapping:

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

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

Course Syllabus:

Module 1: Programming Tools and Libraries (6 hours)

- Introduction to TensorFlow and PyTorch.
- Implementing basic neural networks.
- Utilizing pre-trained models.

Experiments:

1. **Experiment 1.1**: Set up TensorFlow and PyTorch environments.
2. **Experiment 1.2**: Implement a simple feedforward neural network in TensorFlow.
3. **Experiment 1.3**: Implement a simple feedforward neural network in PyTorch.
4. **Experiment 1.4**: Load and utilize pre-trained models from TensorFlow Hub.
5. **Experiment 1.5**: Load and utilize pre-trained models from PyTorch Hub.
6. **Experiment 1.6**: Compare the performance of TensorFlow and PyTorch models.
7. **Experiment 1.7**: Implement a Convolutional Neural Network (CNN) in TensorFlow.
8. **Experiment 1.8**: Implement a Convolutional Neural Network (CNN) in PyTorch.
9. **Experiment 1.9**: Fine-tune a pre-trained model for a specific task in TensorFlow.
10. **Experiment 1.10**: Fine-tune a pre-trained model for a specific task in PyTorch.

Module 2: Model Training and Optimization (6 hours)

- Training techniques and strategies.
- Hyperparameter tuning.
- Optimization algorithms (SGD, Adam, etc.).

Experiments:

1. **Experiment 2.1**: Train a neural network using SGD in TensorFlow.
2. **Experiment 2.2**: Train a neural network using Adam in PyTorch.
3. **Experiment 2.3**: Implement learning rate scheduling in TensorFlow.
4. **Experiment 2.4**: Implement learning rate scheduling in PyTorch.
5. **Experiment 2.5**: Perform hyperparameter tuning using TensorFlow.
6. **Experiment 2.6**: Perform hyperparameter tuning using PyTorch.
7. **Experiment 2.7**: Use TensorBoard to visualize training metrics.

8. **Experiment 2.8**: Use PyTorch's visualization tools to monitor training.
9. **Experiment 2.9**: Apply gradient clipping in TensorFlow.
10. **Experiment 2.10**: Apply gradient clipping in PyTorch.

Module 3: Fine-Tuning and Transfer Learning (6 hours)

- Transfer learning concepts.
- Fine-tuning pre-trained generative models.
- Practical examples and case studies.

Experiments:

1. **Experiment 3.1**: Implement transfer learning using a pre-trained CNN in TensorFlow.
2. **Experiment 3.2**: Implement transfer learning using a pre-trained CNN in PyTorch.
3. **Experiment 3.3**: Fine-tune a GAN for image generation in TensorFlow.
4. **Experiment 3.4**: Fine-tune a GAN for image generation in PyTorch.
5. **Experiment 3.5**: Transfer learning for text generation using GPT in TensorFlow.
6. **Experiment 3.6**: Transfer learning for text generation using GPT in PyTorch.
7. **Experiment 3.7**: Case study: Fine-tuning a VAE for image reconstruction in TensorFlow.
8. **Experiment 3.8**: Case study: Fine-tuning a VAE for image reconstruction in PyTorch.
9. **Experiment 3.9**: Implement domain adaptation using transfer learning in TensorFlow.
10. **Experiment 3.10**: Implement domain adaptation using transfer learning in PyTorch.

Module 4: Evaluation and Metrics (6 hours)

- Evaluating generative models.
- Common metrics (FID, Inception Score).
- Visualizing and interpreting results.

Experiments:

1. **Experiment 4.1**: Evaluate a GAN using the Frechet Inception Distance (FID) score.
2. **Experiment 4.2**: Evaluate a GAN using the Inception Score.
3. **Experiment 4.3**: Visualize GAN outputs using TensorBoard in TensorFlow.
4. **Experiment 4.4**: Visualize GAN outputs using Matplotlib in PyTorch.
5. **Experiment 4.5**: Implement precision and recall metrics for generative models.
6. **Experiment 4.6**: Compare the performance of different generative models using FID.
7. **Experiment 4.7**: Visualize latent space interpolations in VAEs.
8. **Experiment 4.8**: Use t-SNE for visualizing high-dimensional data in generative models.
9. **Experiment 4.9**: Interpret model outputs using SHAP values in TensorFlow.
10. **Experiment 4.10**: Interpret model outputs using SHAP values in PyTorch.

Module 5: Deployment of Generative Models (6 hours)

- Model deployment strategies.
- Using cloud platforms for deployment (AWS, Google Cloud).
- Building APIs for generative models.

Experiments:

1. **Experiment 5.1**: Deploy a generative model on AWS SageMaker.
2. **Experiment 5.2**: Deploy a generative model on Google Cloud AI Platform.
3. **Experiment 5.3**: Create a REST API for a generative model using Flask.
4. **Experiment 5.4**: Create a REST API for a generative model using FastAPI.
5. **Experiment 5.5**: Use Docker to containerize a generative model.
6. **Experiment 5.6**: Deploy a Dockerized model to AWS ECS.
7. **Experiment 5.7**: Deploy a Dockerized model to Google Kubernetes Engine.
8. **Experiment 5.8**: Implement model versioning and rollback strategies.
9. **Experiment 5.9**: Monitor model performance in production using Prometheus.
10. **Experiment 5.10**: Set up continuous deployment for generative models with GitHub Actions.

Module 6: Practical Applications (6 hours)

- Integrating generative models into applications.
- Case studies in various industries.
- Future trends and research directions.

Experiments:

1. **Experiment 6.1**: Integrate a generative model into a web application.
2. **Experiment 6.2**: Develop a chatbot using a generative model.
3. **Experiment 6.3**: Create an image generation application using GANs.
4. **Experiment 6.4**: Implement a music generation system using RNNs.
5. **Experiment 6.5**: Use generative models for data augmentation in computer vision.
6. **Experiment 6.6**: Apply generative models to generate synthetic data for training.
7. **Experiment 6.7**: Case study: Generative AI in healthcare for medical imaging.
8. **Experiment 6.8**: Case study: Generative AI in finance for synthetic data generation.
9. **Experiment 6.9**: Research trends: Latest advancements in generative AI.
10. **Experiment 6.10**: Explore ethical implications and bias in generative models.

Assignments and Projects:

- **Assignment 1**: Implement and fine-tune a generative model for a specific application.
- **Assignment 2**: Deploy a generative AI model as a web service.

Relevant Job Roles:

- **Machine Learning Engineer**: Train, optimize, and deploy generative AI models.

- **Data Scientist**: Evaluate and improve generative models based on performance metrics.
- **AI Developer**: Implement and deploy generative AI solutions in production environments.

Research and Development in Generative AI (54 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGA1014	Research and Development in Generative AI	02	0+2+0

Course Description: This course emphasizes research methodologies and the development of innovative solutions in the field of generative AI.

Course Objectives:

- Conduct research in generative AI.
- Develop innovative solutions and contribute to the field.
- Learn about recent advancements and open research areas.

Course Outcomes (COs):

- **CO1**: Understand and apply AI research methodologies, including literature review and critical analysis. (Understand, Apply)
- **CO2**: Analyze recent advances and emerging trends in generative AI. (Analyze, Evaluate)
- **CO3**: Formulate research questions and design experiments to develop innovative solutions in generative AI. (Create, Evaluate)
- **CO4**: Collaborate effectively in research teams and present research findings. (Apply, Evaluate)
- **CO5**: Identify open problems in generative AI and propose future research directions. (Analyze, Create)

CO-PO-PSO Mapping:

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

CO4	3	3	3	3	3	-	-	3	2	3	3	3	3
CO5	3	3	3	3	3	-	-	3	2	3	3	3	3

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

Course Syllabus:

Module 1: Research Methodologies (9 hours)

- Introduction to AI research methodologies.
- Literature review and critical analysis.
- Identifying research gaps.

Experiments:

1. **Experiment 1.1**: Conduct a literature review on a generative AI topic.
2. **Experiment 1.2**: Critically analyze a seminal paper in generative AI.
3. **Experiment 1.3**: Identify research gaps from recent studies.
4. **Experiment 1.4**: Develop a research proposal based on identified gaps.
5. **Experiment 1.5**: Present a literature review summary.
6. **Experiment 1.6**: Create a mind map of current research trends.
7. **Experiment 1.7**: Use reference management software for organizing literature.
8. **Experiment 1.8**: Write an abstract summarizing a research paper.
9. **Experiment 1.9**: Analyze the methodology section of a research paper.
10. **Experiment 1.10**: Conduct a peer review of a literature review draft.

Module 2: Recent Advances in Generative AI (9 hours)

- Review of recent papers and studies.
- Advanced topics in GANs, VAEs, and Transformers.
- Emerging trends and technologies.

Experiments:

1. **Experiment 2.1**: Summarize key findings from a recent GANs paper.
2. **Experiment 2.2**: Present a recent study on VAEs.
3. **Experiment 2.3**: Analyze a paper on Transformer models.
4. **Experiment 2.4**: Identify emerging trends in generative AI from recent publications.
5. **Experiment 2.5**: Conduct a comparative analysis of two recent papers.
6. **Experiment 2.6**: Present an advanced topic in GANs to the class.
7. **Experiment 2.7**: Review a paper on the latest advancements in VAEs.
8. **Experiment 2.8**: Create a presentation on recent trends in Transformer models.
9. **Experiment 2.9**: Analyze the impact of emerging technologies on generative AI.
10. **Experiment 2.10**: Write a summary of recent advances in generative AI.

Module 3: Developing Innovative Solutions (9 hours)

- Formulating research questions.
- Designing experiments and studies.
- Prototyping and iterative development.

Experiments:

1. **Experiment 3.1**: Formulate a research question in generative AI.
2. **Experiment 3.2**: Design an experiment to test a generative AI hypothesis.
3. **Experiment 3.3**: Develop a prototype for a generative AI model.
4. **Experiment 3.4**: Conduct an initial experiment and gather data.
5. **Experiment 3.5**: Analyze experimental results and refine the prototype.
6. **Experiment 3.6**: Iterate on the prototype based on feedback.
7. **Experiment 3.7**: Write a research methodology section for a paper.
8. **Experiment 3.8**: Create a project plan for iterative development.
9. **Experiment 3.9**: Present the progress of an innovative solution.
10. **Experiment 3.10**: Conduct a peer review of a research design proposal.

Module 4: Collaborative Research (9 hours)

- Collaboration tools and platforms.
- Working in research teams.
- Presenting and publishing research.

Experiments:

1. **Experiment 4.1**: Use a collaboration platform (e.g., Overleaf) to write a research paper.
2. **Experiment 4.2**: Participate in a virtual research team meeting.
3. **Experiment 4.3**: Present research findings to the class.
4. **Experiment 4.4**: Write a draft research paper for submission.
5. **Experiment 4.5**: Use a version control system (e.g., Git) for research collaboration.
6. **Experiment 4.6**: Conduct a peer review of a draft research paper.
7. **Experiment 4.7**: Collaborate on a research presentation.
8. **Experiment 4.8**: Submit a research paper to a conference or journal.
9. **Experiment 4.9**: Create a poster presentation for a research project.
10. **Experiment 4.10**: Conduct a mock research presentation and receive feedback.

Module 5: Case Studies and Applications (9 hours)

- Case studies of successful generative AI applications.
- Analysis of failures and challenges.
- Learning from industry and academia.

Experiments:

1. **Experiment 5.1**: Analyze a case study of a successful generative AI application.
2. **Experiment 5.2**: Present a case study of a failed generative AI project.
3. **Experiment 5.3**: Identify lessons learned from a successful AI project.
4. **Experiment 5.4**: Conduct a SWOT analysis of a generative AI application.
5. **Experiment 5.5**: Review an industry report on generative AI.
6. **Experiment 5.6**: Create a case study of a generative AI project.
7. **Experiment 5.7**: Analyze the challenges faced by a generative AI application.
8. **Experiment 5.8**: Present findings from a case study analysis.
9. **Experiment 5.9**: Compare academic and industry approaches to generative AI.
10. **Experiment 5.10**: Write a case study report on a generative AI application.

Module 6: Future Directions (9 hours)

- Open problems in generative AI.
- Potential impact on society and industries.
- Preparing for a career in AI research.

Experiments:

1. **Experiment 6.1**: Identify open research questions in generative AI.
2. **Experiment 6.2**: Write a research proposal for an open problem.
3. **Experiment 6.3**: Present potential societal impacts of generative AI.
4. **Experiment 6.4**: Analyze the impact of generative AI on a specific industry.
5. **Experiment 6.5**: Create a roadmap for future research in generative AI.
6. **Experiment 6.6**: Review ethical considerations in generative AI research.
7. **Experiment 6.7**: Develop a career plan for AI research.
8. **Experiment 6.8**: Conduct a mock interview for an AI research position.
9. **Experiment 6.9**: Write a statement of research interests.
10. **Experiment 6.10**: Present a vision for the future of generative AI research.

Assignments and Projects:

- **Assignment 1**: Conduct a mini-research project.
- **Assignment 2**: Present findings and propose future research directions.

Relevant Job Roles:

- **AI Research Scientist**: Conduct advanced research and contribute to scientific knowledge in generative AI.
- **Machine Learning Engineer**: Innovate and develop new generative models and techniques.
- **Data Scientist**: Apply research methodologies to analyze and improve generative models.

Capstone Project in Generative AI (112 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGA1015	Capstone Project in Generative AI	04	0+2+2

Course Description: The capstone project is a comprehensive, hands-on project that synthesizes the knowledge and skills acquired throughout the program. Students will work on real-world problems, develop innovative solutions, and present their work.

Course Objectives:

- Apply theoretical knowledge to practical problems.
- Develop a comprehensive generative AI project.
- Demonstrate proficiency in model development, implementation, and deployment.

Project Phases:

Phase 1: Project Proposal and Planning (Weeks 1-2) (18 Hours)

- Identify a real-world problem or research question.
- Conduct a literature review.
- Develop a project proposal outlining objectives, methodology, and timeline.

Deliverables:

1. Project proposal document.
2. Literature review summary.

Experiments:

1. **Experiment 1.1**: Identify and define a real-world problem suitable for a generative AI solution.
2. **Experiment 1.2**: Conduct a comprehensive literature review on the chosen problem.
3. **Experiment 1.3**: Develop a detailed project proposal with objectives, methodology, and timeline.

Phase 2: Research and Development (Weeks 3-6) (36 Hours)

- Conduct necessary research and data collection.
- Develop and train generative models.
- Iterate and refine models based on feedback and results.

Deliverables:

1. Research and development logs.
2. Initial model prototypes.

Experiments:

1. **Experiment 2.1**: Collect and preprocess data required for the project.
2. **Experiment 2.2**: Develop an initial generative model.
3. **Experiment 2.3**: Train the generative model using collected data.
4. **Experiment 2.4**: Evaluate the initial model and gather feedback.
5. **Experiment 2.5**: Refine the model based on feedback and re-evaluate.

Phase 3: Implementation and Testing (Weeks 7-10) (36 Hours)

- Implement the generative AI solution.
- Test the model rigorously using appropriate metrics.
- Optimize and fine-tune the solution.

Deliverables:

1. Finalized generative model.
2. Testing and evaluation reports.

Experiments:

1. **Experiment 3.1**: Implement the generative AI model into a functional system.
2. **Experiment 3.2**: Conduct extensive testing using relevant metrics (e.g., FID, Inception Score).
3. **Experiment 3.3**: Optimize the model for performance and accuracy.
4. **Experiment 3.4**: Fine-tune hyperparameters to enhance model performance.
5. **Experiment 3.5**: Document the testing process and results.

Phase 4: Deployment and Presentation (Weeks 11-12) (24 Hours)

- Deploy the solution to a production environment.
- Prepare a comprehensive project report.
- Present the project to peers and instructors.

Deliverables:

1. Deployed generative AI application.
2. Final project report.
3. Project presentation.

Experiments:

1. **Experiment 4.1**: Deploy the generative AI model to a cloud platform (e.g., AWS, Google Cloud).
2. **Experiment 4.2**: Ensure the deployed model is accessible and functional.
3. **Experiment 4.3**: Prepare a detailed project report documenting the entire process.
4. **Experiment 4.4**: Develop a presentation summarizing the project findings and outcomes.
5. **Experiment 4.5**: Conduct a mock presentation and receive feedback.

Domain Track: Data Analytics and Machine Learning (0+6+12 Credits)

Machine Learning for Predictive Analytics (112 hrs)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUML1021	Machine Learning for Predictive Analytics	04	0+2+2

Course Description: This course covers the foundational concepts and techniques of machine learning with a focus on predictive analytics. Students will learn to build, evaluate, and interpret predictive models using various machine learning algorithms.

Course Objectives:

1. Understand the core concepts of predictive analytics.
2. Develop and evaluate predictive models using supervised and unsupervised learning algorithms.
3. Deploy machine learning models and interpret their outcomes in real-world applications.

Course Outcomes (COs):

1. **CO1**: Explain the fundamental concepts and applications of predictive analytics. (Understand, Remember)
2. **CO2**: Apply data preprocessing techniques to prepare data for machine learning models. (Apply, Analyze)
3. **CO3**: Build and evaluate predictive models using supervised learning algorithms. (Apply, Evaluate)
4. **CO4**: Utilize advanced machine learning techniques and unsupervised learning methods. (Apply, Analyze)
5. **CO5**: Deploy machine learning models and interpret their outcomes in real-world applications. (Apply, Create)

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	1	2	-	-	-	-	-	-	1	3	2	3
CO2	3	3	3	2	2	-	-	-	1	-	-	2	3	3	3
CO3	3	3	3	2	3	-	-	-	1	-	-	2	3	3	3
CO4	3	3	3	3	3	1	1	-	2	-	2	2	3	3	3

CO5	3	3	3	3	3	2	2	3	2	2	2	3	3	3	3

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

Syllabus:

Module 1: Introduction to Predictive Analytics (16 hours)

- Overview of Predictive Analytics
- Applications and Case Studies

Experiments:

1. **Experiment 1.1**: Research and present a case study on predictive analytics.
2. **Experiment 1.2**: Identify and discuss real-world applications of predictive analytics.
3. **Experiment 1.3**: Conduct a literature review on predictive analytics methodologies.
4. **Experiment 1.4**: Develop a project proposal for a predictive analytics project.

Module 2: Data Preprocessing (6 hours)

- Data Cleaning
- Feature Engineering
- Data Transformation

Experiments:

1. **Experiment 2.1**: Clean a raw dataset and handle missing values.
2. **Experiment 2.2**: Perform feature engineering on a dataset.
3. **Experiment 2.3**: Apply data transformation techniques such as normalization and standardization.
4. **Experiment 2.4**: Encode categorical variables in a dataset.
5. **Experiment 2.5**: Implement feature selection techniques.
6. **Experiment 2.6**: Split a dataset into training and testing sets.

Module 3: Supervised Learning Algorithms (16 hours)

- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines

Experiments:

1. **Experiment 3.1**: Implement linear regression on a dataset.
2. **Experiment 3.2**: Apply logistic regression for binary classification.
3. **Experiment 3.3**: Build a decision tree model and visualize it.
4. **Experiment 3.4**: Train a random forest model and evaluate its performance.
5. **Experiment 3.5**: Use support vector machines for classification tasks.
6. **Experiment 3.6**: Compare the performance of different supervised learning algorithms.
7. **Experiment 3.7**: Conduct a hyperparameter tuning experiment for a random forest model.
8. **Experiment 3.8**: Implement a multi-class classification using logistic regression.
9. **Experiment 3.9**: Evaluate model performance using confusion matrix and classification report.
10. **Experiment 3.10**: Document the implementation and evaluation process of a supervised learning model.

Module 4: Model Evaluation and Selection (16 hours)

- Train-Test Split
- Cross-Validation
- Performance Metrics (Accuracy, Precision, Recall, F1 Score, AUC-ROC)

Experiments:

1. **Experiment 4.1**: Split a dataset into training and testing sets and evaluate model performance.
2. **Experiment 4.2**: Implement k-fold cross-validation for model evaluation.
3. **Experiment 4.3**: Calculate and interpret accuracy, precision, recall, and F1 score.
4. **Experiment 4.4**: Plot and analyze ROC curves and calculate AUC.
5. **Experiment 4.5**: Perform model selection using cross-validation scores.
6. **Experiment 4.6**: Compare different performance metrics for model evaluation.

Module 5: Advanced Machine Learning Techniques (16 hours)

- Ensemble Methods (Bagging, Boosting)
- Gradient Boosting Machines (GBM)
- Hyperparameter Tuning

Experiments:

1. **Experiment 5.1**: Implement bagging with decision trees.
2. **Experiment 5.2**: Apply boosting techniques using AdaBoost.
3. **Experiment 5.3**: Train a gradient boosting machine model.
4. **Experiment 5.4**: Compare the performance of bagging, boosting, and GBM.
5. **Experiment 5.5**: Perform hyperparameter tuning for a GBM model.
6. **Experiment 5.6**: Implement stacking ensemble method.
7. **Experiment 5.7**: Analyze the impact of hyperparameters on model performance.

8. **Experiment 5.8**: Use random search for hyperparameter optimization.
9. **Experiment 5.9**: Evaluate the performance of ensemble methods on different datasets.
10. **Experiment 5.10**: Document and present the advanced techniques used in model building.

Module 6: Unsupervised Learning (16 hours)

- Clustering (K-Means, Hierarchical Clustering)
- Dimensionality Reduction (PCA, LDA)

Experiments:

1. **Experiment 6.1**: Implement K-Means clustering on a dataset.
2. **Experiment 6.2**: Perform hierarchical clustering and visualize dendrograms.
3. **Experiment 6.3**: Apply PCA for dimensionality reduction.
4. **Experiment 6.4**: Use LDA for dimensionality reduction and classification.
5. **Experiment 6.5**: Compare the results of K-Means and hierarchical clustering.
6. **Experiment 6.6**: Visualize clusters using scatter plots and pair plots.
7. **Experiment 6.7**: Evaluate clustering performance using silhouette scores.
8. **Experiment 6.8**: Implement t-SNE for high-dimensional data visualization.
9. **Experiment 6.9**: Perform cluster analysis on a real-world dataset.
10. **Experiment 6.10**: Document and present the findings of unsupervised learning experiments.

Module 7: Model Deployment (16 hours)

- Introduction to Model Deployment
- Tools and Techniques for Deploying Models

Experiments:

1. **Experiment 7.1**: Save and load machine learning models using joblib or pickle.
2. **Experiment 7.2**: Deploy a machine learning model using Flask.
3. **Experiment 7.3**: Create a REST API for a predictive model.
4. **Experiment 7.4**: Use Docker to containerize a machine learning application.
5. **Experiment 7.5**: Deploy a model to a cloud platform (e.g., AWS, Heroku).
6. **Experiment 7.6**: Implement model versioning and rollback strategies.
7. **Experiment 7.7**: Monitor model performance in a production environment.
8. **Experiment 7.8**: Set up continuous integration and deployment for machine learning models.
9. **Experiment 7.9**: Test and validate the deployed model API.
10. **Experiment 7.10**: Document and present the model deployment process.

Assignments and Projects:

1. **Assignment 1**: Conduct a mini-research project on a predictive analytics topic.
2. **Assignment 2**: Develop and deploy a predictive model for a real-world application.

Relevant Job Roles:

- **Machine Learning Engineer**: Build, optimize, and deploy predictive models.
- **Data Scientist**: Analyze data and develop predictive models.
- **AI Developer**: Implement and deploy machine learning solutions.

Textbooks and References:

· "Introduction to Statistical Learning" by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani

· "Machine Learning Yearning" by Andrew Ng

Course Code	Course Title	Credits	Type (T+P+Pj)
CUML1022	**Deep Learning for Image Analytics**	**04**	**0+2+2**

Deep Learning for Image Analytics (112 hrs)

Course Description: This course introduces deep learning concepts and techniques specifically applied to image analytics. Students will gain hands-on experience in developing and deploying convolutional neural networks (CNNs) for image classification, object detection, and image generation.

Course Objectives:

1. Understand deep learning fundamentals and CNN architectures.
2. Develop and evaluate CNNs for image classification and object detection.
3. Implement generative models and deploy deep learning applications using modern frameworks.

Course Outcomes (COs):

1. **CO1**: Explain the basic concepts of neural networks, including activation functions and backpropagation. (Understand, Remember)
2. **CO2**: Design and implement CNN architectures for various image analytics tasks. (Apply, Create)
3. **CO3**: Utilize advanced CNN architectures and transfer learning for enhanced performance. (Apply, Analyze)

4. **CO4**: Develop and evaluate models for image classification, object detection, and segmentation. (Apply, Evaluate)
5. **CO5**: Implement generative models and deploy deep learning applications using TensorFlow, Keras, and PyTorch. (Apply, Create)

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	3	2	3
CO2	3	3	3	2	2	-	-	-	1	-	-	2	3	3	3
CO3	3	3	3	2	3	-	-	-	1	-	-	2	3	3	3
CO4	3	3	3	3	3	1	1	-	2	-	2	2	3	3	3
CO5	3	3	3	3	3	2	2	3	2	2	2	3	3	3	3

CO-PO-PSO Mapping

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

Syllabus:

Module 1: Introduction to Deep Learning (16 hours)

- Basics of Neural Networks
- Activation Functions
- Backpropagation and Gradient Descent

Experiments:

1. **Experiment 1.1**: Implement a simple neural network from scratch.
2. **Experiment 1.2**: Explore various activation functions and their effects on model performance.
3. **Experiment 1.3**: Implement backpropagation and gradient descent for a basic neural network.
4. **Experiment 1.4**: Visualize the training process of a neural network.
5. **Experiment 1.5**: Compare different optimization algorithms in training a neural network.
6. **Experiment 1.6**: Implement a neural network using a deep learning framework (e.g., TensorFlow).
7. **Experiment 1.7**: Analyze the impact of different learning rates on training.
8. **Experiment 1.8**: Implement dropout and observe its effect on model performance.
9. **Experiment 1.9**: Conduct experiments with different network architectures.

10. **Experiment 1.10**: Document and present the results of neural network experiments.

Module 2: Convolutional Neural Networks (CNNs) (16 hours)

- Architecture of CNNs
- Convolutional Layers, Pooling Layers
- Fully Connected Layers

Experiments:

1. **Experiment 2.1**: Build a basic CNN for image classification.
2. **Experiment 2.2**: Explore the effects of different pooling techniques.
3. **Experiment 2.3**: Visualize feature maps and filters in a CNN.
4. **Experiment 2.4**: Implement a CNN with multiple convolutional and pooling layers.
5. **Experiment 2.5**: Compare the performance of a CNN with and without fully connected layers.
6. **Experiment 2.6**: Experiment with different activation functions in a CNN.
7. **Experiment 2.7**: Train a CNN on a small image dataset and evaluate its performance.
8. **Experiment 2.8**: Implement dropout and batch normalization in a CNN.
9. **Experiment 2.9**: Optimize a CNN for better performance.
10. **Experiment 2.10**: Document and present the results of CNN experiments.

Module 3: Advanced CNN Architectures (16 hours)

- AlexNet, VGG, ResNet, Inception
- Transfer Learning and Pre-trained Models

Experiments:

1. **Experiment 3.1**: Implement AlexNet for image classification.
2. **Experiment 3.2**: Implement VGG for image classification.
3. **Experiment 3.3**: Explore ResNet and its residual connections.
4. **Experiment 3.4**: Implement Inception architecture and analyze its performance.
5. **Experiment 3.5**: Apply transfer learning using a pre-trained model.
6. **Experiment 3.6**: Fine-tune a pre-trained model for a specific task.
7. **Experiment 3.7**: Compare the performance of different advanced CNN architectures.
8. **Experiment 3.8**: Visualize and interpret the features learned by advanced CNNs.
9. **Experiment 3.9**: Experiment with combining multiple pre-trained models.
10. **Experiment 3.10**: Document and present the results of advanced CNN experiments.

Module 4: Image Classification (16 hours)

- Data Augmentation
- Training and Evaluating CNNs for Image Classification

Experiments:

1. **Experiment 4.1**: Implement data augmentation techniques.
2. **Experiment 4.2**: Train a CNN on augmented data and evaluate its performance.
3. **Experiment 4.3**: Implement a complete pipeline for image classification.
4. **Experiment 4.4**: Evaluate the model using confusion matrix, accuracy, and other metrics.
5. **Experiment 4.5**: Compare the effects of different data augmentation techniques on model performance.
6. **Experiment 4.6**: Experiment with different loss functions and optimization techniques for image classification.
7. **Experiment 4.7**: Implement early stopping and model checkpointing during training.
8. **Experiment 4.8**: Conduct hyperparameter tuning for the image classification model.
9. **Experiment 4.9**: Analyze the model's performance on different subsets of the dataset.
10. **Experiment 4.10**: Document and present the results of image classification experiments.

Module 5: Object Detection (16 hours)

- R-CNN, Fast R-CNN, Faster R-CNN
- YOLO (You Only Look Once)

Experiments:

1. **Experiment 5.1**: Implement R-CNN for object detection.
2. **Experiment 5.2**: Implement Fast R-CNN and compare its performance with R-CNN.
3. **Experiment 5.3**: Implement Faster R-CNN and evaluate its speed and accuracy.
4. **Experiment 5.4**: Implement YOLO for real-time object detection.
5. **Experiment 5.5**: Compare the performance of YOLO with Faster R-CNN.
6. **Experiment 5.6**: Experiment with different object detection datasets and evaluate the models.
7. **Experiment 5.7**: Implement data augmentation for object detection.
8. **Experiment 5.8**: Evaluate the object detection model using precision-recall curves.
9. **Experiment 5.9**: Implement a complete pipeline for object detection.
10. **Experiment 5.10**: Document and present the results of object detection experiments.

Module 6: Image Segmentation (16 hours)

- Semantic Segmentation
- U-Net Architecture

Experiments:

1. **Experiment 6.1**: Implement semantic segmentation using a basic CNN.
2. **Experiment 6.2**: Implement U-Net for image segmentation.
3. **Experiment 6.3**: Train and evaluate a segmentation model on a medical imaging dataset.

4. **Experiment 6.4**: Experiment with different loss functions for image segmentation.
5. **Experiment 6.5**: Compare the performance of semantic segmentation and U-Net.
6. **Experiment 6.6**: Visualize the segmentation results and evaluate their accuracy.
7. **Experiment 6.7**: Implement data augmentation for image segmentation.
8. **Experiment 6.8**: Experiment with different architectures for image segmentation.
9. **Experiment 6.9**: Implement a complete pipeline for image segmentation.
10. **Experiment 6.10**: Document and present the results of image segmentation experiments.

Module 7: Generative Models (16 hours)

- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)

Experiments:

1. **Experiment 7.1**: Implement a simple GAN for image generation.
2. **Experiment 7.2**: Train a GAN on a small image dataset and evaluate its performance.
3. **Experiment 7.3**: Implement a VAE for image generation.
4. **Experiment 7.4**: Train and evaluate a VAE on a custom dataset.
5. **Experiment 7.5**: Compare the performance of GANs and VAEs for image generation.
6. **Experiment 7.6**: Experiment with different architectures and loss functions for GANs and VAEs.
7. **Experiment 7.7**: Implement a conditional GAN (cGAN) for image generation.
8. **Experiment 7.8**: Train a cGAN on a labeled dataset and evaluate its performance.
9. **Experiment 7.9**: Experiment with different training techniques for GANs.
10. **Experiment 7.10**: Document and present the results of generative model experiments.

Textbooks and References:

- "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
- "Deep Learning with Python" by François Chollet

Data Analytics using Tableau (112)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUML1023	**Data Analytics using Tableau**	**04**	**0+2+2**

Course Description: This course focuses on the use of Tableau for data visualization and business analytics. Students will learn how to create interactive and informative dashboards to effectively communicate insights from data.

Course Objectives:

1. Master the use of Tableau for data visualization.
2. Design and create interactive dashboards.
3. Analyze and interpret data to derive meaningful insights.

Course Outcomes (COs):

1. **CO1**: Explain the fundamental concepts and principles of data visualization using Tableau. (Understand, Remember)
2. **CO2**: Connect to various data sources and prepare data for analysis. (Apply, Analyze)
3. **CO3**: Create basic and advanced visualizations to represent data effectively. (Apply, Create)
4. **CO4**: Design interactive dashboards and use storytelling techniques to communicate insights. (Apply, Create)
5. **CO5**: Share and publish Tableau visualizations and dashboards for collaboration. (Apply, Evaluate)

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	1	2	-	-	1	1	-	-	1	3	2	3
CO2	3	3	3	2	3	-	1	2	1	1	-	2	3	3	3
CO3	3	3	3	3	3	1	1	2	2	1	2	2	3	3	3
CO4	3	3	3	3											

CO-PO-PSO Mapping

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

Syllabus:

Module 1: Introduction to Tableau (16 hours)

- Overview of Tableau
- Connecting to Data Sources

Experiments:

1. **Experiment 1.1**: Install and set up Tableau.
2. **Experiment 1.2**: Connect Tableau to various data sources (Excel, SQL, etc.).
3. **Experiment 1.3**: Navigate the Tableau interface and understand its features.
4. **Experiment 1.4**: Load and clean data in Tableau.
5. **Experiment 1.5**: Create a simple data connection and explore data.
6. **Experiment 1.6**: Perform basic data transformations.
7. **Experiment 1.7**: Use Tableau Prep for data preparation.
8. **Experiment 1.8**: Combine data from multiple sources.

9. **Experiment 1.9**: Create a data extract and use it in Tableau.
10. **Experiment 1.10**: Document and present data connection and preparation steps.

Module 2: Data Visualization Principles (16 hours)

- Best Practices in Data Visualization
- Types of Visualizations

Experiments:

1. **Experiment 2.1**: Study and apply principles of effective data visualization.
2. **Experiment 2.2**: Compare different types of visualizations for specific data types.
3. **Experiment 2.3**: Create bar charts to visualize categorical data.
4. **Experiment 2.4**: Create line charts to show trends over time.
5. **Experiment 2.5**: Create pie charts to represent parts of a whole.
6. **Experiment 2.6**: Create scatter plots to show relationships between variables.
7. **Experiment 2.7**: Design and use maps for geographical data visualization.
8. **Experiment 2.8**: Apply color theory in data visualization.
9. **Experiment 2.9**: Use annotations and tooltips effectively.
10. **Experiment 2.10**: Document and present best practices in data visualization.

Module 3: Building Basic Visualizations (16 hours)

- Bar Charts, Line Charts, Pie Charts
- Scatter Plots, Maps

Experiments:

1. **Experiment 3.1**: Create and customize bar charts.
2. **Experiment 3.2**: Create and customize line charts.
3. **Experiment 3.3**: Create and customize pie charts.
4. **Experiment 3.4**: Create and customize scatter plots.
5. **Experiment 3.5**: Create and customize maps for spatial data.
6. **Experiment 3.6**: Combine multiple basic visualizations in a single worksheet.
7. **Experiment 3.7**: Use filters and highlight actions in basic visualizations.
8. **Experiment 3.8**: Add reference lines and bands to visualizations.
9. **Experiment 3.9**: Create dual-axis charts.
10. **Experiment 3.10**: Document and present basic visualizations.

Module 4: Advanced Visualizations (16 hours)

- Heat Maps, Tree Maps, Bullet Charts
- Histograms, Box Plots

Experiments:

1. **Experiment 4.1**: Create and customize heat maps.
2. **Experiment 4.2**: Create and customize tree maps.
3. **Experiment 4.3**: Create and customize bullet charts.
4. **Experiment 4.4**: Create and customize histograms.
5. **Experiment 4.5**: Create and customize box plots.
6. **Experiment 4.6**: Use advanced chart types to uncover insights.
7. **Experiment 4.7**: Combine advanced visualizations in a single dashboard.
8. **Experiment 4.8**: Use advanced calculations in visualizations.
9. **Experiment 4.9**: Implement advanced interactivity in visualizations.
10. **Experiment 4.10**: Document and present advanced visualizations.

Module 5: Calculations in Tableau (16 hours)

- Calculated Fields
- Table Calculations
- Level of Detail (LOD) Expressions

Experiments:

1. **Experiment 5.1**: Create and use calculated fields.
2. **Experiment 5.2**: Implement basic table calculations.
3. **Experiment 5.3**: Use running totals and moving averages.
4. **Experiment 5.4**: Apply quick table calculations.
5. **Experiment 5.5**: Create and use LOD expressions.
6. **Experiment 5.6**: Perform cohort analysis using LOD expressions.
7. **Experiment 5.7**: Use advanced table calculations.
8. **Experiment 5.8**: Combine calculated fields and table calculations.
9. **Experiment 5.9**: Use parameter controls with calculations.
10. **Experiment 5.10**: Document and present calculations in Tableau.

Module 6: Dashboard Design (16 hours)

- Creating Dashboards
- Dashboard Interactivity
- Storytelling with Data

Experiments:

1. **Experiment 6.1**: Create a basic dashboard.
2. **Experiment 6.2**: Add interactivity to dashboards using actions.
3. **Experiment 6.3**: Use filters and parameters in dashboards.
4. **Experiment 6.4**: Design effective dashboard layouts.
5. **Experiment 6.5**: Use containers for better dashboard organization.
6. **Experiment 6.6**: Create a storytelling narrative with dashboards.
7. **Experiment 6.7**: Implement drill-down functionality.

8. **Experiment 6.8**: Use dashboard extensions for enhanced features.
9. **Experiment 6.9**: Optimize dashboard performance.
10. **Experiment 6.10**: Document and present dashboard design and interactivity.

Module 7: Data Analysis and Reporting (16 hours)

- Parameters and Filters
- Trend Analysis, Forecasting
- Cohort Analysis

Experiments:

1. **Experiment 7.1**: Create and use parameters.
2. **Experiment 7.2**: Implement filters in visualizations and dashboards.
3. **Experiment 7.3**: Perform trend analysis using line charts.
4. **Experiment 7.4**: Implement forecasting in Tableau.
5. **Experiment 7.5**: Analyze seasonal trends and patterns.
6. **Experiment 7.6**: Conduct cohort analysis and visualize results.
7. **Experiment 7.7**: Combine multiple analysis techniques in a single report.
8. **Experiment 7.8**: Create custom reports using Tableau.
9. **Experiment 7.9**: Use statistical analysis tools in Tableau.
10. **Experiment 7.10**: Document and present data analysis and reporting techniques.

Textbooks and References:

- "Learning Tableau" by Joshua N. Milligan
- "Tableau Your Data!" by Daniel G. Murray

Project (168 hrs)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUML1025	**Project**	**06**	**0+0+6**

Course Description: The major project is a capstone course that allows students to apply the knowledge and skills they have acquired throughout the program to a comprehensive project. This project involves identifying a research question, conducting a thorough analysis, and presenting the findings in both written and oral formats.

Course Objectives:

1. Develop advanced research and analytical skills.

2. Apply theoretical knowledge to real-world problems.
3. Demonstrate proficiency in project planning, execution, and presentation.

Course Outcomes (COs):

1. CO1: Summarize and integrate knowledge from various sources to define a research question. (Knowledge)
2. CO2: Apply appropriate research methods and analytical tools to collect and analyze data. (Application)
3. CO3: Evaluate and interpret research findings to draw meaningful conclusions. (Analysis)
4. CO4: Develop a comprehensive project report that demonstrates innovation and critical thinking. (Synthesis)
5. CO5: Present project outcomes effectively through written reports and oral presentations. (Evaluation)

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**

Module-wise Breakdown:

Module 1: Project Proposal (6 hours)

- **Topics:** Identifying a Research Topic; Defining Objectives and Scope; Project Planning and Timeline.

Module 2: Literature Review and Methodology (8 hours)

- **Topics:** Comprehensive Literature Survey; Selecting Appropriate Research Methods.

Module 3: Data Collection and Analysis (10 hours)

- **Topics:** Data Collection Techniques; Data Cleaning and Preprocessing; Analytical Methods and Tools.

Module 4: Implementation (10 hours)

- **Topics:** Developing Models/Systems; Experimentation and Testing.

Module 5: Results and Discussion (8 hours)

- **Topics:** Analyzing Results; Discussing Findings and Implications

Module 6: Report Writing (8 hours)

- **Topics:** Structuring the Final Report; Writing and Revising.

Module 7: Presentation (10 hours)

- **Topics:** Preparing for the Oral Presentation; Presenting Findings to an Audience.

Textbooks and References:

- "Writing for Computer Science" by Justin Zobel
- "How to Write a Thesis" by Umberto Eco

Domain Track: Cloud Technology (0+6+6 Credits)

Course Overview

This 12-credit program is designed to provide comprehensive knowledge and practical skills in AWS Cloud Technology. The program is divided into six subjects, each carrying 3 credits, covering the fundamentals, architecture, services, security, DevOps practices, specialized services, and applications of AWS cloud solutions.

Advanced Cloud Architecture and Design (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCT1031	Advanced Cloud Architecture and Design	03	0+2+1

Course Description: This course delves into the architecture and design of AWS solutions, focusing on best practices, advanced services, and architectural patterns.

Course Objectives:

- Design scalable and resilient architectures on AWS.
- Implement best practices for high availability and fault tolerance.
- Explore advanced AWS services and architectural patterns.

Course Outcomes (COs):

- CO1: Understand advanced cloud computing principles and architecture
- CO2: Design scalable and resilient cloud-based systems
- CO3: Optimize cloud infrastructure for performance and cost-efficiency
- CO4: Implement security measures in cloud environments
- CO5: Leverage automation and DevOps practices in cloud environments

1. **CO-PO-PSO Mapping**

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
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**

AWS Well-Architected Framework (12 hours)

- **Topics:** Overview of the Well-Architected Framework; Pillars of AWS architecture (operational excellence, security, reliability, performance efficiency, cost optimization); Applying the framework to design resilient systems.

Experiments:

1. **Experiment 1.1:** Evaluate a given architecture using the Well-Architected Framework checklist.
2. **Experiment 1.2:** Design a scalable solution focusing on operational excellence principles.
3. **Experiment 1.3:** Implement security best practices in an AWS architecture.
4. **Experiment 1.4:** Perform cost optimization using AWS Cost Explorer.
5. **Experiment 1.5:** Design for performance efficiency using AWS CloudFront.
6. **Experiment 1.6:** Implement reliability principles with AWS Auto Scaling.
7. **Experiment 1.7:** Implement logging and monitoring using AWS CloudWatch.
8. **Experiment 1.8:** Implement fault tolerance with AWS Multi-AZ deployments.
9. **Experiment 1.9:** Secure data at rest and in transit using AWS KMS.
10. **Experiment 1.10:** Implement disaster recovery using AWS services.

Module 2: High Availability and Fault Tolerance (12 hours)

- **Topics:** Designing for high availability; Implementing fault-tolerant architectures; Using AWS services for disaster recovery.

Experiments:

1. **Experiment 2.1:**Configure AWS Auto Scaling to handle varying load demands.
2. **Experiment 2.2:**Implement multi-region redundancy using AWS Route 53 and S3.
3. **Experiment 2.3:**Create a disaster recovery plan using AWS CloudFormation.
4. **Experiment 2.4:**Design fault-tolerant architectures using AWS Elastic Load Balancing.
5. **Experiment 2.5:**Implement cross-region failover using AWS Global Accelerator.
6. **Experiment 2.6:**Use AWS CloudWatch alarms for automated scaling.
7. **Experiment 2.7:**Implement high availability with AWS RDS Multi-AZ deployments.
8. **Experiment 2.8:**Design multi-tier architectures with AWS CloudFormation stacks.
9. **Experiment 2.9:**Implement fault tolerance with AWS SQS and SNS.
10. **Experiment 2.10:**Configure health checks and automated responses with AWS CloudWatch Events.

Module 3: Advanced Compute Services (12 hours)

- **Topics:** Amazon ECS: Elastic Container Service; Amazon EKS: Elastic Kubernetes Service; AWS Lambda: Serverless computing.

Experiments:

1. **Experiment 3.1:**Deploy Docker containers using Amazon ECS.
2. **Experiment 3.2:**Orchestrate Kubernetes clusters with Amazon EKS.
3. **Experiment 3.3:**Develop serverless applications with AWS Lambda.
4. **Experiment 3.4:**Implement batch processing with AWS Batch.
5. **Experiment 3.5:**Configure high-performance computing with AWS ParallelCluster.
6. **Experiment 3.6:**Optimize containerized applications with AWS Fargate.
7. **Experiment 3.7:**Implement serverless APIs using AWS API Gateway.
8. **Experiment 3.8:**Design microservices architecture with AWS Lambda and API Gateway.
9. **Experiment 3.9:**Implement CI/CD pipelines with AWS CodePipeline.
10. **Experiment 3.10:**Monitor and scale applications with AWS App Runner.

Module 4: Advanced Storage and Database Services (12 hours)

- **Topics:** Amazon DynamoDB: NoSQL database; Amazon Redshift: Data warehousing; Amazon Aurora: High-performance relational database.

Experiments:

1. **Experiment 4.1:**Design a schema for high-performance querying using Amazon DynamoDB.
2. **Experiment 4.2:**Optimize data warehousing with Amazon Redshift.
3. **Experiment 4.3:**Implement multi-master replication with Amazon Aurora.
4. **Experiment 4.4:**Configure data lakes with Amazon S3 and AWS Glue.
5. **Experiment 4.5:**Implement real-time analytics with Amazon Kinesis.
6. **Experiment 4.6:**Secure data storage with AWS Encryption SDK.
7. **Experiment 4.7:**Implement versioning and lifecycle policies with Amazon S3.
8. **Experiment 4.8:**Use Amazon DocumentDB for MongoDB compatibility.
9. **Experiment 4.9:**Implement disaster recovery with AWS Backup.
10. **Experiment 4.10:**Optimize cost and performance with Amazon EBS and EFS.

Module 5: Application Integration and Messaging (12 hours)

- **Topics:** Amazon SQS: Simple Queue Service; Amazon SNS: Simple Notification Service; AWS Step Functions: Coordinating microservices.

Experiments:

1. **Experiment 5.1:**Coordinate microservices using AWS Step Functions.

2. **Experiment 5.2:**Integrate applications with Amazon SQS for message queuing.
3. **Experiment 5.3:**Implement event-driven architectures using Amazon SNS.
4. **Experiment 5.4:**Orchestrate workflows with AWS Data Pipeline.
5. **Experiment 5.5:**Design event-driven data processing with AWS Lambda.
6. **Experiment 5.6:**Implement pub/sub messaging with AWS IoT Core.
7. **Experiment 5.7:**Configure message filtering with Amazon SNS topics.
8. **Experiment 5.8:**Implement cross-account message delivery with Amazon SNS.
9. **Experiment 5.9:**Monitor and troubleshoot message queues with Amazon SQS.
10. **Experiment 5.10:**Implement message encryption and access controls with AWS KMS.

Module 6: Security and Compliance (12 hours)

- **Topics:** Advanced security services (AWS KMS, AWS Shield, AWS WAF); Compliance and data protection; Best practices for securing AWS environments.

Experiments:

1. **Experiment 6.1:**Configure encryption at rest and in transit using AWS KMS.
2. **Experiment 6.2:**Implement DDoS protection using AWS Shield and AWS WAF.
3. **Experiment 6.3:**Conduct a compliance audit of an AWS environment.
4. **Experiment 6.4:**Implement IAM policies for least privilege access.
5. **Experiment 6.5:**Configure VPC peering and security groups.
6. **Experiment 6.6:**Implement network ACLs for traffic control.
7. **Experiment 6.7:**Monitor and analyze security logs with AWS CloudTrail.
8. **Experiment 6.8:**Implement data protection with AWS Secrets Manager.
9. **Experiment 6.9:**Configure AWS Config for compliance monitoring.
10. **Experiment 6.10:**Implement secure application deployments with AWS CodeDeploy.

Module 7: Cost Management and Optimization (12 hours)

- **Topics:** AWS pricing models and billing; Cost management tools (AWS Budgets, Cost Explorer); Strategies for cost optimization.

Experiments:

1. **Experiment 7.1:**Analyze AWS billing and cost using Cost Explorer.
2. **Experiment 7.2:**Implement budget alerts and forecasts with AWS Budgets.
3. **Experiment 7.3:**Optimize costs by leveraging Reserved Instances and Spot Instances.
4. **Experiment 7.4:**Implement tagging strategies for cost allocation.
5. **Experiment 7.5:**Use AWS Trusted Advisor for cost optimization recommendations.
6. **Experiment 7.6:**Implement AWS Cost Anomaly Detection for cost monitoring.
7. **Experiment 7.7:**Design cost-effective architectures with AWS Cost Explorer.
8. **Experiment 7.8:**Implement AWS Cost Allocation tags for resource categorization.
9. **Experiment 7.9:**Monitor cost and usage with AWS Cost and Usage Reports.
10. **Experiment 7.10:**Analyze cost breakdowns with AWS Cost Management tools.

Cloud Development and DevOps (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCT1032	Cloud Development and DevOps	03	0+2+1

Course Description: This course focuses on developing applications on AWS and implementing DevOps practices using AWS services and tools.

Course Objectives:

1. Develop and deploy applications using AWS services.
2. Implement DevOps practices for CI/CD on AWS.
3. Gain proficiency in using AWS developer tools.

Course Outcomes (COs):

1. CO1: Utilize AWS development tools to create, manage, and deploy applications. (Apply, Create)
2. CO2: Implement CI/CD pipelines using AWS services for automated deployment. (Apply, Create)
3. CO3: Apply Infrastructure as Code (IaC) principles using AWS CloudFormation and AWS CDK. (Apply, Create)
4. CO4: Monitor and log applications using AWS CloudWatch and AWS X-Ray. (Analyze, Evaluate)
5. CO5: Develop containerized and serverless applications on AWS. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	3	2	3	-	-	-	-	2	3	3	3	3	3
CO2	3	2	3	2	3	-	-	-	-	2	3	3	3	3	3
CO3	3	2	3	2	3	-	-	-	-	2	3	3	3	3	3
CO4	3	2	3	2	3	-	-	-	-	2	3	3	3	3	3
CO5	3	2	3	2	3	-	-	-	-	2	3	3	3	3	3

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

Module 1: AWS Development Tools (12 hours)

- **Topics:** AWS SDKs and CLI; AWS Cloud9: Integrated development environment; AWS CodeCommit: Source control service.

Experiments:

1. **Experiment 1.1:**Set up AWS CLI and configure access to AWS services.
2. **Experiment 1.2:**Create and manage repositories using AWS CodeCommit.
3. **Experiment 1.3:**Use AWS Cloud9 to develop a sample application.
4. **Experiment 1.4:**Integrate AWS SDKs into a development environment.
5. **Experiment 1.5:**Version control and collaboration with AWS CodeCommit.
6. **Experiment 1.6:**Automate deployments using AWS CLI.
7. **Experiment 1.7:**Debug applications using AWS Cloud9.
8. **Experiment 1.8:**Secure repositories with AWS CodeCommit and IAM policies.
9. **Experiment 1.9:**Implement CI/CD with AWS CodeCommit triggers.
10. **Experiment 1.10:**Document and present the setup and use of AWS development tools.

Module 2: Continuous Integration and Continuous Delivery (CI/CD) (12 hours)

- **Topics:** Overview of CI/CD pipelines; AWS CodePipeline: Continuous integration and delivery; AWS CodeBuild and CodeDeploy: Build and deployment automation.

Experiments:

1. **Experiment 2.1:**Create a CI/CD pipeline using AWS CodePipeline.
2. **Experiment 2.2:**Configure AWS CodeBuild for automated builds.
3. **Experiment 2.3:**Implement deployment automation with AWS CodeDeploy.
4. **Experiment 2.2:**Integrate AWS CodePipeline with GitHub for source control.
5. **Experiment 2.5:**Set up automated testing in a CI/CD pipeline.
6. **Experiment 2.6:**Implement blue-green deployments using AWS CodeDeploy.
7. **Experiment 2.7:**Monitor pipeline performance with AWS CloudWatch.
8. **Experiment 2.8:**Create custom actions in AWS CodePipeline.
9. **Experiment 2.9:**Implement approval workflows in a CI/CD pipeline.
10. **Experiment 2.10:**Document and present the CI/CD pipeline setup.

Module 3: Infrastructure as Code (IaC) (12 hours)

- **Topics:** AWS CloudFormation: Infrastructure as code; AWS CDK: Cloud Development Kit; Best practices for IaC.

Experiments:

1. **Experiment 3.1:**Create a CloudFormation template to deploy EC2 instances.

2. **Experiment 3.2:**Use AWS CDK to define infrastructure as code.
3. **Experiment 3.3:**Implement parameterized templates in CloudFormation.
4. **Experiment 3.4:**Manage infrastructure updates with CloudFormation stacks.
5. **Experiment 3.5:**Automate resource provisioning using AWS CDK.
6. **Experiment 3.6:**Implement cross-stack references in CloudFormation.
7. **Experiment 3.7:**Use CloudFormation to deploy a multi-tier application.
8. **Experiment 3.8:**Version control infrastructure as code templates.
9. **Experiment 3.9:**Implement security best practices in CloudFormation.
10. **Experiment 3.10:**Document and present an IaC implementation.

Module 4: Monitoring and Logging (12 hours)

- **Topics:** AWS X-Ray: Distributed tracing; Amazon CloudWatch Logs and Metrics; Centralized logging solutions.

Experiments:

1. **Experiment 4.1:**Set up AWS X-Ray for distributed tracing.
2. **Experiment 4.2:**Create CloudWatch Alarms for proactive monitoring.
3. **Experiment 4.3:**Implement centralized logging with CloudWatch Logs.
4. **Experiment 4.4:**Visualize logs and metrics using CloudWatch Dashboards.
5. **Experiment 4.5:**Analyze application logs using CloudWatch Insights.
6. **Experiment 4.6:**Implement custom metrics in CloudWatch.
7. **Experiment 4.7:**Use AWS X-Ray to trace application requests.
8. **Experiment 4.8:**Set up log retention policies in CloudWatch Logs.
9. **Experiment 4.9:**Document and present a monitoring and logging solution.

Module 5: Containerization and Orchestration (12 hours)

- **Topics:** Docker on AWS; Amazon ECS and EKS for container orchestration; Best practices for containerized applications.

Experiments:

1. **Experiment 5.1:**Dockerize a sample application.
2. **Experiment 5.2:**Deploy Docker containers using Amazon ECS.
3. **Experiment 5.3:**Set up an Amazon EKS cluster for Kubernetes orchestration.
4. **Experiment 5.4:**Implement container networking in Amazon ECS.
5. **Experiment 5.5:**Manage containerized applications with AWS Fargate.
6. **Experiment 5.6:**Monitor container performance using CloudWatch Container Insights.
7. **Experiment 5.7:**Implement auto-scaling for containerized applications.
8. **Experiment 5.8:**Secure containerized applications using IAM roles.
9. **Experiment 5.9:**Deploy a microservices architecture using ECS and EKS.
10. **Experiment 5.10:**Document and present a containerized application deployment.

Module 6: Serverless Development (12 hours)

- **Topics:** AWS Lambda: Function as a Service; Building serverless applications with Lambda; Event-driven architectures.

Experiments:

1. **Experment 6.1:**Create and deploy an AWS Lambda function.
2. **Experment 6.2:**Build a serverless API using AWS Lambda and API Gateway.
3. **Experiment 6.3:**Implement event-driven processing with AWS Lambda and S3.
4. **Experiment 6.4:**Integrate AWS Lambda with DynamoDB for serverless data processing.
5. **Experiment 6.5:**Monitor serverless applications using CloudWatch Logs.
6. **Experiment 6.6:**Implement retries and error handling in AWS Lambda.
7. **Experiment 6.7:**Use AWS SAM to manage serverless applications.
8. **Experiment 6.8:**Secure serverless applications with IAM policies.
9. **Experiment 6.9:**Optimize AWS Lambda performance and cost.
10. **Experiment 6.10:**Document and present a serverless application.

Module 7: Hands-on Lab and Project (12 hours)

- **Topics:** Build and deploy a CI/CD pipeline using AWS CodePipeline; Develop a serverless application with AWS Lambda; Hands-on lab exercises.

Experiments:

1. **Experiment 7.1:**Design and implement a CI/CD pipeline for a sample application.
2. **Experiment 7.2:**Develop and deploy a serverless application using AWS Lambda.
3. **Experiment 7.3:**Integrate monitoring and logging into the CI/CD pipeline.
4. **Experiment 7.4:**Optimize the CI/CD pipeline for performance and reliability.
5. **Experiment 7.5:**Test and validate the CI/CD pipeline with automated testing.
6. **Experiment 7.6:**Deploy containerized applications as part of the CI/CD pipeline.
7. **Experiment 7.7:**Document the CI/CD pipeline setup and configuration.
8. **Experiment 7.8:**Present the CI/CD pipeline and serverless application project.
9. **Experiment 7.9:**Receive peer feedback and make improvements.

Cloud Security and Compliance (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCT1033	Cloud Security and Compliance	03	0+2+1

Course Description: This course covers the security and compliance aspects of AWS, focusing on best practices, advanced security services, and compliance frameworks.

Course Objectives:

1. Understand AWS security fundamentals.
2. Learn to implement advanced security measures on AWS.
3. Gain proficiency in ensuring compliance with industry standards.

Course Outcomes (COs):

1. CO1: Explain the shared responsibility model and security best practices on AWS. (Understand, Remember)
2. CO2: Implement identity and access management using AWS IAM. (Apply, Create)
3. CO3: Apply data protection and encryption techniques using AWS services. (Apply, Analyze)
4. CO4: Implement network security best practices using AWS services. (Apply, Create)
5. CO5: Ensure compliance with industry standards using AWS compliance tools. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module 1: Security Fundamentals (12 hours)

- **Topics:** Shared responsibility model; Security best practices; AWS security services overview.

Experiments:

1. **Experiment 1.1:**Study and explain the shared responsibility model on AWS.
2. **Experiment 1.2:**Identify and document AWS security best practices.
3. **Experiment 1.3:**Explore and summarize AWS security services.
4. **Experiment 1.4:**Implement basic security settings in an AWS account.
5. **Experiment 1.5:**Evaluate the security posture of an AWS environment.
6. **Experiment 1.6:**Set up a secure AWS environment using best practices.
7. **Experiment 1.7:**Conduct a security audit of an AWS environment.
8. **Experiment 1.8:**Implement AWS Identity and Access Management (IAM) for users.
9. **Experiment 1.9.:**Use AWS Trusted Advisor for security recommendations.
10. **Experiment 1.10:**Present findings on AWS security fundamentals.

Module 2: Identity and Access Management (12 hours)

- **Topics:** AWS IAM deep dive; Managing users, groups, and roles; Implementing multi-factor authentication (MFA).

Experiments:

1. **Experiment 2.1:**Set up AWS IAM users, groups, and roles.
2. **Experiment 2.2:**Implement and manage IAM policies.
3. **Experiment 2.3:**Enable and configure MFA for AWS accounts.
4. **Experiment 2.4:**Audit and review IAM policies for compliance.
5. **Experiment 2.5:**Implement least privilege access using IAM.
6. **Experiment 2.6:**Monitor and log IAM activities using CloudTrail.
7. **Experiment 2.7:**Use IAM roles for cross-account access.
8. **Experiment 2.8:**Implement federated access with IAM.
9. **Experiment 2.9:**Set up AWS Organizations for multi-account management.
10. **Experiment 2.10:**Document and present IAM configurations and best practices.

Module 3: Data Protection and Encryption (12 hours)

- **Topics:** AWS Key Management Service (KMS); AWS CloudHSM: Hardware security module; Data encryption at rest and in transit.

Experiments:

1. **Experiment 3.1:**Set up and manage AWS KMS keys.
2. **Experiment 3.2:**Encrpt data at rest using AWS KMS.
3. **Experiment 3.3:**Encrpt data in transit using SSL/TLS.
4. **Experiment 3.4:**Configure and use AWS CloudHSM.

5. **Experiment 3.5:**Implement S3 bucket encryption using KMS keys.
6. **Experiment 3.6:**Use KMS with AWS RDS for database encryption.
7. **Experiment 3.7:**Implement EBS volume encryption with KMS.
8. **Experiment 3.8:**Set up and manage customer managed keys (CMKs).
9. **Experiment 3.9:**Implement server-side encryption with S3.
10. **Experiment 3.10:**Document and present data protection strategies.

Module 4: Network Security (12 hours)

- **Topics:** VPC security best practices; Security groups and network ACLs; AWS WAF and Shield: Web application firewall and DDoS protection.

Experiments:

1. **Experiment 4.1:**Configure VPC security groups and network ACLs.
2. **Experiment 4.2:**Implement best practices for securing VPCs.
3. **Experiment 4.3:**Set up AWS WAF for a web application.
4. **Experiment 4.4:**Configure AWS Shield for DDoS protection.
5. **Experiment 4.5:**Monitor VPC traffic using VPC Flow Logs.
6. **Experiment 4.6:**Implement private sunets with NAT gateways.
7. **Experiment 4.7:**Secure VPC endpoints for S3 and DynamoDB.
8. **Experiment 4.8:**Use AWS Network Firewall for network traffic control.
9. **Experiment 4.9:**Implement a bastion host for secure access.
10. **Experiment 4.10:**Document and present network security configurations.

Module 5: Monitoring and Logging for Security (12 hours)

- **Topics:** AWS CloudTrail: Security auditing; Amazon GuardDuty: Threat detection; AWS Security Hub: Centralized security view.

Experiments:

1. **Experiment 5.1:**Set up and configure AWS CloudTrail.
2. **Experiment 5.2:**Use CloudTrail to monitor API activity
3. **Experiment 5.3:**Configure Amazon GuardDuty for threat detection.
4. **Experiment 5.4:**Analyze GuardDuty findings and alerts.
5. **Experiment 5.5:**Set up AWS Security Hub for centralized security management.
6. **Experiment 5.6:**Integrate Security Hub with other AWS services.
7. **Experiment 5.7:**Use CloudWatch to monitor security metrics.
8. **Experiment 5.8:**Implement AWS Config for security compliance.
9. **Experiment 5.9:**Set up AWS Inspector for vulnerability assessments.
10. **Experiment 5.10:**Document and present security monitoring configurations.

Module 6: Compliance and Governance (12 hours)

- **Topics:** AWS compliance programs and certifications; Implementing governance frameworks (CIS, NIST, ISO); Using AWS Config for continuous compliance.

Experiments:

1. **Experiment 6.1:**Study and summarize AWS compliance programs.
2. **Experiment 6.2:**Implement AWS Config rules for continuous compliance.
3. **Experiment 6.3:**Configure AWS CloudFormation for compliance automation.
4. **Experiment 6.4:**Set up AWS Control Tower for governance.
5. **Experiment 6.5:**Use AWS Artifact to access compliance reports.
6. **Experiment 6.6:**Implement a governance framework using AWS services.
7. **Experiment 6.7:**Audit AWS environments for compliance with CIS benchmarks.
8. **Experiment 6.8:**Use AWS Service Catalog for compliant resource provisioning.
9. **Experiment 6.9:**Conduct a gap analysis against NIST standards.
10. **Experiment 6.10:**Document and present compliance strategies.

Module 7: Incident Response and Forensics (12 hours)

- **Topics:** AWS best practices for incident response; Setting up an incident response plan; Conducting forensics on AWS.

Experiments:

1. **Experiment 7.1:**Develop an incident response plan for AWS environments.
2. **Experiment 7.2:**Set up AWS IAM roles for incident response.
3. **Experiment 7.3:**Implement automated incident response with AWS Lambda.
4. **Experiment 7.4:**Use AWS CloudFormation to deploy incident response resources.
5. **Experiment 7.5:**Conduct a simulated security incident and response.
6. **Experiment 7.6:**Collect and analyze forensic data using AWS services.
7. **Experiment 7.7:**Use AWS Config and CloudTrail for forensic investigations.
8. **Experiment 7.8:**Implement AWS Systems Manager for incident response automation.
9. **Experiment 7.9:**Create a runbook for security incident response.
10. **Experiment 7.10:**Document and present incident response and forensic strategies.

.Capstone Project (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCT1034	Capstone Project	03	0+0+3

Course Description: This capstone course allows students to apply their AWS knowledge and skills to solve a real-world problem or develop a comprehensive solution using multiple AWS services.

Course Objectives:

1. Develop a comprehensive AWS-based solution.
2. Apply best practices and advanced AWS services.
3. Demonstrate proficiency in AWS architecture, development, and security.

Course Outcomes (COs):

- CO1: Identify and scope a real-world problem or project idea suitable for an AWS-based solution. (Understand, Remember)
- CO2: Develop a comprehensive project plan and timeline, defining clear objectives and deliverables. (Apply, Create)
- CO3: Implement the project using AWS services, adhering to best practices for architecture, security, and DevOps. (Apply, Create)
- CO4: Perform thorough testing and validation of the AWS-based solution. (Analyze, Evaluate)
- CO5: Present and demonstrate the final project, showcasing the solution's effectiveness and alignment with the defined objectives. (Evaluate, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module 1: Project Proposal and Planning (12 hours)

- **Topics:** Identify a real-world problem or project idea; Define project scope, objectives, and deliverables; Develop a project plan and timeline.

Module 2: Project Development (12 hours)

- **Topics:** Implement the project using AWS services; Apply best practices for architecture, security, and DevOps; Regular progress reviews and feedback.

Module 3: Project Testing and Validation (12 hours)

- **Topics:** Perform testing and validation.

Module 4: Project Documentation (12 hours)

- **Topics:** Prepare project documentation.

Module 5: Project Presentation and Demonstration (12 hours)

- **Topics:** Final presentation and demonstration of the project.

Evaluation and Grading:

- Assignments and Quizzes: 20%
- Projects (excluding Capstone): 30%
- Capstone Project: 40%
- Participation and Attendance: 10%

Recommended Textbooks and Resources:

- **AWS Cloud:**
  - "AWS Certified Solutions Architect Official Study Guide" by Joe Baron, Hisham Baz, and Tim Bixler.
  - "Amazon Web Services in Action" by Andreas Wittig and Michael Wittig.
- **Cloud Computing:**
  - "Architecting the Cloud" by Michael J. Kavis.
  - "Cloud Native Transformation" by Pini Reznik, Jamie Dobson, and Michelle Gienow.
- **DevOps:**
  - "The DevOps Handbook" by Gene Kim, Patrick Debois, and John Willis.
  - "Infrastructure as Code" by Kief Morris.
- **Additional Resources:**
  - AWS online documentation and whitepapers.
  - AWS training and certification courses.
  - GitHub repositories for sample projects and code.
  - Community forums and study groups.

Domain Track: Drone Imaging and Spectral Analysis (0+6+6 Credits)

Drone Image Processing using Pix4D(84hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUDS1041	**Drone Image Processing using Pix4D**	**03**	**0+2+1**

Course Description: This course provides an in-depth understanding of drone image processing using Pix4D. Students will learn how to process aerial images to generate various outputs like orthomosaics, 3D models, and digital surface models (DSMs).

Course Objectives:

1. Understand the fundamentals of drone imaging and Pix4D software.
2. Learn to plan drone missions and acquire high-quality data.
3. Develop proficiency in processing drone images to generate various outputs.

Course Outcomes (COs):

1. CO1: Explain the fundamentals of drone imaging and Pix4D software. (Understand, Remember)
2. CO2: Plan and execute drone missions for data acquisition. (Apply, Create)
3. CO3: Process drone images using Pix4D to generate point clouds and align images. (Apply, Analyze)
4. CO4: Create orthomosaics, 3D models, and DSMs using Pix4D. (Apply, Create)
5. CO5: Implement advanced processing techniques including multi-spectral and thermal image processing. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Drone Imaging (6 hours)

- **Topics:** Overview of Drone Technology; Applications of Drone Imaging.

Experiments:

1. **Experiment 1.1:**Study and summarize drone technology and its components.
2. **Experiment 1.2:**Explore and document various applications of drone imaging.
3. **Experiment 1.3:**Research and present the evolution of drone technology.
4. **Experiment 1.4:**Compare different types of drones used in imaging.
5. **Experiment 1.5:**Understand the regulatory aspects of drone usage.
6. **Experiment 1.6:**Explore the role of drones in different industries.
7. **Experiment 1.7:**Analyze case studies of successful drone imaging projects.
8. **Experiment 1.8:**Identify challenges and solutions in drone imaging.
9. **Experiment 1.9:**Conduct a seminar on future trends in drone technology.
10. **Experiment 1.10:**Present findings on the applications of drone imaging.

Module 2: Introduction to Pix4D (6 hours)

- **Topics:** Overview of Pix4D Software; Installing and Setting up Pix4D.

Experiments:

1. **Experiment 2.1:**Install Pix4D software on a computer.
2. **Experiment 2.2:**Set up a Pix4D project workspace.
3. **Experiment 2.3:**Navigate the Pix4D interface and explore its features.
4. **Experiment 2.4:**Import sample data into Pix4D.
5. **Experiment 2.5:**Explore the various modules and tools in Pix4D.
6. **Experiment 2.6:**Set up project parameters in Pix4D.
7. **Experiment 2.7:**Troubleshoot common installation issues.
8. **Experiment 2.8:**Understand the licensing and subscription options for Pix4D.
9. **Experiment 2.9:**Conduct a hands-on session on Pix4D setup.
10. **Experiment 2.10:**Document and present the Pix4D installation and setup process.

Module 3: Flight Planning and Data Acquisition (6 hours)

- **Topics:** Mission Planning; Best Practices for Data Collection; Data Quality Considerations.

Experiments:

1. **Experiment 3.1:**Plan a drone mission using Pix4D.
2. **Experiment 3.2:**Set up flight parameters for data acquisition.
3. **Experiment 3.3:**Execute a drone flight mission and collect data.
4. **Experiment 3.4:**Evaluate the quality of collected data.
5. **Experiment 3.5:**Implement best practices for drone flight safety.
6. **Experiment 3.6:**Use Pix4D Capture app for mission planning.
7. **Experiment 3.7:**Analyze factors affecting data quality.
8. **Experiment 3.8:**Conduct a field survey for data collection.
9. **Experiment 3.9:**Document flight planning and data acquisition process.
10. **Experiment 3.10:**Present findings on data quality considerations.

Module 4: Image Processing in Pix4D (6 hours)

- **Topics:** Importing Images and Initial Setup; Image Alignment and Calibration; Point Cloud Generation.

Experiments:

1. **Experiment 4.1:**Import drone images into Pix4D.
2. **Experiment 4.2:**Set up initial project parameters for image processing.
3. **Experiment 4.3:**Align images using Pix4D's calibration tools.
4. **Experiment 4.4:**Generate a point cloud from imported images.
5. **Experiment 4.5:**Analyze the accuracy of image alignment.
6. **Experiment 4.6:**Adjust processing parameters to improve results.
7. **Experiment 4.7:**Conduct quality checks on generated point clouds.
8. **Experiment 4.8:**Explore different calibration techniques in Pix4D.
9. **Experiment 4.9:**Document the image processing workflow.
10. **Experiment 4.10:**Present a processed point cloud and its analysis.

Module 5: Creating Outputs (6 hours)

- **Topics:** Generating Orthomosaics; Creating 3D Models and Meshes; Generating DSMs and DTMs.

Experiments:

1. **Experiment 5.1:**Generate an orthomosaic using Pix4D.
2. **Experiment 5.2:**Create a 3D model from drone images.
3. **Experiment 5.3:**Develop meshes for 3D visualization.
4. **Experiment 5.4:**Generate DSMs and DTMs from processed images.
5. **Experiment 5.5:**Analyze the quality of generated outputs.
6. **Experiment 5.6:**Compare different output formats and their uses.
7. **Experiment 5.7:**Conduct a project to create multiple outputs from a dataset.
8. **Experiment 5.8:**Implement techniques to enhance output quality.
9. **Experiment 5.9:**Document the process of creating outputs in Pix4D.

10. **Experiment 5.10:**Present and evaluate generated outputs.

Module 6: Advanced Processing Techniques (6 hours)

- **Topics:** Multi-spectral and Thermal Image Processing; Integrating Ground Control Points (GCPs); Quality Assessment and Improvement.

Experiments:

1. **Experiment 6.1:**Process multi-spectral images using Pix4D.
2. **Experiment 6.2:**Implement thermal image processing techniques.
3. **Experiment 6.3:**Integrate GCPs into a Pix4D project.
4. **Experiment 6.4:**Conduct a quality assessment of processed images.
5. **Experiment 6.5:**Apply techniques to improve image processing results.
6. **Experiment 6.6:**Analyze the benefits of using GCPs.
7. **Experiment 6.7:**Compare multi-spectral and thermal image outputs.
8. **Experiment 6.8:**Conduct a project using advanced processing techniques.
9. **Experiment 6.9:**Document advanced processing workflows.
10. **Experiment 6.10:**Present findings on quality assessment and improvement.

Module 7: Case Studies and Applications (6 hours)

- **Topics:** Real-World Applications and Case Studies; Hands-on Projects with Pix4D.

Experiments:

1. **Experiment 7.1:**Study and present real-world applications of drone imaging.
2. **Experiment 7.2:**Analyze case studies of Pix4D projects.
3. **Experiment 7.3:**Conduct a hands-on project using Pix4D.
4. **Experiment 7.4:**Implement a real-world application using Pix4D.
5. **Experiment 7.5:**Evaluate the success and challenges of case studies.
6. **Experiment 7.6:**Develop a project plan for a real-world application.
7. **Experiment 7.7:**Document the project implementation process.
8. **Experiment 7.8:**Present a case study analysis.
9. **Experiment 7.9:**Conduct a peer review of project implementations.
10. **Experiment 7.10:**Submit and present the final project.

Textbooks and References:

- "Introduction to Drone Imaging" by Douglas Spotted Eagle
- Pix4D Documentation and Tutorials

Course Code	Course Title	Credits	Type (T+P+Pj)
CUDS1042	**Multispectral Image Analytics for Agriculture**	**03**	**0+2+1**

Multispectral Image Analytics for Agriculture(84hours)

Course Description: This course explores the use of drone imaging and multispectral sensors for agricultural applications. Students will learn how to capture and analyze multispectral data to monitor crop health, detect diseases, and optimize agricultural practices.

Course Objectives:

1. Understand the application of drone imaging and multispectral sensors in agriculture.
2. Learn to process and analyze multispectral data for agricultural insights.
3. Apply advanced analytical methods to improve agricultural practices.

Course Outcomes (COs):

1. CO1: Explain the benefits and applications of drones and multispectral imaging in agriculture. (Understand, Remember)
2. CO2: Acquire and calibrate multispectral data using appropriate sensors. (Apply, Create)
3. CO3: Process multispectral data and calculate vegetation indices. (Apply, Analyze)
4. CO4: Analyze vegetation indices to monitor crop health and predict yield. (Apply, Evaluate)
5. CO5: Implement precision agriculture techniques and advanced analytical methods. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Agricultural Drone Imaging (6 hours)

- **Topics:** Overview of Drones in Agriculture; Benefits of Multispectral Imaging.

Experiments:

1. **Experiment 1.1:**Research and 1the role of drones in modern agriculture.
2. **Experiment 1.2:**Explore and document the benefits of using multispectral imaging in agriculture.
3. **Experiment 1.3:**Compare different types of drones used in agricultural imaging.
4. **Experiment 1.4:**Study and present case studies of successful agricultural drone projects.
5. **Experiment 1.5:**Identify the challenges and solutions in agricultural drone imaging.
6. **Experiment 1.6:**Conduct a seminar on the future trends in agricultural drone technology.
7. **Experiment 1.7:**Analyze the economic impact of using drones in agriculture.
8. **Experiment 1.8:**Present findings on the applications of multispectral imaging in agriculture.
9. **Experiment 1.9:**Research the legal and regulatory aspects of using drones in agriculture.
10. **Experiment 1.10:**Document and present the overall benefits of drones and multispectral imaging.

Module 2: Multispectral Sensors and Data Acquisition (6 hours)

- **Topics:** Types of Multispectral Sensors; Calibration and Data Collection.

Experiments:

1. **Experiment 2.1:**Study different types of multispectral sensors used in agriculture.
2. **Experiment 2.2:**Calibrate a multispectral sensor for data collection.
3. **Experiment 2.3:**Set up a drone for multispectral imaging.
4. **Experiment 2.4:**Plan and execute a data acquisition mission using a multispectral sensor.
5. **Experiment 2.5:**Evaluate the quality of collected multispectral data.
6. **Experiment 2.6:**Implement best practices for data collection in agriculture.
7. **Experiment 2.7:**Analyze the factors affecting multispectral data quality.
8. **Experiment 2.8:**Conduct a field survey for multispectral data collection.
9. **Experiment 2.9:**Document the process of sensor calibration and data acquisition.

10. **Experiment 2.10:**Present findings on multispectral sensors and data collection techniques.

Module 3: Image Processing for Agriculture (6 hours)

- **Topics:** Pre-processing Multispectral Data; Reflectance Calculation and Vegetation Indices.

Experiments:

1. **Experiment 3.1:**Pre-process raw multispectral data for analysis.
2. **Experiment 3.2:**Calculate reflectance values from multispectral data.
3. **Experiment 3.3:**Generate NDVI maps using multispectral data.
4. **Experiment 3.4:**Explore different vegetation indices and their calculation.
5. **Experiment 3.5:**Implement data normalization techniques for accurate analysis.
6. **Experiment 3.6:**Conduct quality checks on pre-processed data.
7. **Experiment 3.7:**Use software tools for multispectral image processing.
8. **Experiment 3.8:**Analyze the accuracy of vegetation indices.
9. **Experiment 3.9:**Document the image processing workflow for agriculture.
10. **Experiment 3.10:**Present findings on pre-processing and reflectance calculation.

Module 4: Vegetation Indices and Analysis (6 hours)

- **Topics:** NDVI (Normalized Difference Vegetation Index); Other Indices (EVI, SAVI, etc.); Analyzing Vegetation Indices.

Experiments:

1. **Experiment 4.1:**Calculate NDVI from multispectral data.
2. **Experiment 4.2:**Generate Enhanced Vegetation Index (EVI) maps.
3. **Experiment 4.3:**Calculate Soil-Adjusted Vegetation Index (SAVI).
4. **Experiment 4.4:**Compare different vegetation indices for crop analysis.
5. **Experiment 4.5:**Implement methods for analyzing vegetation indices.
6. **Experiment 4.6:**Conduct a case study on vegetation indices for crop health monitoring.
7. **Experiment 4.7:**Use software tools for vegetation index calculation.
8. **Experiment 4.8:**Analyze the temporal changes in vegetation indices.
9. **Experiment 4.9:**Document the process of calculating and analyzing vegetation indices.
10. **Experiment 4.10:**Present findings on the use of vegetation indices in agriculture.

Module 5: Crop Health Monitoring (6 hours)

- **Topics:** Identifying Stress and Disease; Yield Prediction and Assessment.

Experiments:

1. **Experiment 5.1:**Identify crop stress using multispectral data.

2. **Experiment 5.2:**Detect plant diseases using vegetation indices.
3. **Experiment 5.3:**Implement methods for yield prediction using multispectral data.
4. **Experiment 5.4:**Conduct a case study on crop health monitoring.
5. **Experiment 5.5:**Analyze the effectiveness of multispectral imaging for disease detection.
6. **Experiment 5.6:**Use software tools for crop health assessment.
7. **Experiment 5.7:**Implement advanced algorithms for yield prediction.
8. **Experiment 5.8:**Conduct field validation of crop health monitoring results.
9. **Experiment 5.9:**Document the process of crop health monitoring and yield prediction.
10. **Experiment 5.10:**Present findings on the use of multispectral data for crop health monitoring.

Module 6: Precision Agriculture Techniques (6 hours)

- **Topics:** Variable Rate Application; Field Mapping and Zoning.

Experiments:

1. **Experiment 6.1:**Implement variable rate application techniques using multispectral data.
2. **Experiment 6.2:**Conduct field mapping for precision agriculture.
3. **Experiment 6.3:**Generate zoning maps for targeted interventions.
4. **Experiment 6.4:**Use software tools for precision agriculture mapping.
5. **Experiment 6.5:**Analyze the benefits of variable rate application.
6. **Experiment 6.6:**Conduct a case study on precision agriculture techniques.
7. **Experiment 6.7:**Implement methods for optimizing field inputs.
8. **Experiment 6.8:**Evaluate the economic impact of precision agriculture.
9. **Experiment 6.8:**Document the process of field mapping and zoning.
10. **Experiment 6.10:**Present findings on the use of precision agriculture techniques.

Module 7: Advanced Analytical Methods (6 hours)

- **Topics:** Machine Learning for Crop Analysis; Time-Series Analysis of Crop Data.

Experiments:

1. **Experiment 7.1:**Implement machine learning algorithms for crop analysis.
2. **Experiment 7.2:**Conduct time-series analysis of multispectral crop data.
3. **Experiment 7.3:**Use software tools for machine learning in agriculture.
4. **Experiment 7.4:**Analyze the effectiveness of machine learning for crop health prediction.
5. **Experiment 7.5:**Conduct a case study on advanced analytical methods.
6. **Experiment 7.6:**Implement methods for integrating multispectral data with other datasets.
7. **Experiment 7.7:**Evaluate the benefits of using machine learning in agriculture.

8. **Experiment 7.8:**Document the process of applying advanced analytical methods.
9. **Experiment 7.9:**Present findings on machine learning and time-series analysis for agriculture.
10. **Experiment 7.10:**Submit a project report on advanced analytical methods for crop analysis.

Textbooks and References:

- "Precision Agriculture Technology for Crop Farming" by Qin Zhang
- Research Papers and Case Studies on Agricultural Drone Imaging

Drone Imaging Applications(56hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUDS1043	**Drone Imaging Applications**	**02**	**0+2+0**

Course Description: This course covers various applications of drone imaging across different industries. Students will learn about the diverse uses of drone data in areas such as environmental monitoring, construction, mining, and disaster management.

Course Objectives:

1. Understand the diverse applications of drone imaging across various industries.
2. Learn to implement drone imaging techniques for specific industry needs.
3. Analyze and interpret drone data for practical applications.

Course Outcomes (COs):

1. CO1: Explain the regulatory considerations and best practices for drone applications in different industries. (Understand, Remember)
2. CO2: Implement drone imaging techniques for environmental monitoring. (Apply, Create)
3. CO3: Apply drone imaging for construction and infrastructure projects. (Apply, Analyze)
4. CO4: Utilize drone imaging in mining and resource management. (Apply, Evaluate)
5. CO5: Implement advanced imaging techniques and integrate various data types for comprehensive analysis. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO1 2	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Drone Applications (4 hours)

- **Topics:** Overview of Industry Applications; Regulatory Considerations and Best Practices.

Experiments:

1. **Experiment 1.1:**Research and present the various industry applications of drone imaging.
2. **Experiment 1.2:**Explore and document regulatory considerations for drone usage.
3. **Experiment 1.3:**Study best practices for drone operations in different industries.
4. **Experiment 1.4:**Analyze case studies of successful drone applications.
5. **Experiment 1.5:**Identify challenges and solutions in implementing drone projects.
6. **Experiment 1.6:**Conduct a seminar on future trends in drone applications.
7. **Experiment 1.7:**Explore the economic impact of drone technology in various sectors.
8. **Experiment 1.8:**Present findings on regulatory and best practice guidelines.
9. **Experiment 1.9:**Research the ethical considerations of using drones.
10. **Experiment 1.10:**Document and present an overview of drone applications.

Module 2: Environmental Monitoring (4 hours)

- **Topics:** Habitat Mapping and Conservation; Forest Health and Wildlife Monitoring; Water Quality Assessment.

Experiments:

1. **Experiment 2.1:**Implement drone-based habitat mapping techniqu8es.
2. **Experiment 2.2:**Monitor forest health using drone imaging.
3. **Experiment 2.3:**Assess wildlife populations using drone data.
4. **Experiment 2.4:**Conduct a water quality assessment with drone sensors.
5. **Experiment 2.5:**Analyze the effectiveness of drones in conservation projects.

6. **Experiment 2.6:**Use software tools for environmental data analysis.
7. **Experiment 2.7:**Conduct a case study on drone applications in environmental monitoring.
8. **Experiment 2.8:**Implement best practices for environmental data collection.
9. **Experiment 2.9:**Document the process of environmental monitoring with drones.
10. **Experiment 2.10:**Present findings on the impact of drones in environmental conservation.

Module 3: Construction and Infrastructure (4 hours)

- **Topics:** Site Surveying and Progress Monitoring; Structural Inspections and 3D Modeling; Safety and Compliance Monitoring.

Experiments:

1. **Experiment 3.1:**Conduct a site survey using drone imaging.
2. **Experiment 3.2:**Monitor construction progress with drone data.
3. **Experiment 3.3:**Perform structural inspections using drones.
4. **Experiment 3.4:**Create 3D models of infrastructure projects.
5. **Experiment 3.5:**Implement safety and compliance monitoring techniques.
6. **Experiment 3.6:**Analyze the benefits of drones in construction projects.
7. **Experiment 3.7:**Use software tools for construction data analysis.
8. **Experiment 3.8:**Conduct a case study on drone applications in construction.
9. **Experiment 3.9:**Document the process of using drones in infrastructure projects.
10. **Experiment 3.10:**Present findings on the effectiveness of drones in construction.

Module 4: Mining and Resources (4 hours)

- **Topics:** Site Mapping and Resource Estimation; Stockpile Measurement and Management; Environmental Impact Assessment.

Experiments:

1. **Experiment 4.1:**Map a mining site using drone imaging.
2. **Experiment 4.2:**Estimate resources with drone data.
3. **Experiment 4.3:**Measure and manage stockpiles using drones.
4. **Experiment 4.4:**Conduct an environmental impact assessment with drone sensors.
5. **Experiment 4.5:**Analyze the benefits of drones in resource management.
6. **Experiment 4.6:**Use software tools for mining data analysis.
7. **Experiment 4.7:**Conduct a case study on drone applications in mining.
8. **Experiment 4.8:**Implement best practices for resource estimation with drones.
9. **Experiment 4.9:**Document the process of using drones in mining projects.
10. **Experiment 4.10:**Present findings on the impact of drones in the mining industry.

Module 5: Disaster Management (4 hours)

- **Topics:** Disaster Response and Damage Assessment; Search and Rescue Operations; Post-Disaster Recovery Planning.

Experiments:

1. **Experiment 5.1:**Implement drone-based disaster response techniques.
2. **Experiment 5.2:**Assess damage using drone imaging.
3. **Experiment 5.3:**Conduct search and rescue operations with drones.
4. **Experiment 5.4:**Plan post-disaster recovery using drone data.
5. **Experiment 5.5:**Analyze the effectiveness of drones in disaster management.
6. **Experiment 5.6:**Use software tools for disaster data analysis.
7. **Experiment 5.7:**Conduct a case study on drone applications in disaster response.
8. **Experiment 5.8:**Implement best practices for disaster management with drones.
9. **Experiment 5.9:**Document the process of using drones in disaster scenarios.
10. **Experiment 5.10:**Present findings on the role of drones in disaster management.

Module 6: Urban Planning and Development (4 hours)

- **Topics:** Urban Mapping and Analysis; Smart City Applications; Infrastructure Planning and Monitoring.

Experiments:

1. **Experiment 6.1:**Map urban areas using drone imaging.
2. **Experiment 6.2:**Analyze urban development with drone data.
3. **Experiment 6.3:**Implement smart city applications with drones.
4. **Experiment 6.4:**Monitor urban infrastructure using drones.
5. **Experiment 6.5:**Analyze the benefits of drones in urban planning.
6. **Experiment 6.6:**Use software tools for urban data analysis.
7. **Experiment 6.7:**Conduct a case study on drone applications in urban planning.
8. **Experiment 6.8:**Implement best practices for urban mapping with drones.
9. **Experiment 6.9:**Document the process of using drones in urban development projects.
10. **Experiment 6.10:**Present findings on the impact of drones in urban planning.

Module 7: Advanced Imaging Techniques (4 hours)

- **Topics:** Thermal Imaging Applications; Hyperspectral Imaging; LiDAR and Photogrammetry Integration.

Experiments:

1. **Experiment 7.1:**Implement thermal imaging techniques with drones.

2. **Experiment 7.2:**Conduct hyperspectral imaging with drone sensors.
3. **Experiment 7.3:**Integrate LiDAR and photogrammetry data.
4. **Experiment 7.4:**Analyze the effectiveness of advanced imaging techniques.
5. **Experiment 7.5:**Use software tools for advanced imaging analysis.
6. **Experiment 7.6:**Conduct a case study on advanced drone imaging applications.
7. **Experiment 7.7:**Implement best practices for integrating multiple data types.
8. **Experiment 7.8:**Document the process of using advanced imaging techniques.
9. **Experiment 7.9:**Present findings on the benefits of advanced imaging in various industries.
10. **Experiment 7.10:**Submit a project report on advanced imaging techniques.

Textbooks and References:

- "Drones for Dummies" by Mark LaFay and Chase Guttman
- Industry Reports and Case Studies

Project(224hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUDS1044	**Project**	**04**	**0+0+4**

Course Description: The project course allows students to apply the knowledge and skills they have acquired in drone imaging and spectral analysis to a comprehensive, real-world project. This involves planning, data collection, processing, analysis, and presentation of findings.

Course Objectives:

1. Develop and execute a comprehensive project in drone imaging and spectral analysis.
2. Apply advanced data collection, processing, and analysis techniques.
3. Present findings effectively through written reports and oral presentations.

Course Outcomes (COs):

1. CO1: Identify and scope a research topic or industry problem relevant to drone imaging and spectral analysis. (Understand, Remember)
2. CO2: Develop a project plan and timeline, defining clear objectives and scope. (Apply, Create)
3. CO3: Conduct data collection using drones and ensure data quality and accuracy. (Apply, Analyze)
4. CO4: Process and analyze data using advanced tools and techniques. (Apply, Evaluate)
5. CO5: Present project findings through structured reports and oral presentations. (Evaluate, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Project Proposal (6 hours)

- **Topics:** Identifying a Research Topic or Industry Problem; Defining Objectives and Scope; Project Planning and Timeline.

Module 2: Literature Review and Methodology (6 hours)

- **Topics:** Conducting a Literature Review; Selecting Appropriate Research Methods.

Module 3: Data Collection (6 hours)

- **Topics:** Planning and Executing Drone Flights; Ensuring Data Quality and Accuracy.

Module 4: Data Processing (6 hours)

- **Topics:** Using Pix4D and Other Tools for Image Processing; Generating Relevant Outputs (Orthomosaics, 3D Models, etc.).

Module 5: Data Analysis (6 hours)

- **Topics:** Applying Analytical Techniques; Interpreting Results.

Module 6: Report Writing (6 hours)

- **Topics:** Structuring the Final Report; Writing and Revising the Report.

Module 7: Presentation (6 hours)

- **Topics:** Preparing for the Oral Presentation; Presenting Findings to an Audience.

Textbooks and References:

- "Research Design: Qualitative, Quantitative, and Mixed Methods Approaches" by John W. Creswell
- Project Management and Technical Documentation Resources

Domain Track: Software Technology (18 Credits)

Advanced Java(112hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUST1051	Advanced Java	4	0+2+2

Course Description: This course delves into advanced topics in Java programming, including multithreading, networking, database connectivity, and Java frameworks. Students will gain a deep understanding of Java's capabilities and how to apply them in real-world applications.

Course Objectives:

1. Master advanced Java programming concepts and techniques.
2. Implement Java-based solutions for web, and database applications.
3. Apply Java frameworks and best practices in real-world scenarios.

Course Outcomes (COs):

1. CO1: Understand and apply advanced object-oriented programming concepts in Java. (Understand, Apply)
2. CO2: Implement servlet to control web applications. (Apply, Analyse)
3. CO3: Develop dynamic web pages using Java Server Page. (Apply, Create)
4. CO4: Integrate Java applications with databases using JDBC and manage transactions. (Apply, Evaluate)
5. CO5: Utilize Java EE and frameworks to build robust enterprise applications. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Web Architecture (6 hours)

- Overview of Web Architecture, Client-Server Model, HTTP Protocol Basics, Web Servers and Application Servers, Introduction to MVC Architecture.

Experiments:

- Set up a simple HTTP server and client.
- Create a basic web page and deploy it on a web server.
- Implement a simple MVC pattern in a web application.
- Analyze HTTP request and response headers using browser tools.
- Configure and deploy an application on an Apache Tomcat server..

Module 2: Introduction to GitHub (6 hours)

- **Topics:** Version Control Basics, Git Commands and Workflow, GitHub Repository Management, Branching and Merging, Collaborating with GitHub.

Experiments:

- Initialize a Git repository and commit changes.
- Push local repository to GitHub and manage remote repositories.
- Create and merge branches.
- Resolve merge conflicts.
- Collaborate with others using pull requests and issues on GitHub.

Module 3: Java Database Connectivity (JDBC) (10 hours)

- **Topics:** Introduction to Java Database Connectivity (JDBC), JDBC Architecture and Drivers, Establishing Database Connections, Executing SQL Queries and Updates, Handling ResultSets, Prepared Statements and Callable Statements, Batch Processing in JDBC, Transaction Management, Connection Pooling

Experiments:

- Set up a database and connect to it using JDBC.
- Execute SQL SELECT queries and process the results.
- Perform INSERT, UPDATE, and DELETE operations using JDBC.
- Handle SQL exceptions and errors.
- Develop a simple CRUD application using JDBC.
- Use prepared statements and callable statements
- Implement batch processing in JDBC
- Manage transactions effectively

- Utilize connection pooling for efficient database access.

Module 4: Introduction to Servlets (6 hours)

- **Topics:** Introduction to Servlets, Servlet Lifecycle, Handling Requests and Responses, Servlet Configuration and Context, Session Management.

Experiments:

- Create and deploy a simple servlet.
- Handle GET and POST requests in a servlet.
- Implement session tracking using cookies and HTTP sessions.
- Use servlet context and configuration for initialization parameters.
- Develop a login system using servlets and session management.

Module 5: Advanced Servlet Concepts (6 hours)

- **Topics:**Request Dispatching and Redirecting, Servlet Filters, Servlet Listner, Asynchronous Servlets, Error Handling in Servlets, Security and Authentication in Servlets

Experiments:

- Implement request forwarding and redirection.
- Create and configure a servlet filter.
- Develop an asynchronous servlet for long-running tasks.
- Implement custom error pages for handling different HTTP errors.
- Secure a servlet using basic authentication and HTTPS.

Module 6:Introduction to JSP (8 hours)

- **Topics:** Basics of JavaServer Pages (JSP), JSP Lifecycle, JSP Directives, Scriptlets, and Expressions, JSP Implicit Objects, Using JavaBeans in JSP,JSP Tag Libraries (JSTL), Custom Tags in JSP, Expression Language (EL), JSP and MVC Architecture, Error Handling in JSP

- **Experiments:**
  - Create a basic JSP page and deploy it.
  - Use JSP scriptlets to embed Java code in HTML.
  - Access JSP implicit objects to handle requests and responses.
  - Integrate a JavaBean in a JSP page.
  - Create a simple form processing application using JSP and JavaBeans.
  - Use JSTL core tags to manage control flow and iteration.

- Create and use custom JSP tags.
- Utilize EL to access Java objects and properties.
- Implement MVC architecture with JSP as the view.
- Develop error handling mechanisms in JSP

Module 7: Introduction to Hibernate (8 hours)

- **Topics:** Basics of Hibernate ORM, Hibernate Architecture, Configuring Hibernate, Mapping Entities to Tables, CRUD Operations with Hibernate

Experiments:

- Set up Hibernate in a Java project.
- Map a Java class to a database table using annotations.
- Perform CRUD operations with Hibernate.
- Configure Hibernate using XML.
- Implement relationships (one-to-many, many-to-many) in Hibernate

Textbooks and References:

- "Core Servlets and JavaServer Pages" by Marty Hall and Larry Brown

Angular(112hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUST1052	Angular	4	0+2+2

Course Description: This course provides a comprehensive understanding of Angular, focusing on advanced concepts and techniques required to build dynamic, responsive, and efficient web applications. Students will gain hands-on experience in developing and deploying Angular applications.

Course Objectives:

1. Master the fundamentals and advanced concepts of Angular.
2. Develop and optimize complex Angular applications.
3. Implement best practices and deploy Angular applications.

Course Outcomes (COs):

1. CO1: Understand the core concepts and architecture of Angular applications. (Understand, Remember)
2. CO2: Develop Angular applications using TypeScript and advanced Angular features. (Apply, Create)

3. CO3: Implement data binding, directives, and component communication in Angular. (Apply, Analyze)
4. CO4: Utilize Angular services, dependency injection, and HTTP client for dynamic applications. (Apply, Evaluate)
5. CO5: Optimize, test, and deploy Angular applications following best practices. (Apply, Create)

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	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Angular and TypeScript Basics (8 hours)

Theory

Overview of Angular framework; Setting up the development environment (Node.js, Angular CLI); Creating a new Angular project with Angular CLI; Introduction to TypeScript; TypeScript features and syntax; Variables, types, and functions; Classes and interfaces; TypeScript in Angular.

Practice

- Experiment 1.1: Install Node.js and Angular CLI.
- Experiment 1.2: Create a new Angular project using Angular CLI.
- Experiment 1.3: Explore the Angular project structure.
- Experiment 1.4: Configure the Angular development environment.
- Experiment 1.5: Implement a simple Angular application.
- Experiment 1.6: Conduct a seminar on the evolution of Angular.
- Experiment 1.7: Present case studies of successful Angular applications.
- Experiment 1.8: Install and set up TypeScript.
- Experiment 1.9: Implement basic TypeScript syntax (variables, types, functions).
- Experiment 1.10: Create and use classes and interfaces in TypeScript.

Module 2: Angular Architecture (8 hours)

Theory

Components, templates, and modules; Component lifecycle hooks; Creating and using Angular modules; Angular application structure and best practices.

Practice

- Experiment 2.1: Create and use Angular components.
- Experiment 2.2: Implement component lifecycle hooks.
- Experiment 2.3: Create and manage Angular modules.
- Experiment 2.4: Structure an Angular application following best practices.
- Experiment 2.5: Document and present the Angular component lifecycle.
- Experiment 2.6: Compare different approaches to structuring Angular applications.
- Experiment 2.7: Conduct a seminar on Angular architecture.
- Experiment 2.8: Implement a modular Angular application.
- Experiment 2.9: Present case studies of Angular application architectures.
- Experiment 2.10: Research and present advanced Angular architectural patterns.

Module 3: Data Binding and Directives (8 hours)

Theory

Property binding and event binding; Two-way data binding with ngModel; Built-in directives (ngIf, ngFor, ngClass, ngStyle); Custom directives.

Practice

- Experiment 3.1: Implement property binding in an Angular application.
- Experiment 3.2: Use event binding to handle user interactions.
- Experiment 3.3: Implement two-way data binding with ngModel.
- Experiment 3.4: Use built-in directives (ngIf, ngFor, ngClass, ngStyle).
- Experiment 3.5: Create and use custom directives.
- Experiment 3.6: and present the use of data binding in Angular.
- Experiment 3.7: Compare different data binding techniques.
- Experiment 3.8: Conduct a seminar on Angular directives.
- Experiment 3.9: Implement a directive-based project.
- Experiment 3.10: Present case studies of data binding and directives in Angular applications.

Module 4: Component Communication (8 hours)

Theory

Input and output decorators; Event emitters; Parent-child component communication; ViewChild and ContentChild decorators.

Practice

- Experiment 4.1: Implement component communication using input and output decorators.
- Experiment 4.2: Use event emitters for component interaction.
- Experiment 4.3: Implement parent-child communication in Angular.
- Experiment 4.4: Use ViewChild and ContentChild decorators.
- Experiment 4.5: Document and present component communication techniques.
- Experiment 4.6: Compare different approaches to component communication.

- Experiment 4.7: Conduct a seminar on advanced component communication.
- Experiment 4.8: Implement a project involving complex component interaction.
- Experiment 4.9: Present case studies of component communication in Angular applications.
- Experiment 4.10: Research and present advanced component communication patterns.

Module 5: Services, Dependency Injection and HTTP Client (8 hours)

- **Theory**: Creating and using services; Dependency injection in Angular; Hierarchical injectors; Managing application-wide services; Introduction to Angular HttpClient; Performing GET,POST,PUT,DELETE requests

Practice

- Experiment 5.1: Creating and Injecting a Simple Service in Angular
- Experiment 5.2: Exploring Angular Dependency Injection: Singleton Services
- Experiment 5.3: Hierarchical Dependency Injection: Providing Services in Modules
- Experiment 5.4: Managing Application-Wide Services in Angular
- Experiment 5.5: Introduction to Angular HttpClient: Performing GET Requests.
- Experiment 5.6: Performing POST Requests with Angular HttpClient
- Experiment 5.7: Updating Data with PUT Requests in Angular HttpClient
- Experiment 5.8: Deleting Data with DELETE Requests in Angular HttpClient
- Experiment 5.9: Handling HTTP Errors and Responses in Angular HttpClient
- Experiment 5.10: Interceptors in Angular HttpClient: Modifying Requests and Responses

Module 6: Routing and Navigation (8 hours)

Theory: Configuring routes; RouterLink and router-outlet; Route guards and lazy loading; Child routes and nested routing.

Practice

- Experiment 6.1: Configure routes in an Angular application.
- Experiment 6.2: Use RouterLink and router-outlet for navigation.
- Experiment 6.3: Implement route guards for security.
- Experiment 6.4: Implement lazy loading for performance optimization.
- Experiment 6.5: Create and use child routes and nested routing.
- Experiment 6.6: Document and present routing and navigation techniques.
- Experiment 6.7: Compare different approaches to routing in Angular.
- Experiment 6.8: Conduct a seminar on advanced routing techniques.
- Experiment 6.9: Implement a project involving complex routing and navigation.
- Experiment 6.10: Present case studies of routing and navigation in Angular applications.

Module 7: RxJX, Testing and Deployment (8 hours)

Theory

Introduction to RxJS; Using observables with Angular HttpClient; Introduction to testing in Angular; Deployment options(FireBase,AWS).

Practice

- Experiment 7.1: Creating and Subscribing to Observables in RxJS
- Experiment 7.2: Using RxJS Operators for Data Transformation
- Experiment 7.3: Combining Observables Using RxJS Operators
- Experiment 7.4: Implementing Error Handling in RxJS
- Experiment 7.5: Using Observables with Angular HttpClient: Performing GET Requests
- Experiment 7.6: Sending Data with POST Requests Using Angular HttpClient and RxJS
- Experiment 7.7: Testing Angular Services and HTTP Requests with HttpClientTestingModule
- Experiment 7.8: Testing Angular Services that Use HttpClient.
- Experiment 7.9: Deploying an Angular Application to Firebase and AWS.
- Experiment 7.10: Best Practices for Angular Application Deployment.

Project (28 hours)

Textbooks and References:

- Angular Documentation: Angular.io
- Books:
  - "Angular Up & Running" by Shyam Seshadri
  - "Pro Angular" by Adam Freeman
- Online Courses:
  - Angular courses on platforms like Udemy, Coursera, and Pluralsight
- Tools:
  - Visual Studio Code, Angular CLI, Postman

Course 3: Spring Boot (4 Credits) (112 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUST1053	Spring Boot	4	0+2+2

Course Description: This course focuses on Spring Boot, a framework for building production-ready applications with Spring. Students will learn how to create RESTful services, work with databases, and deploy Spring Boot applications.

Course Objectives:

1. Understand the core concepts of Spring and Spring Boot.
2. Develop RESTful services and secure Spring Boot applications.
3. Implement data access and perform testing in Spring Boot applications.

Course Outcomes (COs):

1. CO1: Explain the features and benefits of Spring and Spring Boot. (Understand, Remember)
2. CO2: Set up and configure a Spring Boot project. (Apply, Create)
3. CO3: Develop RESTful services using Spring Boot. (Apply, Create)
4. CO4: Implement data access using Spring Data JPA and perform CRUD operations. (Apply, Evaluate)
5. CO5: Secure Spring Boot applications and perform testing. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Spring (8 hours)

Theory

Overview of Spring and Its Features; Dependency Injection using Setter and Constructor; Bean and Application Context.

Practice

- Experiment 1.1: Install and set up a Spring project.
- Experiment 1.2: Create and configure beans using XML and Java annotations.
- Experiment 1.3: Implement dependency injection using setter methods.
- Experiment 1.4: Implement dependency injection using constructor methods.
- Experiment 1.5: Explore the ApplicationContext in Spring.
- Experiment 1.6: Document and present the features of the Spring framework.

- Experiment 1.7: Compare different dependency injection methods.
- Experiment 1.8: Conduct a seminar on Spring framework best practices.
- Experiment 1.9: Present case studies of successful Spring applications.
- Experiment 1.10: Research and present the evolution of the Spring framework.

Module 2: Introduction to Spring Boot (8 hours)

Theory

Overview of Spring Boot and Its Features; Setting Up a Spring Boot Project.

Practice

- Experiment 2.1: Install and configure Spring Boot.
- Experiment 2.2: Set up a new Spring Boot project using Spring Initializr.
- Experiment 2.3: Explore the Spring Boot project structure.
- Experiment 2.4: Implement a simple Spring Boot application.
- Experiment 2.5: Document and present the benefits of using Spring Boot.
- Experiment 2.6: Compare Spring Boot with traditional Spring.
- Experiment 2.7: Conduct a seminar on Spring Boot features.
- Experiment 2.8: Implement a Spring Boot-based project.
- Experiment 2.9: Present case studies of Spring Boot applications.
- Experiment 2.10: Research and present advanced Spring Boot features.

Module 3: Spring Boot Basics (6 hours)

Theory

Spring Boot Annotations and Configuration; Creating RESTful Services.

Practice

- Experiment 3.1: Implement RESTful services using Spring Boot annotations.
- Experiment 3.2: Configure Spring Boot applications using properties and YAML files.
- Experiment 3.3: Develop a simple RESTful API using Spring Boot.
- Experiment 3.4: Implement exception handling in Spring Boot RESTful services.
- Experiment 3.5: Document and present the use of annotations in Spring Boot.
- Experiment 3.6: Compare different configuration methods in Spring Boot.
- Experiment 3.7: Conduct a seminar on RESTful service design.
- Experiment 3.8: Implement a project involving complex RESTful services.
- Experiment 3.9: Present case studies of RESTful services in Spring Boot.
- Experiment 3.10: Research and present best practices for Spring Boot configuration.

Module 4: Data Access with Spring Boot (8 hours)

Theory

Spring Data JPA; Connecting to Databases; CRUD Operations.

Practice

- Experiment 4.1: Set up database connectivity in a Spring Boot application.
- Experiment 4.1: Implement CRUD operations using Spring Data JPA.
- Experiment 4.1: Configure data sources and JPA properties.
- Experiment 4.1: Develop a simple database-driven application.
- Experiment 4.1: Document and present the use of Spring Data JPA.
- Experiment 4.1: Compare different database connectivity methods.
- Experiment 4.1: Conduct a seminar on data access best practices.
- Experiment 4.1: Implement a project involving complex data access operations.
- Experiment 4.1: Present case studies of data-driven Spring Boot applications.
- Experiment 4.1: Research and present advanced features of Spring Data JPA.

Module 5: Spring Boot Security (8 hours)

Theory

Securing Applications with Spring Security; Authentication and Authorization.

Practice

- Experiment 5.1: Implement security features in a Spring Boot application.
- Experiment 5.2: Configure authentication using Spring Security.
- Experiment 5.3: Implement authorization roles and permissions.
- Experiment 5.4: Secure RESTful APIs with Spring Security.
- Experiment 5.5: Document and present the use of Spring Security.
- Experiment 5.6: Compare different security methods in Spring Boot.
- Experiment 5.7: Conduct a seminar on application security best practices.
- Experiment 5.8: Implement a project involving complex security requirements.
- Experiment 5.9: Present case studies of secure Spring Boot applications.
- Experiment 5.10: Research and present advanced security features in Spring Boot.

Module 6: Spring Boot Testing (8 hours)

Theory

Writing Unit and Integration Tests; Testing with Spring Boot Test.

Practice

- Experiment 6.1: Implement unit tests using JUnit and Mockito.

- Experiment 6.2: Write integration tests for Spring Boot applications.
- Experiment 6.3: Use Spring Boot Test for testing Spring Boot applications.
- Experiment 6.4: Configure test environments and properties.
- Experiment 6.5: Document and present the use of testing frameworks.
- Experiment 6.6: different testing methods in Spring Boot.
- Experiment 6.7: Conduct a seminar on testing best practices.
- Experiment 6.8: Implement a project involving comprehensive testing.
- Experiment 6.9: Present case studies of tested Spring Boot applications.
- Experiment 6.10: Research and present advanced testing techniques in Spring Boot.

Module 7: Case Studies and Applications (8 hours)

Theory

Real-World Spring Boot Projects; Hands-on Labs and Assignments.

Practice

- Experiment 7.1: Analyze real-world Spring Boot projects.
- Experiment 7.2: Implement a Spring Boot application based on a case study.
- Experiment 7.3: hands-on labs for Spring Boot features.
- Experiment 7.4: Develop a comprehensive Spring Boot project.
- Experiment 7.5: Document and present project findings.
- Experiment 7.6: Compare different Spring Boot applications.
- Experiment 7.7: Conduct a seminar on Spring Boot application development.
- Experiment 7.8: Implement a project involving multiple Spring Boot features.
- Experiment 7.9: Present case studies of successful Spring Boot projects.
- Experiment 7.10: Research and present future trends in Spring Boot development.

Project (28 Hours)

Textbooks and References:

- "Spring Boot in Action" by Craig Walls
- "Pro Spring Boot" by Felipe Gutierrez

Product Development(168hours)

Course Code	Course Title	Credits	Type (T+P+Pj)

CUST1054	Product Development	6	0+0+6

Course Description: This course provides a comprehensive overview of the product development process, from ideation to launch. Students will learn about product design, development methodologies, project management, and user experience (UX) design.

Course Objectives:

1. Understand the complete product development lifecycle.
2. Apply design and development methodologies to create products.
3. Manage projects effectively and ensure a great user experience.

Course Outcomes (COs):

1. CO1: Explain the roles and responsibilities in the product development lifecycle. (Understand, Remember)
2. CO2: Utilize brainstorming techniques and market research for product ideation. (Apply, Analyze)
3. CO3: Design user-centered products using wireframing and prototyping techniques. (Apply, Create)
4. CO4: Implement Agile, Scrum, and Lean methodologies in product development. (Apply, Create)
5. CO5: Manage resources, risks, and communication in product development projects. (Evaluate, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Product Development (6 hours)

- **Topics:** Overview of Product Development Lifecycle; Roles and Responsibilities in Product Development.

Module 2: Ideation and Conceptualization (6 hours)

- **Topics:** Brainstorming Techniques; Market Research and Competitive Analysis; Creating Product Roadmaps.

Module 3: Product Design (8 hours)

- **Topics:** User-Centered Design Principles; Wireframing and Prototyping; User Interface (UI) Design.

Module 4: Development Methodologies (8 hours)

- **Topics:** Agile and Scrum Frameworks; Lean Product Development; Kanban and Continuous Delivery.

Module 5: Project Management (8 hours)

- **Topics:** Planning and Scheduling; Resource Allocation and Risk Management; Communication and Collaboration Tools.

Module 6: User Experience (UX) (8 hours)

- **Topics:** Conducting User Research; Usability Testing; Iterative Design and Feedback Loops.

Module 7: Technical Development (8 hours)

- **Topics:** Backend and Frontend Development; Integration and APIs; DevOps and Continuous Integration/Continuous Deployment (CI/CD).

Module 8: Quality Assurance (8 hours)

- **Topics:** Testing Strategies and Automation; Performance and Security Testing; Bug Tracking and Resolution.

Module 9: Product Launch (8 hours)

- **Topics:** Marketing and Go-to-Market Strategies; Customer Support and Feedback Collection; Post-Launch Evaluation and Iteration.

Module 10: Case Studies and Applications (8 hours)

- **Topics:** Real-World Product Development Projects; Hands-on Labs and Assignments.

Textbooks and References:

- "Inspired: How To Create Products Customers Love" by Marty Cagan
- "Lean Product and Lean Analytics" by Ben Yoskovitz and Alistair Croll

Domain Track: Mobile App Development (12 Credits)

Introduction to Mobile App Development(70hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUMA1071	Introduction to Mobile App Development	3	1+2+0

Course Description: This foundational course provides an overview of mobile app development, focusing on the principles of designing and building mobile applications. Students will learn the basics of mobile operating systems, development environments, and user interface design.

Course Objectives:

1. Understand the mobile app development ecosystems and operating systems.
2. Gain proficiency in setting up development environments and using IDEs.
3. Learn the principles of mobile UI/UX design and basic programming concepts.

Course Outcomes (COs):

1. CO1: Explain the mobile app ecosystems and operating systems. (Understand, Remember)
2. CO2: Set up and use development environments and IDEs for mobile app development. (Apply, Analyze)
3. CO3: Design mobile user interfaces following UI/UX principles. (Apply, Create)
4. CO4: Develop basic mobile applications using JavaScript and Dart. (Apply, Create)
5. CO5: Utilize version control systems for collaborative development. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Mobile App Development (6 hours)

- **Topics:** Overview of Mobile App Ecosystems; Mobile Operating Systems (iOS, Android).

Experiments:

Experiment 1.1. Explore the mobile app ecosystems for iOS and Android.
Experiment 1.2. Compare different mobile operating systems.
Experiment 1.3. Document and present the key features of iOS and Android.
Experiment 1.4. Research and present current trends in mobile app development.
Experiment 1.5. Analyze case studies of successful mobile apps.
Experiment 1.6. Discuss the impact of mobile apps on modern society.
Experiment 1.7. Investigate career opportunities in mobile app development.
Experiment 1.8. Present findings on the evolution of mobile operating systems.
Experiment 1.9. Research the future directions of mobile app ecosystems.
Experiment 1.10. Explore the role of mobile apps in various industries.

Module 2: Development Environments (6 hours)

- **Topics:** Setting Up Development Environments (Xcode, Android Studio); Introduction to Integrated Development Environments (IDEs).

Experiments:

Experiment 2.1. Set up Xcode for iOS development.
Experiment 2.2. Set up Android Studio for Android development.
Experiment 2.3. Explore the features of Xcode and Android Studio.
Experiment 2.4. Document and present the setup process for both IDEs.
Experiment 2.5. Compare the development environments for iOS and Android.
Experiment 2.6. Conduct a seminar on the importance of IDEs in mobile app development.
Experiment 2.7. Implement a project involving basic app development in Xcode.
Experiment 2.8. Implement a project involving basic app development in Android Studio.
Experiment 2.9. Present case studies on the use of IDEs in successful mobile apps.
Experiment 2.10. Research the development of new IDEs for mobile app development.

Module 3: User Interface Design (8 hours)

- **Topics:** Mobile UI/UX Design Principles; Wireframing and Prototyping.

Experiments:

Experiment 3.1. Design a basic user interface for a mobile app.
Experiment 3.2. Create wireframes for a mobile app using design tools.
Experiment 3.3. Develop prototypes for a mobile app.
Experiment 3.4. Document and present the UI/UX design principles.
Experiment 3.5. Compare different UI/UX design tools.
Experiment 3.6. Conduct a seminar on the importance of UI/UX design in mobile apps.
Experiment 3.7. Implement a project involving UI/UX design for a mobile app.
Experiment 3.8. Present case studies on the impact of UI/UX design in successful mobile apps.
Experiment 3.9. Research the development of new UI/UX design principles.
Experiment 3.10. Investigate the challenges in mobile UI/UX design.

Module 4: App Development Lifecycle (8 hours)

- **Topics:** Planning and Requirements Gathering; Development, Testing, and Deployment.

Experiments:

Experiment 4.1. Plan and gather requirements for a mobile app project.
Experiment 4.2. Document and present the app development lifecycle.
Experiment 4.3. Develop a basic mobile app following the development lifecycle.
Experiment 4.4. Test a mobile app using various testing tools.
Experiment 4.5. Deploy a mobile app to a virtual device.
Experiment 4.6. Compare different app development lifecycles.
Experiment 4.7. Conduct a seminar on the importance of planning and testing in app development.
Experiment 4.8. Implement a project involving the complete app development lifecycle.
Experiment 4.9. Present case studies on the app development lifecycle in successful mobile apps.
Experiment 4.10. Research the challenges in the app development lifecycle.

Module 5: Introduction to Cross-Platform Development (6 hours)

- **Topics:** Advantages of Cross-Platform Development; Overview of React Native and Flutter.

Experiments:

Experiment 5.1. Explore the advantages of cross-platform development.
Experiment 5.2. Set up React Native for cross-platform app development.
Experiment 5.3. Set up Flutter for cross-platform app development.
Experiment 5.4. Document and present the features of React Native and Flutter.
Experiment 5.5. Compare React Native and Flutter for cross-platform development.
Experiment 5.6. Conduct a seminar on the importance of cross-platform development.

Experiment 5.7. Implement a project involving basic app development in React Native.
Experiment 5.8. Implement a project involving basic app development in Flutter.
Experiment 5.9. Present case studies on successful cross-platform apps.
Experiment 5.10. Research the future of cross-platform development.

Module 6: Basic Programming Concepts (6 hours)

- **Topics:** Introduction to JavaScript and Dart; Mobile-Specific Programming Constructs.

Experiments:

Experiment 6.1. Write basic programs in JavaScript.
Experiment 6.2. Write basic programs in Dart.
Experiment 6.3. Explore mobile-specific programming constructs.
Experiment 6.4. Document and present basic programming concepts.
Experiment 6.5. Compare JavaScript and Dart for mobile app development.
Experiment 6.6. Conduct a seminar on the importance of programming in mobile app development.
Experiment 6.7. Implement a project involving basic programming in JavaScript.
Experiment 6.8. Implement a project involving basic programming in Dart.
Experiment 6.9. Present case studies on the use of programming in successful mobile apps.
Experiment 6.10. Research the development of new programming languages for mobile apps.

Module 7: Version Control Systems (6 hours)

- **Topics:** Introduction to Git and GitHub; Collaborative Development Practices.

Experiments:

Experiment 7.1. Set up a Git repository for a mobile app project.
Experiment 7.2. Explore the features of GitHub for collaborative development.
Experiment 7.3. Document and present the use of version control systems.
Experiment 7.4. Compare different version control systems for mobile app development.
Experiment 7.5. Conduct a seminar on the importance of version control systems.
Experiment 7.6. Implement a project involving collaborative development using Git and GitHub.
Experiment 7.7. Present case studies on the use of version control systems in successful mobile apps.
Experiment 7.8. Research the development of new version control systems.
Experiment 7.9. Investigate the challenges in using version control systems.
Experiment 7.10. Analyze the impact of version control systems on collaborative development.

Textbooks and References:

- "Mobile App Development for Beginners" by John Horton
- "Don't Make Me Think: A Common Sense Approach to Web Usability" by Steve Krug

React Native Development(84hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUMA1072	React Native Development	3	0+2+1

Course Description: This course covers the development of mobile applications using React Native. Students will learn how to build cross-platform apps with a focus on performance and native-like user experiences.

Course Objectives:

1. Understand the core concepts and components of React Native.
2. Develop cross-platform mobile applications using React Native.
3. Optimize and deploy React Native applications.

Course Outcomes (COs):

1. CO1: Explain the fundamental concepts and components of React Native. (Understand, Remember)
2. CO2: Develop mobile applications using React Native with effective state management. (Apply, Create)
3. CO3: Implement navigation and routing in React Native applications. (Apply, Analyze)
4. CO4: Integrate APIs and handle asynchronous data in React Native apps. (Apply, Evaluate)
5. CO5: Optimize, test, and deploy React Native applications for production. (Apply, Create)

6. **CO-PO-PSO Mapping:**

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to React Native (6 hours)

- **Topics:** Overview of React Native; Setting Up the Development Environment.

Experiments:

Experiment 1.1. Set up the React Native development environment.
Experiment 1.2. Create a "Hello World" app in React Native.
Experiment 1.3. Explore the React Native project structure.
Experiment 1.4. Document and present the React Native setup process.
Experiment 1.5. Compare React Native with other mobile app development frameworks.
Experiment 1.6. Conduct a seminar on the benefits of using React Native.
Experiment 1.7. Implement a project involving basic app development in React Native.
Experiment 1.8. Present case studies on successful React Native apps.
Experiment 1.9. Research the future of React Native development.
Experiment 1.10. Investigate the challenges in setting up React Native environments.

Module 2: Core Concepts and Components (6 hours)

- **Topics:** JSX and React Components; Styling Components with Flexbox.

Experiments:

Experiment 2.1. Create and style components using JSX and Flexbox.
Experiment 2.2. Implement different types of React components.
Experiment 2.3. Explore component lifecycle methods in React Native.
Experiment 2.4. Document and present the core concepts of React Native.
Experiment 2.5. Compare different component styling techniques.
Experiment 2.6. Conduct a seminar on component-based development in React Native.
Experiment 2.7. Implement a project involving advanced component development.
Experiment 2.8. Present case studies on component usage in successful React Native apps.
Experiment 2.9. Research the development of new components in React Native.
Experiment 2.10. Investigate the challenges in component styling and management.

Module 3: State Management (8 hours)

- **Topics:** State and Props; Context API and Redux for State Management.

Experiments:

Experiment 3.1. Manage state using React's built-in state and props.
Experiment 3.2. Implement state management using Context API.
Experiment 3.3. Use Redux for state management in React Native.
Experiment 3.4. Document and present state management techniques.
Experiment 3.5. Compare Context API and Redux for state management.
Experiment 3.6. Conduct a seminar on state management best practices.
Experiment 3.7. Implement a project involving complex state management.
Experiment 3.8. Present case studies on state management in React Native apps.
Experiment 3.9. Research the development of new state management tools.
Experiment 3.10. Investigate the challenges in state management.

Module 4: Navigation and Routing (8 hours)

- **Topics:** React Navigation Library; Stack, Tab, and Drawer Navigation.

Experiments:

- Implement stack navigation using React Navigation.
- Implement tab navigation using React Navigation.
- Implement drawer navigation using React Navigation.
- Document and present navigation techniques.
- Compare different navigation methods.
- Conduct a seminar on navigation best practices.
- Implement a project involving advanced navigation techniques.
- Present case studies on navigation in successful React Native apps.
- Research the development of new navigation tools.
- Investgate the challenges in implementing navigation.

Module 5: Handling User Input (8 hours)

- **Topics:** Forms and User Input Handling; Touch and Gesture Handling.

Experiments:

Experiment 5.1. Create forms and handle user input in React Native.
Experiment 5.2. Implement touch and gesture handling.
Experiment 5.3. Explore form validation techniques.
Experiment 5.4. Document and present user input handling techniques.
Experiment 5.5. Compare different methods for handling user input.

Experiment 5.6. Conduct a seminar on user input best practices.
Experiment 5.7. Implement a project involving advanced user input handling.
Experiment 5.8. Present case studies on user input handling in successful React Native apps.
Experiment 5.9. Research the development of new user input handling tools.
Experiment 5.10. Investigate the challenges in handling user input.

Module 6: Networking and API Integration (8 hours)

- **Topics:** Fetch API and Axios; Handling Asynchronous Data.

Experiments:

Experiment 6.1. Integrate APIs using Fetch API.
Experiment 6.2. Integrate APIs using Axios.
Experiment 6.3. Handle asynchronous data in React Native.
Experiment 6.4. Document and present API integration techniques.
Experiment 6.5. Compare Fetch API and Axios for API integration.
Experiment 6.6. Conduct a seminar on API integration best practices.
Experiment 6.7. Implement a project involving complex API integration.
Experiment 6.8. Present case studies on API integration in successful React Native apps.
Experiment 6.9. Research the development of new API integration tools.
Experiment 6.10. Investigate the challenges in integrating APIs.

Module 7: Native Modules and Plugins (8 hours)

- **Topics:** Linking Native Code; Using Third-Party Libraries.

Experiments:

Experiment 7.1. Link native code in React Native.
Experiment 7.2. Use third-party libraries in React Native.
Experiment 7.3. Explore the process of creating custom native modules.
Experiment 7.4. Document and present native module integration techniques.
Experiment 7.5. Compare different methods for integrating native modules.
Experiment 7.6. Conduct a seminar on native module best practices.
Experiment 7.7. Implement a project involving advanced native module integration.
Experiment 7.8. Present case studies on native module usage in successful React Native apps.
Experiment 7.9. Research the development of new native modules.
Experiment 7.10. Investigate the challenges in linking native code.

Textbooks and References:

- "React Native in Action" by Nader Dabit

- "Learning React Native: Building Native Mobile Apps with JavaScript" by Bonnie Eisenman

Flutter Development(84hours)

Course Description: This course focuses on building mobile applications using Flutter. Students will learn how to create high-performance, visually attractive cross-platform apps with Dart.

Course Objectives:

1. Understand the core concepts and components of Flutter and Dart.
2. Develop cross-platform mobile applications using Flutter.
3. Optimize and deploy Flutter applications for production.

Course Outcomes (COs):

1. CO1: Explain the fundamental concepts and components of Flutter and Dart. (Understand, Remember)
2. CO2: Develop mobile applications using Flutter with effective state management. (Apply, Create)
3. CO3: Implement navigation and routing in Flutter applications. (Apply, Analyze)
4. CO4: Integrate APIs and handle asynchronous data in Flutter apps. (Apply, Evaluate)
5. CO5: Optimize, test, and deploy Flutter applications for production. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Introduction to Flutter (6 hours)

- **Topics:** Overview of Flutter and Dart; Setting Up the Flutter Development Environment.

Experiments:

Experiment 1.1. Set up the Flutter development environment.
Experiment 1.2. Create a "Hello World" app in Flutter.
Experiment 1.3. Explore the Flutter project structure.
Experiment 1.4. Document and present the Flutter setup process.
Experiment 1.5. Compare Flutter with other mobile app development frameworks.
Experiment 1.6. Conduct a seminar on the benefits of using Flutter.
Experiment 1.7. Implement a project involving basic app development in Flutter.
Experiment 1.8. Present case studies on successful Flutter apps.
Experiment 1.9. Research the future of Flutter development.
Experiment 1.10. Investigate the challenges in setting up Flutter environments.

Module 2: Dart Programming Basics (6 hours)

- **Topics:** Syntax and Basic Constructs; Object-Oriented Programming in Dart.

Experiments:

- Write basic programs in Dart.
- Explore Dart's syntax and constructs.
- Implement object-oriented programming concepts in Dart.
- Document and present Dart programming basics.
- Compare Dart with other programming languages for mobile app development.
- Conduct a seminar on the importance of Dart in Flutter development.
- Implement a project involving basic programming in Dart.
- Present case studies on Dart usage in successful Flutter apps.
- Research the development of new programming languages for mobile apps.
- Investigate the challenges in Dart programming.

Module 3: Flutter Widgets (8 hours)

- **Topics:** Stateless and Stateful Widgets; Building Complex Layouts.

Experiments:

Experiment 3.1. Create and style stateless widgets in Flutter.
Experiment 3.2. Create and style stateful widgets in Flutter.
Experiment 3.3. Implement complex layouts using Flutter widgets.
Experiment 3.4. Document and present Flutter widget concepts.
Experiment 3.5. Compare stateless and stateful widgets.
Experiment 3.6. Conduct a seminar on widget-based development in Flutter.
Experiment 3.7. Implement a project involving advanced widget development.
Experiment 3.8. Present case studies on widget usage in successful Flutter apps.
Experiment 3.9. Research the development of new Flutter widgets.

Experiment 3.10. Investigate the challenges in widget management and styling.

Module 4: State Management in Flutter (8 hours)

- **Topics:** setState and InheritedWidget; Provider, Riverpod, and Bloc.

Experiments:

Experiment 4.1. Manage state using Flutter's setState.
Experiment 4.2. Implement state management using InheritedWidget.
Experiment 4.3. Use Provider for state management in Flutter.
Experiment 4.4. Document and present state management techniques.
Experiment 4.5. Compare Provider, Riverpod, and Bloc for state management.
Experiment 4.6. Conduct a seminar on state management best practices.
Experiment 4.7. Implement a project involving complex state management.
Experiment 4.8. Present case studies on state management in Flutter apps.
Experiment 4.9. Research the development of new state management tools.
Experiment 4.10. Investigate the challenges in state management.

Module 5: Navigation and Routing (8 hours)

- **Topics:** Navigator and Routing Mechanisms; Nested Navigation and Navigation Patterns.

Experiments:

Experiment 5.1. Implement basic navigation using Navigator.
Experiment 5.2. Implement nested navigation using Flutter's routing mechanisms.
Experiment 5.3. Explore different navigation patterns in Flutter.
Experiment 5.4. Document and present navigation techniques.
Experiment 5.5. Compare different methods for implementing navigation.
Experiment 5.6. Conduct a seminar on navigation best practices.
Experiment 5.7. Implement a project involving advanced navigation techniques.
Experiment 5.8. Present case studies on navigation in successful Flutter apps.
Experiment 5.9. Research the development of new navigation tools.
Experiment 5.10. Investigate the challenges in implementing navigation.

Module 6: Handling User Input (8 hours)

- **Topics:** Forms and Validation; Gesture Detection and Handling.

Experiments:

Experiment 6.1. Create forms and handle user input in Flutter.
Experiment 6.2. Implement form validation in Flutter.
Experiment 6.3. Explore gesture detection and handling in Flutter.

Experiment 6.4. Document and present user input handling techniques.
Experiment 6.5. Compare different methods for handling user input.
Experiment 6.6. Conduct a seminar on user input best practices.
Experiment 6.7. Implement a project involving advanced user input handling.
Experiment 6.8. Present case studies on user input handling in successful Flutter apps.
Experiment 6.9. Research the development of new user input handling tools.
Experiment 6.10. Investigate the challenges in handling user input.

Module 7: Networking and Backend Integration (8 hours)

- **Topics:** HTTP Requests with Dio; Handling Asynchronous Data and JSON Parsing.

Experiments:

Experiment 7.1. Integrate APIs using HTTP requests with Dio.
Experiment 7.2. Handle asynchronous data in Flutter.
Experiment 7.3. Implement JSON parsing in Flutter.
Experiment 7.4. Document and present API integration techniques.
Experiment 7.5. Compare Dio with other HTTP libraries for API integration.
Experiment 7.6. Conduct a seminar on API integration best practices.
Experiment 7.7. Implement a project involving complex API integration.
Experiment 7.8. Present case studies on API integration in successful Flutter apps.
Experiment 7.9. Research the development of new API integration tools.
Experiment 7.10. Investigate the challenges in integrating APIs.

Textbooks and References:

- "Flutter in Action" by Eric Windmill
- "Flutter for Beginners" by Alessandro Biessek

Advanced Mobile App Development Project(84hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUMA1074	Advanced Mobile App Development Project	3	0+0+3

Course Description: In this capstone course, students will apply their knowledge of React Native and Flutter to design, develop, and deploy a comprehensive mobile application. The project will encompass the entire app development lifecycle from ideation to deployment.

Course Objectives:

1. Apply advanced knowledge of mobile app development using React Native or Flutter.

2. Design and develop a comprehensive mobile application from scratch.
3. Deploy and present the mobile application, demonstrating its features and functionality.

Course Outcomes (COs):

1. CO1: Propose and plan a comprehensive mobile application project. (Create, Apply)
2. CO2: Design user interfaces and experiences using wireframes and prototypes. (Create, Evaluate)
3. CO3: Develop and implement core and advanced features in a mobile application. (Apply, Analyze)
4. CO4: Test, debug, and ensure the quality of the mobile application. (Evaluate, Analyze)
5. CO5: Deploy the mobile application to app stores and present the completed project. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Project Proposal and Planning (6 hours)

- **Topics:** Identifying a Project Idea; Defining Objectives, Scope, and Requirements.

Module 2: Design and Prototyping (6 hours)

- **Topics:** Creating Wireframes and Prototypes; User Interface and Experience Design.

Module 3: Development Phase 1: Setting Up (6 hours)

- **Topics:** Choosing React Native or Flutter; Setting Up the Project Structure.

Module 4: Development Phase 2: Core Features (6 hours)

- **Topics:** Implementing Core Features and Functionalities; State Management and Navigation.

Module 5: Development Phase 3: Advanced Features (6 hours)

- **Topics:** Integrating Advanced Features (e.g., Push Notifications, In-App Purchases); Backend Integration and API Usage.

Module 6: Testing and Quality Assurance (6 hours)

- **Topics:** Conducting Unit, Integration, and User Testing; Debugging and Resolving Issues.

Module 7: Performance Tuning (6 hours)

- **Topics:** Profiling the App and Optimizing Performance; Ensuring Cross-Platform Consistency.

Module 8: Deployment and Publishing (6 hours)

- **Topics:** Preparing the App for Release; App Store and Google Play Store Submission Process.

Module 9: Final Presentation and Review (6 hours)

- **Topics:** Presenting the Completed Project; Peer Review and Instructor Feedback.

Module 10: Case Studies and Applications (6 hours)

- **Topics:** Reviewing Other Advanced Mobile Applications; Learning from Industry Best Practices.

Textbooks and References:

- "The Lean Startup" by Eric Ries
- "App Store Optimization: ASO Secrets" by Gabriel Machuret

Domain Track: Gaming and Immersive Learning-AR/VR (20 Credits)

Introduction to Gaming & Simulation(70hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGI1081	Introduction to Gaming & Simulation	2	1+1+0

Course Description:

This course provides an introduction to game development and simulation, focusing on the use of Unity as a game engine. Students will learn about the game development lifecycle, Unity's interface, game object manipulation, asset management, and project management within the Unity environment.

Course Objectives:

1. Understand the fundamentals of game development and the economics behind it.
2. Gain proficiency in using the Unity game engine and its interface.
3. Learn to create and manage game objects, assets, and projects within Unity.

Course Outcomes (COs):

1. CO1: Explain the importance of storyboarding and the economics of game development. (Understand, Remember)
2. CO2: Describe the game production pipeline and various roles in game development. (Understand, Remember)
3. CO3: Navigate and utilize the Unity Editor interface effectively. (Apply, Analyze)
4. CO4: Create and manage game objects and assets in Unity. (Apply, Create)
5. CO5: Implement project management techniques for game development in Unity. (Apply, Create)

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	1	2	-	-	1	1	-	-	1	3	2	3
CO2	3	3	3	2	3	-	1	2	1	1	-	2	3	3	3

CO3	3	3	3	3	3	1	1	2	2	1	2	2	3	3	3
CO4	3	3	3	3	3	1	1	2	3	1	3	3	3	3	3
CO5	3	3	3	3	3	1	1	2	2	3	2	2	3	3	3

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

Course Syllabus:

Module I: Welcome to Game Engine (2+2.5 hours)

- Importance of Storyboarding a Game Idea
- The Economics of Game Development
- Assessing Game Markets and Platforms
- Marketing Methods for Games
- Monetizing Games and Upgrades

Experiments:

Experiment 1.1. Create a storyboard for a simple game idea.
Experiment 1.2. Conduct market research for a chosen game concept.
Experiment 1.3. Develop a basic game marketing strategy.
Experiment 1.4. Outline monetization strategies for a game.
Experiment 1.5. Analyze the economic viability of different game platforms.
Experiment 1.6. Present a game idea with a focus on its market potential.
Experiment 1.7. Evaluate different game genres and their market trends.
Experiment 1.8. Create a business model for a game project.
Experiment 1.9. Design a simple game prototype based on market analysis.
Experiment 1.10. Document the storyboard and market research findings.

Module II: Introduction to Game Production (1 hour)

- Video Game Platforms and Genres
- Describing the Game Production Pipeline
- Game Development Jobs and Roles
- The Game Design Document
- The Technical Design Document
- Getting Started in Unity
- Creating a New Unity Project
- Using the Unity Asset Store
- Source Control for Working in Teams

Experiments:

Experiment 2.1. Create a game design document for a simple game.
Experiment 2.2. Develop a technical design document for the game.
Experiment 2.3. Set up a new Unity project and explore the interface.
Experiment 2.4. Download and import assets from the Unity Asset Store.
Experiment 2.5. Implement version control for a Unity project.
Experiment 2.6. Explore different game genres and their production pipelines.
Experiment 2.7. Analyze various game development roles and responsibilities.
Experiment 2.8. Create a Unity project using asset store resources.
Experiment 2.9. Collaborate on a Unity project using source control.
Experiment 2.10. Document the game production process.

Module III: The Game Engine User Interface (2+2 hours)

- Introduction to the Unity Editor Interface
- Analyzing the Unity Editor User Interface
- Utilizing the Unity Editor User Interface
- Navigating the Scene View Window
- Utilizing the Game View Window
- Navigating the Hierarchy Window
- Using the Inspector Window
- Managing Assets in the Project Window
- Searching and Filtering in the Project Window
- Organizing the Scene with Layers

Experiments:

Experiment 3.1. Explore and navigate the Unity Editor interface.
Experiment 3.2. Create and modify scenes in Unity.
Experiment 3.3. Organize assets in the Project Window.
Experiment 3.4. Use the Inspector Window to edit game objects.
Experiment 3.5. Implement layers to organize the scene.
Experiment 3.6. Search and filter assets in the Project Window.
Experiment 3.7. Navigate and use the Scene View and Game View windows.
Experiment 3.8. Create a simple game scene in Unity.
Experiment 3.9. Analyze the different windows and their functions in Unity.
Experiment 3.10. Document the navigation and organization of Unity projects.

Module IV: Using Game Objects and Assets (1+2 hours)

- Creating and Modifying Game Objects
- Defining Unity Editor Units
- Describing Assets in the Production Pipeline
- Review: Defining an Asset
- Organizing Assets in the Unity Editor
- Defining a Game Object

Experiments:

Experiment 4.1. Create and modify basic game objects in Unity.
Experiment 4.2. Define and organize assets in the Unity Editor.
Experiment 4.3. Develop a simple game using predefined assets.
Experiment 4.4. Explore the production pipeline of assets.
Experiment 4.5. Implement and manage game objects in a Unity project.
Experiment 4.6. Document the creation and modification of game objects.
Experiment 4.7. Review and organize assets for a Unity project.
Experiment 4.8. Define and utilize Unity Editor units.
Experiment 4.9. Create a game prototype using organized assets.
Experiment 4.10. Present the organization and implementation of game objects and assets.

Module V: Defining a Game Object (2+2 hours)

- Creating Unity-native Game Objects
- Manipulating Game Objects in the Unity Editor
- Describing What is a Unity-native gameObject
- The Role of Components in the Unity Editor
- Defining Prefabs and Scene Structure
- Defining the Role of the Prefab in Unity
- Creating and Saving a Scene

Experiments:

Experiment 5.1. Create and manipulate Unity-native game objects.
Experiment 5.2. Define and use components in the Unity Editor.
Experiment 5.3. Create and manage prefabs in Unity.
Experiment 5.4. Develop a scene structure using prefabs.
Experiment 5.5. Save and organize scenes in a Unity project.
Experiment 5.6. Explore the role of components in game object manipulation.
Experiment 5.7. Create a game scene using Unity-native objects.
Experiment 5.8. Document the creation and manipulation of game objects.
Experiment 5.9. Implement and manage prefabs in a Unity project.
Experiment 5.10. Present a complete game scene with organized prefabs and components.

Module VI: The Hierarchy of Scenes within a Game (1+2 hours)

- Importing Assets into a Project
- Importing and Configuring a 3D Model
- Importing Textures for Use in Materials
- Importing FBX Files with Animation
- Working with Sprites
- Introduction to Sprites in Game Development

Experiments:

Experiment 6.1. Import and configure 3D models in Unity.

Experiment 6.2. Import and apply textures to materials.
Experiment 6.3. Work with FBX files and animations in Unity.
Experiment 6.4. Create and use sprites in a game project.
Experiment 6.5. Explore the hierarchy of scenes within a Unity project.
Experiment 6.6. Document the import and configuration of assets.
Experiment 6.7. Develop a game scene using imported 3D models and textures.
Experiment 6.8. Implement animations in a Unity project.
Experiment 6.9. Create a game prototype using sprites.
Experiment 6.10. Present a game project with organized scenes and imported assets.

Module VII: Managing Projects and Assets (1+2 hours)

- Project Management in Unity
- Introduction to Game Project Management
- Managing Assets
- Using the Unity Asset Store (Reprise)
- Importing Offline Content
- Creating Project Structure Based on Assets
- Sorting the Zombie Toys Prop Model Assets
- Setting Resolution and Type of Texture Files

Experiments:

Experiment 7.1. Implement project management techniques in Unity.
Experiment 7.2. Manage and organize assets for a game project.
Experiment 7.3. Import and use offline content in Unity.
Experiment 7.4. Develop a project structure based on assets.
Experiment 7.5. Sort and manage prop model assets in Unity.
Experiment 7.6. Set resolution and texture file types for a game project.
Experiment 7.7. Document project management practices in Unity.
Experiment 7.8. Create a game project using organized assets and project structure.
Experiment 7.9. Implement asset management techniques in a Unity project.
Experiment 7.10. Present a well-managed and organized game project.

Textbooks and References:

- "Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
- "Game Engine Architecture" by Jason Gregory
- Unity Documentation and Tutorials

Game Assets and Objects(70hours)

Course Code	Course Title	Credits	Type (T+P+Pj)

CUGI1082	Game Assets and Objects	3	1+1+1

Course Description:

This course focuses on the creation, management, and implementation of game assets and objects within Unity. Students will learn best practices for 3D content creation, material management, lighting, and physics to build interactive and visually appealing game levels.

Course Objectives:

1. Master the creation and implementation of game assets and objects.
2. Understand and utilize Unity's material and lighting systems.
3. Learn to design and assemble game levels using physics and lighting techniques.

Course Outcomes (COs):

1. CO1: Demonstrate proficiency in 3D content creation, modeling, and texturing for game assets. (Apply, Create)
2. CO2: Utilize Unity's material and shader systems effectively. (Apply, Analyze)
3. CO3: Design and assemble interactive game levels with proper physics and lighting. (Apply, Create)
4. CO4: Implement colliders and physics components to enhance game interactivity. (Apply, Analyze)
5. CO5: Apply lighting techniques to create desired visual effects and moods in game scenes. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	2	2	1	2	-	-	1	1	-	-	1	3	2	3
CO2	3	3	3	2	3	-	1	2	1	1	-	2	3	3	3
CO3	3	3	3	3	3	1	1	2	2	1	2	2	3	3	3
CO4	3	3	3	3	3	1	1	2	3	1	3	3	3	3	3
CO5	3	3	3	3	3	1	1	2	2	3	2	2	3	3	3

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

Course Syllabus:

Module I: Preparing Assets for Implementation (1+2.5 hours)

- Best Practices in 3D Content Creation
- Modeling for Games
- Animating for Games
- UV Mapping and Texturing Techniques
- Exporting to Unity
- Importing into Unity
- Materials in Unity
- The Interaction of Lighting and Materials

Experiments:

Experiment 1.1. Create a 3D model suitable for a game.
Experiment 1.2. Animate the 3D model for in-game use.
Experiment 1.3. Apply UV mapping and texturing to a 3D model.
Experiment 1.4. Export the textured model to Unity.
Experiment 1.5. Import the model into Unity and apply materials.
Experiment 1.6. Adjust material properties to interact with lighting.
Experiment 1.7. Create various materials for game assets.
Experiment 1.8. Compare different texturing techniques.
Experiment 1.9. Implement best practices in 3D content creation.
Experiment 1.10. Document the asset creation and import process.

Module II: Discovering the Standard Shader in Unity (1+2 hours)

- Exploring other Material Types
- Analyzing the Benefits of Custom Shaders
- Creating the Materials for Zombie Toys Props
- Duplicating and Modifying Materials
- Case Studies in Material Creation
- Managing and Using Textures in the Unity Editor
- Texturing for Game Development
- Optimization and Reuse of Textures

Experiments:

Experiment 2.1. Explore and use Unity's standard shader.
Experiment 2.2. Create and apply custom shaders.
Experiment 2.3. Duplicate and modify existing materials for different effects.
Experiment 2.4. Analyze case studies of effective material creation.
Experiment 2.5. Manage textures in the Unity Editor.
Experiment 2.6. Optimize textures for performance.
Experiment 2.7. Create materials for specific game props.
Experiment 2.8. Apply different textures to game objects.
Experiment 2.9. Reuse textures efficiently across multiple assets.
Experiment 2.10. Document shader and material modifications.

Module III: Assembling the Game Level (2+2 hours)

- Branching and Hierarchies
- Creating Hierarchies in Unity
- Using Empty Game Objects as Pivots
- Introduction to Physics in Unity
- Understanding the Physics System in Unity
- Introduction to the Rigidbody Component

Experiments:

Experiment 3.1. Create hierarchies for organizing game objects.
Experiment 3.2. Use empty game objects as pivots for grouping.
Experiment 3.3. Implement basic physics in Unity.
Experiment 3.4. Apply Rigidbody components to game objects.
Experiment 3.5. Create a simple game level with organized hierarchies.
Experiment 3.6. Explore the Unity physics system.
Experiment 3.7. Apply physics properties to game objects.
Experiment 3.8. Develop a game prototype with physics-based interactions.
Experiment 3.9. Adjust Rigidbody settings for different effects.
Experiment 3.10. Document the game level assembly process.

Module IV: Introduction to Colliders (1+2 hours)

- Creating the Colliders for Zombie Toys Props
- Introduction to Game Level Design
- The Level Design in Zombie Toys
- Placing Objects in a Scene
- Importing the Prop Prefabs into the Scene
- Cloning the Stars
- Creating the Level Boundaries

Experiments:

Experiment 4.1. Create colliders for various game objects.
Experiment 4.2. Design a basic game level layout.
Experiment 4.3. Place objects strategically in a scene.
Experiment 4.4. Import and utilize prop prefabs in a game level.
Experiment 4.5. Clone objects to populate the game level.
Experiment 4.6. Create boundaries for the game level.
Experiment 4.7. Test and adjust colliders for desired interactions.
Experiment 4.8. Design a game level for a specific gameplay experience.
Experiment 4.9. Integrate colliders into the level design.
Experiment 4.10. Document the collider implementation and level design.

Module V: Lighting in Games (1+2 hours)

- Introduction to Game Lighting
- Differences in Lighting for Games and for Film
- Placing and Adjusting Lights in a Scene

- Analyzing the Different Lights and Properties
- Light Types and Behaviors
- Using Layers to Exclude Objects from Lighting
- Casting and Modifying Shadows
- Mesh Renderer Attributes for Shadows

Experiments:

Experiment 5.1. Implement basic lighting in a game scene.
Experiment 5.2. Place and adjust various types of lights.
Experiment 5.3. Analyze the properties of different light types.
Experiment 5.4. Use layers to manage lighting effects.
Experiment 5.5. Cast and modify shadows in a game scene.
Experiment 5.6. Adjust mesh renderer attributes for shadow effects.
Experiment 5.7. Create a lighting setup for a game level.
Experiment 5.8. Compare lighting techniques for different moods.
Experiment 5.9. Optimize lighting for performance.
Experiment 5.10. Document the lighting setup and adjustments.

Module VI: Differentiating Shadow Types (1+2 hours)

- Creating Cookies to Shape Lights
- Faking Shadows for Better Performance
- Benefits of Faking Shadows in Games
- Utilizing Painted Shadows
- Using Projectors to Project Shadow Cookies
- Lighting the Zombie Toys Game
- Lighting the Zombie Toys Scene
- Lighting Variations for Changing the Mood

Experiments:

Experiment 6.1. Create light cookies to shape lighting effects.
Experiment 6.2. Implement fake shadows for performance optimization.
Experiment 6.3. Utilize painted shadows in a game scene.
Experiment 6.4. Use projectors to create shadow effects.
Experiment 6.5. Apply different lighting setups to change the mood.
Experiment 6.6. Test and compare various shadow techniques.
Experiment 6.7. Optimize shadows for performance.
Experiment 6.8. Document the creation and application of light cookies.
Experiment 6.9. Develop a game scene with varying lighting effects.
Experiment 6.10. Present a game level with optimized shadows and lighting.

Module VII: Baking Lighting in Game Production (1+2 hours)

- Light Baking in Video Games
- Setting Objects to Participate in Light Baking
- Marking Objects as Static for Light Baking

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGI1083	Building Game Environment	3	1+1+1

- Creating UV Coordinates for Light Baking
- Baking Lighting
- Continuous and Manual Light Baking
- Placing Light Probes for Moving Objects
- Creating Reflection Probes
- Baking the Lighting in Zombie Toys
- Creating the Light Probes in Zombie Toys

Experiments:

Experiment 7.1. Implement light baking in a game scene.
Experiment 7.2. Set objects to participate in light baking.
Experiment 7.3. Mark objects as static for efficient light baking.
Experiment 7.4. Create UV coordinates for light baking.
Experiment 7.5. Perform continuous and manual light baking.
Experiment 7.6. Place light probes for dynamic objects.
Experiment 7.7. Create and configure reflection probes.
Experiment 7.8. Test and adjust baked lighting for desired effects.
Experiment 7.9. Optimize light baking for performance.
Experiment 7.10. Document the light baking process and results.

Textbooks and References:

- "Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
- "Game Engine Architecture" by Jason Gregory
- Unity Documentation and Tutorials

Building Game Environment(70hours)

Course Description:

This course provides an in-depth understanding of creating game environments using Unity. Students will learn how to develop player characters, allies, enemies, and integrate particle systems and audio to build immersive game environments.

Course Objectives:

1. Develop player characters, allies, and enemies in Unity.
2. Create and integrate particle systems for game effects.
3. Implement audio to enhance game environments.

Course Outcomes (COs):

1. CO1: Create and configure player and ally characters with appropriate behaviors. (Apply, Create)
2. CO2: Design and implement various enemy characters and their behaviors. (Apply, Analyze)
3. CO3: Utilize Unity's particle system to create and integrate game effects. (Apply, Create)
4. CO4: Develop and integrate audio effects to enhance the gaming experience. (Apply, Analyze)
5. CO5: Apply best practices in game development to create interactive and engaging game environments. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	3	3	2	2	-	-	-	2	2	2	2	3	3	3
CO2	3	3	3	2	2	-	-	-	2	2	2	2	3	3	3
CO3	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3
CO4	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3
CO5	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3

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

Course Syllabus:

Module I: Building the Player and Allies (2+2 hours)

- Creating a Player Controller
- Examining Why to Use a Custom Controller
- Creating the Player Controller Game Object
- Adding a Game Manager
- Explaining the Purpose of the Game Manager
- Making the Controller Functional
- Adding Scripts for Behavior
- Configuring the Camera
- Creating the Sheep Ally
- Building the Sheep Ally From a Model
- Creating the Dog Ally
- Building the Dog Ally From a Model

Experiments:

Experiment 1.1. Create and configure a player controller.
Experiment 1.2. Develop and integrate a game manager.
Experiment 1.3. Script behaviors for the player controller.
Experiment 1.4. Configure the camera for optimal gameplay.
Experiment 1.5. Create and implement the sheep ally character.
Experiment 1.6. Develop and integrate the dog ally character.
Experiment 1.7. Test and refine ally behaviors.
Experiment 1.8. Document the player and ally creation process.
Experiment 1.9. Analyze the impact of different player controls.
Experiment 1.10. Present a functional player and ally setup.

Module II: Building the Enemies (2+2 hours)

- Creating an Enemy
- Designing the Enemy Behaviors
- Creating the First Enemy Character
- Creating the Enemy Animator Controller
- Creating Additional Enemies
- Creating the Zombear Enemy
- Creating the Zombie Duck Enemy
- Creating the Other Enemies
- Integrating Enemies into the Game
- Placing the Spawn Points
- Spawning the Enemies

Experiments:

Experiment 2.1. Create an enemy character with appropriate animations.
Experiment 2.2. Design and implement enemy behaviors.
Experiment 2.3. Develop additional enemy characters (Zombear, Zombie Duck).
Experiment 2.4. Integrate enemies into the game environment.
Experiment 2.5. Set up and test spawn points for enemies.
Experiment 2.6. Script enemy behaviors and interactions.
Experiment 2.7. Create an enemy animator controller.
Experiment 2.8. Test and refine enemy spawning mechanics.
Experiment 2.9. Document the enemy creation and integration process.
Experiment 2.10. Analyze the impact of different enemy behaviors on gameplay.

Module III: Introduction to Unity's Particle System (1+1 hours)

- Analyzing Existing Particle Effects
- Setting Up the Interface for Effects
- Case Study: Developing the Lightning Attack

- Overview of the Lightning Attack
- Building the Lightning Attack Hit
- Building the Lightning Attack Emitter
- Building the Lightning Bolt
- Integrating the Lightning Attack into the Game

Experiments:

Experiment 3.1. Analyze existing particle effects in Unity.
Experiment 3.2. Set up the particle system interface for effect creation.
Experiment 3.3. Develop and integrate a lightning attack effect.
Experiment 3.4. Create and configure particle emitters.
Experiment 3.5. Test and refine the lightning attack in-game.
Experiment 3.6. Document the particle system creation process.
Experiment 3.7. Compare different particle effects for impact.
Experiment 3.8. Present a functional particle effect in-game.
Experiment 3.9. Analyze the performance impact of particle effects.
Experiment 3.10. Optimize particle systems for better performance.

Module IV: Creating Particle Systems (1+2 hours)

- Introduction to Particle Systems in the Unity Editor
- Examples of Unity Particles in Video Games
- The Role of the Effects Artist in Video Games
- Comparing Game Effects with Other Media
- Production Best Practices for Particle Systems

Experiments:

Experiment 4.1. Create basic particle systems in Unity.
Experiment 4.2. Implement particle systems in a game scene.
Experiment 4.3. Analyze examples of particle effects in video games.
Experiment 4.4. Compare particle effects in games with other media.
Experiment 4.5. Apply best practices for particle system production.
Experiment 4.6. Develop complex particle systems for specific effects.
Experiment 4.7. Test and refine particle systems in-game.
Experiment 4.8. Document the particle system creation and integration process.
Experiment 4.9. Present particle systems with different visual effects.
Experiment 4.10. Optimize particle systems for performance.

Module V: Case Study (1+2 hours)

- Developing the Frost Attack
  - Introduction to the Frost Attack
  - Building the Frost Debuff

- Building the Frost Attack Emitter
- Building the Frost Cone Effect
- Integrating the Frost Attack into the Game

- Case Study: Developing the Stink Bomb Attack
  - Introduction to the Stink Bomb Attack
  - Creating the Stink Bomb Hit Effect

Experiments:

Experiment 5.1. Develop and integrate a frost attack effect.
Experiment 5.2. Create and configure a frost debuff and emitter.
Experiment 5.3. Test and refine the frost attack in-game.
Experiment 5.4. Develop and integrate a stink bomb attack effect.
Experiment 5.5. Create and configure the stink bomb hit effect.
Experiment 5.6. Test and refine the stink bomb attack in-game.
Experiment 5.7. Document the development of attack effects.
Experiment 5.8. Analyze the impact of different attack effects on gameplay.
Experiment 5.9. Compare different attack effects for visual and functional impact.
Experiment 5.10. Present functional attack effects in-game.

Module VI: Case Study (1+2 hours)

- Developing the Slime Attack
  - Introduction to the Slime Attack
  - Creating the Slime Hit Effect
  - Creating the Slime Debuff
  - Creating the Slime Attack Reticle
  - Building the Slime Attack Emitter
  - Building the Slime Projectile
  - Integrating the Slime Attack into the Game
  - Finalizing Player Attacks
  - Adding the Ally Manager

Experiments:

Experiment 6.1. Develop and integrate a slime attack effect.
Experiment 6.2. Create and configure a slime hit effect and debuff.
Experiment 6.3. Build and test the slime attack emitter and projectile.
Experiment 6.4. Integrate the slime attack into the game environment.
Experiment 6.5. Finalize and refine player attacks.
Experiment 6.6. Develop and integrate an ally manager.
Experiment 6.7. Document the development of the slime attack.
Experiment 6.8. Compare different attack effects for impact.
Experiment 6.9. Present a functional slime attack in-game.

Experiment 6.10. Optimize attack effects for performance.

Module VII: Adding Audio to Game Levels (1+2.5 hours)

- Introduction to Audio in Game Development
- Importing Audio into Unity
    - Supported Audio Formats in Unity
    - Playing Audio in the Unity Editor
    - Testing Audio Sources in the Scene
- Mixing Audio in Unity
    - Using Audio Mixers and Audio Mixer Groups
    - Setting up the Zombie Toys Audio Mixers
    - Creating Audio Effects

Experiments:

Experiment 7.1. Import audio files into Unity.
Experiment 7.2. Play and test audio sources in a game scene.
Experiment 7.3. Create and configure audio mixers.
Experiment 7.4. Implement and test audio effects.
Experiment 7.5. Develop audio effects for specific game scenarios.
Experiment 7.6. Document the audio integration process.
Experiment 7.7. Compare different audio effects for impact.
Experiment 7.8. Present functional audio effects in-game.
Experiment 7.9. Optimize audio effects for performance.
Experiment 7.10. Analyze the impact of audio on the gaming experience.

Textbooks and References:

- "Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
- "Game Engine Architecture" by Jason Gregory
- Unity Documentation and Tutorials

Game Animation, Scripting & UI (70hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGI1084	Game Animation, Scripting & UI	3	1+1+1

Course Description:

This course focuses on animating game objects, scripting for game development, and designing user interfaces using the Unity editor. Students will learn how to create and integrate animations, write scripts to control game behavior, and develop user-friendly interfaces.

Course Objectives:

1. Develop and integrate animations for game objects in Unity.
2. Write and implement scripts to control game behavior and interactions.
3. Design and create user interfaces for an engaging game experience.

Course Outcomes (COs):

1. CO1: Create and refine animations for game objects using the Unity editor. (Apply, Create)
2. CO2: Import and integrate animated characters and rigs into the game environment. (Apply, Analyze)
3. CO3: Develop and control animations using AnimationClips and Animator Controllers. (Apply, Create)
4. CO4: Write and implement scripts for game mechanics and interactions. (Apply, Create)
5. CO5: Design and develop user interfaces for games using Unity's UI tools. (Apply, Create)

CO-PO-PSO Mapping:

CO/PO/ PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	3	3	2	2	-	-	-	2	2	2	2	3	3	3
CO2	3	3	3	2	2	-	-	-	2	2	2	2	3	3	3
CO3	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3
CO4	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3
CO5	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3

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

Course Syllabus:

Module I: Animating Game Objects in the Unity Editor (1+2 hours)

- Introduction to Animation in Game Development
- Animating in the Unity Editor
- Creating and Refining Animations in the Unity Editor

Experiments:

Experiment 1.1. Create basic animations for game objects.
Experiment 1.2. Refine animations using the Unity editor.
Experiment 1.3. Implement smooth transitions between animations.
Experiment 1.4. Test and optimize animations for performance.
Experiment 1.5. Document the animation creation process.

Module II: Bringing Animation into the Game (1+2 hours)

- Importing Animated Characters
- Introduction to Rigging and Imported Animation
- Recognizing Asset Data when Importing
- Differentiating Available Rig Animation Types

Experiments:

Experiment 2.1. Import animated characters into Unity.
Experiment 2.2. Set up rigs and animations for imported characters.
Experiment 2.3. Test and refine imported animations.
Experiment 2.4. Document the character import and setup process.
Experiment 2.5. Analyze the impact of different animation types on gameplay.

Module III: Animation Creation and Controlling (1+2 hours)

- Creating and Naming AnimationClips
- Creating an Animator Controller
- Creating and Modifying Animation States
- Creating Parameters to Control Transitions
- Creating an Animator Override Controller

Experiments:

Experiment 3.1. Create and name AnimationClips.
Experiment 3.2. Develop and configure an Animator Controller.
Experiment 3.3. Create and modify animation states.
Experiment 3.4. Implement parameters to control animation transitions.
Experiment 3.5. Test and refine the Animator Controller.

Experiment 3.6. Document the animation control setup process.
Experiment 3.7. Compare different animation control methods.
Experiment 3.8. Present a functional animation control system.

Module IV: Scripting in Game Development (2+2.5 hours)

- Intro to Scripting in Game Development
- Creating Scripts in Unity
- Creating and Saving a Script in Unity
- Analyzing the Default Script Methods
- Scripting Primer and Best Practices
- Attaching a Script to a Game Object
- Declaring Variables
- Creating Conditions in Scripting
- Looping

Experiments:

Experiment 1.1. Create and save scripts in Unity.
Experiment 4.2. Analyze and utilize default script methods.
Experiment 4.3. Attach scripts to game objects.
Experiment 4.4. Declare and use variables in scripts.
Experiment 4.5. Implement conditions and loops in scripts.
Experiment 4.6. Develop complex scripting conditions.
Experiment 4.7. Test and debug scripts for functionality.
Experiment 4.8. Document the scripting process.
Experiment 4.9. Present functional scripts controlling game mechanics.

Module V: Designing User Interfaces for Games (1+2 hours)

- Introduction to Designing the User Interface
- Assessing User Interface Design Needs
- Examining the UI Tools in the Unity Editor
- Creating a User Interface
- Investigating the Canvas Functionality
- Utilizing the Power of the Rect Transform
- Creating UI Elements (Button, Image, Text)
- Creating Interaction in the UI with Events

Experiments:

Experiment 5.1. Design and create a basic user interface.
Experiment 5.2. Implement UI elements (buttons, images, text).
Experiment 5.3. Test and refine UI interactions.
Experiment 5.4. Document the UI design process.

Experiment 5.5. Analyze the impact of different UI designs on user experience.

Module VI: Introduction to Looping (1+2 hours)

- The “while” Loop
- The “for” Loop
- Creating Custom Methods
- Utilizing Arguments and Method Return Types
- Coroutines
- Accessing Components via Script
- Utilizing the GetComponent() Function
- Common Code Cases in Game Development

Experiments:

Experiment 6.1. Implement while and for loops in scripts.
Experiment 6.2. Create and utilize custom methods.
Experiment 6.3. Develop and test coroutines.
Experiment 6.4. Access and manipulate components via scripts.
Experiment 6.5. Document the looping and method creation process.
Experiment 6.6. Present functional scripts using loops and methods.

Module VII: Implementing Navigation and Path Finding (1+2 hours)

- Introduction to Navigation and Path Finding
- Describing a NavMesh
- Defining a NavMesh Agent
- Describing a NavMesh Obstacle

Experiments:

Experiment 7.1. Set up navigation and pathfinding in Unity.
Experiment 7.2. Create and configure a NavMesh.
Experiment 7.3. Develop and test NavMesh agents and obstacles.
Experiment 7.4. Document the navigation setup process.
Experiment 7.5. Analyze the impact of navigation and pathfinding on gameplay.

Textbooks and References:

- "Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
- "Game Engine Architecture" by Jason Gregory
- Unity Documentation and Tutorials

Binary Deployment and Cross-Platform Controls(70hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGI1085	Binary Deployment and Cross-Platform Controls	3	1+1+1

Course Description:

This course covers the processes of building, deploying, and optimizing games for multiple platforms using Unity. Students will learn about camera systems, player selection, animations, and handling cross-platform inputs for mobile, WebGL, VR, and other Unity-supported platforms.

Course Objectives:

1. Develop camera and player selection systems for games.
2. Understand and implement cross-platform controls and inputs.
3. Learn the processes of building and deploying games for multiple platforms.

Course Outcomes (COs):

1. CO1: Implement and configure camera and player selection systems in Unity. (Apply, Create)
2. CO2: Create and manage multiple player options and integrate them into the game environment. (Apply, Create)
3. CO3: Develop and apply camera animations and behaviors for enhanced game interactions. (Apply, Create)
4. CO4: Build and deploy games for various platforms, ensuring compatibility and performance. (Apply, Create)
5. CO5: Implement cross-platform inputs and controls for mobile, WebGL, VR, and other platforms. (Apply, Analyze)

CO-PO-PSO Mapping:

CO/PO/ PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO1	PSO2	PSO3
CO1	3	3	3	2	2	-	-	-	2	2	2	2	3	3	3
CO2	3	3	3	2	2	-	-	-	2	2	2	2	3	3	3
CO3	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3
CO4	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3

CO5	3	3	3	3	2	-	-	-	3	2	2	2	3	3	3

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

Course Syllabus:

Module I: Building the Camera and Player Selection System (1+3 hours)

- Intro to the Camera and Player Selection Behaviors
- Analyzing the Player Selection System

Experiments:

Experiment 1.1. Set up a basic camera system in Unity.
Experiment 1.2. Implement player selection mechanics.
Experiment 1.3. Test and refine camera and player selection behaviors.
Experiment 1.4. Document the camera and player selection setup process.
Experiment 1.5. Present a functional camera and player selection system.

Module II: Creating Another Player Option (2+2.5 hours)

- Making the Player Selectable
- Adding Another Player
- Finalizing the Camera

Experiments:

Experiment 2.1. Add a new player character to the game.
Experiment 2.2. Implement selection functionality for the new player.
Experiment 2.3. Finalize and test the camera integration.
Experiment 2.4. Document the player addition and camera setup process.
Experiment 2.5. Analyze the impact of multiple player options on gameplay.

Module III: Adding Camera Animations (2+2 hours)

- Configuring the Camera Animator Controller
- Applying Behavior to the Camera
- Adding Character Selection Spotlights

Experiments:

Experiment 3.1. Configure and test the Camera Animator Controller.
Experiment 3.2. Develop and integrate camera animations.

Experiment 3.3. Add and test character selection spotlights.
Experiment 3.4. Document the camera animation process.
Experiment 3.5. Present a functional camera animation system.

Module IV: Building and Deploying the Game (1+1 hours)

- Introduction to the Build Process
- Adjusting the Player Settings
- Building the Game

Experiments:

Experiment 4.1. Prepare the game for building and deployment.
Experiment 4.2. Adjust player settings for optimal performance.
Experiment 4.3. Build and test the game for deployment.
Experiment 4.4. Document the build and deployment process.
Experiment 4.5. Present a deployed game build.

Module V: Protecting Your Creation (1 hour)

- Legal Considerations for Your Game
- Unity Services
- Unlocking the Unity Platform Potential
- Surveying Unity Services

Experiments:

Experiment 5.1. Explore legal considerations for game development.
Experiment 5.2. Utilize Unity services to enhance game functionality.
Experiment 5.3. Document the use of Unity services.
Experiment 5.4. Analyze the impact of legal considerations on game development.

Module VI: Understanding Cross-Platform Inputs (1+2 hours)

- Different Input Types (Mobile, WebGL, OpenVR, etc.)
- Implementing Cross-Platform Inputs

Experiments:

Experiment 6.1. Implement and test mobile inputs.
Experiment 6.2. Implement and test WebGL inputs.
Experiment 6.3. Implement and test VR inputs.
Experiment 6.4. Document the cross-platform input setup process.
Experiment 6.5. Present a functional cross-platform input system.

Module VII: Preparing for Mobile Deployment (2+2 hours)

- Modifying Zombie Toys for Mobile
- Introduction to Mobile Development in Unity
- Changing the Build Platform to Mobile
- Adding the Mobile Interface UI
- Implementing Mobile Input Behaviors

Experiments:

Experiment 7.1. Modify the game for mobile deployment.
Experiment 7.2. Change the build platform to mobile.
Experiment 7.3. Add and test the mobile interface UI.
Experiment 7.4. Implement and test mobile input behaviors.
Experiment 7.5. Document the mobile deployment process.
Experiment 7.6. Present a deployed mobile game build.

Textbooks and References:

- "Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
- "Game Engine Architecture" by Jason Gregory
- Unity Documentation and Tutorials

Project(168hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUGI1086	Project	6	0+0+6

Course Description:

The Capstone Project course allows students to apply their practical skills and knowledge acquired throughout the program to develop a comprehensive, real-world gaming or AR/VR application. This involves planning, designing, developing, testing, and presenting a fully functional project.

Course Objectives:

1. Develop advanced technical skills in gaming and AR/VR development.
2. Apply best practices and techniques in the design and development of interactive applications.
3. Demonstrate proficiency in project planning, execution, and presentation.

Course Outcomes (COs):

1. CO1: Formulate a project idea and define objectives, scope, and requirements. (Create, Evaluate)
2. CO2: Design and prototype the user interface and experience. (Create, Apply)
3. CO3: Develop the core features and functionalities of the project. (Create, Apply)
4. CO4: Test and debug the application to ensure functionality and performance. (Analyze, Apply)
5. CO5: Present the completed project effectively through written documentation and oral presentations. (Create, Evaluate)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	3	3	3	2	-	-	-	2	2	2	2	3	3	3
CO2	3	3	3	3	2	-	-	-	2	2	2	2	3	3	3
CO3	3	3	3	3	2	-	-	-	2	2	2	2	3	3	3
CO4	3	3	3	3	2	-	-	-	2	2	2	2	3	3	3
CO5	3	3	3	3	2	-	-	-	2	2	2	2	3	3	3

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

Course Syllabus:

Module 1: Project Proposal and Planning (10 hours)

- Identifying a Project Idea
- Defining Objectives, Scope, and Requirements
- Developing a Project Plan and Timeline
- Reviewing and Finalizing the Project Proposal

Module 2: Design and Prototyping (10 hours)

- Creating Wireframes and Prototypes
- User Interface and Experience Design
- Finalizing Design Specifications
- Preparing Design Documentation

Module 3: Development Phase 1: Core Features (10 hours)

- Setting Up the Project Structure
- Implementing Core Features and Functionalities
- Integrating Key Components
- Ensuring Basic Functionality

Module 4: Development Phase 2: Advanced Features (10 hours)

- Integrating Advanced Features (e.g., AR/VR interactions, animations)
- Backend Integration and API Usage
- Enhancing User Experience
- Iterating Based on Feedback

Module 5: Testing and Quality Assurance (10 hours)

- Conducting Unit, Integration, and User Testing
- Debugging and Resolving Issues
- Ensuring Cross-Platform Compatibility
- Finalizing the Application for Deployment

Module 6: Deployment and Presentation (10 hours)

- Preparing the App for Release
- App Store and Google Play Store Submission Process
- Creating Visual Aids and Supporting Materials for Presentation
- Presenting the Completed Project to an Audience
- Responding to Questions and Feedback

Textbooks and References:

- "The Lean Startup" by Eric Ries
- "Game Development Essentials: An Introduction" by Jeannie Novak
- Unity Documentation and Tutorials
- Relevant Online Resources and Communities

Domain Track: Blockchain Technology (18 Credits) (0+7+11)

Introduction to Blockchain (54 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUBD1091	Introduction to Blockchain	2	0+2+0

Course Description: This course introduces students to the fundamental concepts and tools used in blockchain technology. Topics include blockchain architecture, consensus mechanisms, and applications.

Course Objectives:

- Understand the fundamentals of blockchain technology and its key components.
- Explore various consensus mechanisms and their applications in blockchain.
- Gain knowledge about cryptocurrencies, cryptoeconomics, and blockchain use cases.

Course Outcomes (COs):

1. CO1: Gain a comprehensive understanding of blockchain technology and its evolution. (Understand, Remember)
2. CO2: Develop skills in understanding decentralized systems and distributed ledger technology. (Apply, Analyze)
3. CO3: Build a basic blockchain system with key components and consensus mechanisms. (Apply, Create)
4. CO4: Implement basic cryptographic techniques in a blockchain context. (Apply, Evaluate)
5. CO5: Design a blockchain use case and analyze its benefits and challenges. (Create, Evaluate)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	2	2	2	3	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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 Syllabus:

Module 1: Overview of Blockchain Technology and Cryptography (9 hours)

- **Topics:** Introduction to blockchain and its characteristics; Evolution of blockchain from Bitcoin to smart contracts; Key components: blocks, transactions, decentralization; Basics of cryptographic techniques used in blockchain (hash functions, digital signatures, cryptographic hashing).

Experiments:

Experiment 1.1. Study the evolution of blockchain technology.
Experiment 1.2. Explore the characteristics of blockchain.
Experiment 1.3. Analyze the key components of a blockchain system.
Experiment 1.4. Implement hash functions and cryptographic hashing.
Experiment 1.5. Create and verify digital signatures.
Experiment 1.6. Compare different cryptographic techniques.
Experiment 1.7. Document the basics of cryptography.
Experiment 1.8. Present a seminar on the evolution of blockchain technology.
Experiment 1.9. Research and present the future of blockchain technology.
Experiment 1.10. Investigate the challenges in implementing blockchain systems.

Module 2: Decentralization and Distributed Ledger Technology (9 hours)

- **Topics:** Understanding decentralization and its importance in blockchain; Peer-to-peer networks and their role in decentralized systems; Introduction to distributed ledger technology (DLT); Consensus mechanisms and their role in maintaining a distributed ledger; Permissioned vs. permissionless ledgers.

Experiments:

Experiment 2.1. Study the importance of decentralization.
Experiment 2.2. Analyze peer-to-peer networks.
Experiment 2.3. Explore distributed ledger technology.
Experiment 2.4. Compare consensus mechanisms.
Experiment 2.5. Implement a basic consensus mechanism.
Experiment 2.6. Document and present DLT concepts.
Experiment 2.7. Research permissioned vs. permissionless ledgers.
Experiment 2.8. Present a seminar on the role of consensus mechanisms.
Experiment 2.9. Investigate challenges in maintaining a distributed ledger.

Experiment 2.10. Create a report on decentralized systems.

Module 3: Consensus Mechanisms: Proof of Work, Proof of Stake, and More (9 hours)

- **Topics:** Principles of proof of work (PoW) and the mining process; Energy consumption and scalability issues with PoW; Concepts and principles of proof of stake (PoS); Benefits and limitations of PoS compared to PoW; Brief overview of other consensus mechanisms: Proof of Authority (PoA), Delegated Proof of Stake (DPoS), Practical Byzantine Fault Tolerance (PBFT).

Experiments:

Experiment 3.1. Implement proof of work (PoW).
Experiment 3.2. Study the mining process in PoW.
Experiment 3.3. Analyze energy consumption and scalability issues.
Experiment 3.4. Implement proof of stake (PoS).
Experiment 3.5. Compare PoW and PoS.
Experiment 3.6. Explore other consensus mechanisms.
Experiment 3.7. Document the principles of PoS.
Experiment 3.8. Present a seminar on PoW and PoS.
Experiment 3.9. Research the benefits and limitations of PoS.
Experiment 3.10. Investigate challenges in implementing consensus mechanisms.

Module 4: Cryptocurrency Fundamentals and Economics (9 hours)

- **Topics:** Definition and characteristics of cryptocurrencies; Key cryptocurrencies: Bitcoin, Ethereum, Ripple, etc.; Wallets, addresses, and transactions in cryptocurrencies; Basics of crypto-economics and incentive mechanisms; Mining rewards, transaction fees, and economic implications.

Experiments:

Experiment 4.1. Study the characteristics of cryptocurrencies.
Experiment 4.2. Analyze key cryptocurrencies.
Experiment 4.3. Create and manage cryptocurrency wallets.
Experiment 4.4. Conduct cryptocurrency transactions.
Experiment 4.5. Explore crypto-economics and incentive mechanisms.
Experiment 4.6. Document mining rewards and transaction fees.
Experiment 4.7. Present a seminar on the economic implications of cryptocurrencies.
Experiment 4.8. Research the impact of cryptocurrencies on the economy.
Experiment 4.9. Investigate challenges in managing cryptocurrency transactions.
Experiment 4.10. Create a report on the fundamentals of cryptocurrencies.

Module 5: Use Cases and Applications of Blockchain Technology (9 hours)

- **Topics:** Overview of various use cases and applications of blockchain technology; Examples in finance, supply chain, healthcare, identity management, etc.; Benefits and challenges of implementing blockchain solutions; Designing a blockchain use case and analyzing its feasibility and challenges.

Experiments:

Experiment 5.1. Study various use cases of blockchain technology.
Experiment 5.2. Analyze applications in finance.
Experiment 5.3. Explore blockchain use in supply chain management.
Experiment 5.4. Document blockchain applications in healthcare.
Experiment 5.5. Present a seminar on identity management with blockchain.
Experiment 5.6. Research the benefits of implementing blockchain solutions.
Experiment 5.7. Investigate challenges in designing blockchain use cases.
Experiment 5.8. Create a report on real-world blockchain applications.
Experiment 5.9. Design a blockchain use case.
Experiment 5.10. Analyze the feasibility and challenges of a blockchain use case.

Module 6: Project: Understanding and Implementing a Basic Blockchain System (9 hours)

- **Topics:** Designing the architecture of a basic blockchain system; Implementing core components such as blocks, transactions, and consensus mechanisms; Simulating the blockchain network and validating transactions; Testing and debugging the basic blockchain implementation; Documenting the project with explanations of the blockchain architecture, implementation details, and testing procedures.

Experiments:

Experiment 6.1. Design the architecture of a blockchain system.
Experiment 6.2. Implement blocks and transactions.
Experiment 6.3. Implement consensus mechanisms.
Experiment 6.4. Simulate a blockchain network.
Experiment 6.5. Validate transactions in the blockchain.
Experiment 6.6. Test the blockchain implementation.
Experiment 6.7. Debug the blockchain system.
Experiment 6.8. Document the architecture and implementation details.
Experiment 6.9. Present a seminar on the implemented blockchain system.
Experiment 6.10. Create a final report on the blockchain project.

Textbooks and References:

- "Mastering Blockchain" by Imran Bashir
- "Blockchain Basics" by Daniel Drescher

Cryptocurrencies and Smart Contracts (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUBD1092	Cryptocurrencies and Smart Contracts	3	0+2+1

Course Prerequisites:

- Completion of "Introduction to Blockchain" or equivalent knowledge
- Basic programming skills

Course Objectives:

1. Understand the foundational principles of cryptocurrencies and their economic implications.
2. Learn about the development and deployment of smart contracts.
3. Gain hands-on experience with cryptocurrency transactions and smart contract coding.

Course Outcomes (COs):

1. **CO1:** Develop a deep understanding of various cryptocurrencies and their unique features. (Understand, Apply)
2. **CO2:** Acquire skills to write and deploy smart contracts. (Create, Evaluate)
3. **CO3:** Conduct cryptocurrency transactions and manage wallets. (Apply, Analyze)
4. **CO4:** Implement a simple decentralized application (DApp) using smart contracts. (Create, Apply)
5. **CO5:** Analyze the economic model of a chosen cryptocurrency. (Analyze, Evaluate)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	2	2	1	2	-	-	1	1	-	-	1	3	2	3
CO2	3	3	3	2	3	-	1	2	1	1	-	2	3	3	3
CO3	3	3	3	3	3	1	1	2	2	1	2	2	3	3	3
CO4	3	3	3	3	3	2	1	2	2	1	2	2	3	3	3
CO5	3	3	3	3	3	2	1	2	2	1	2	2	3	3	3

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

Course Syllabus:

Module 1: Introduction to Cryptocurrencies (14 hours)

- **Topics:** Definition and history of cryptocurrencies; Key components of a cryptocurrency system; Overview of major cryptocurrencies: Bitcoin, Ethereum, Litecoin, etc.; How cryptocurrencies differ from traditional currencies.

Experiments:

Experiment 1.1. Study the history and evolution of cryptocurrencies.
Experiment 1.2. Analyze the key components of a cryptocurrency system.
Experiment 1.3. Compare Bitcoin, Ethereum, and Litecoin.
Experiment 1.4. Create a simple cryptocurrency wallet.
Experiment 1.5. Conduct a basic cryptocurrency transaction.
Experiment 1.6. Research the economic implications of cryptocurrencies.
Experiment 1.7. Present a seminar on the differences between cryptocurrencies and traditional currencies.
Experiment 1.8. Investigate the regulatory landscape for cryptocurrencies.
Experiment 1.9. Analyze the impact of cryptocurrencies on global finance.
Experiment 1.10. Document the key features of major cryptocurrencies.

Module 2: Cryptocurrency Transactions and Wallets (14 hours)

- **Topics:** Detailed study of wallets: types, security, and management; How to create and manage cryptocurrency wallets; Understanding cryptocurrency transactions and the blockchain ledger; Practical session on conducting cryptocurrency transactions.

Experiments:

Experiment 2.1. Create and manage different types of cryptocurrency wallets.
Experiment 2.2. Conduct secure cryptocurrency transactions.
Experiment 2.3. Analyze the blockchain ledger for transaction verification.
Experiment 2.4. Study wallet security measures.
Experiment 2.5. Implement multi-signature wallets.
Experiment 2.6. Research the impact of wallet security on cryptocurrency adoption.
Experiment 2.7. Document the process of setting up a cryptocurrency wallet.
Experiment 2.8. Present a seminar on cryptocurrency transaction mechanisms.
Experiment 2.9. Investigate the challenges in managing cryptocurrency wallets.
Experiment 2.10. Create a report on cryptocurrency wallet management.

Module 3: Smart Contracts: Basics and Development (14 hours)

- **Topics:** Introduction to smart contracts: Definition, use cases, and benefits; Smart contract languages: Solidity, Vyper, etc.; Setting up the development environment for

smart contracts; Writing and deploying simple smart contracts on the Ethereum blockchain.

Experiments:

Experiment 3.1. Write a basic smart contract in Solidity.
Experiment 3.2. Deploy a smart contract on the Ethereum blockchain.
Experiment 3.3. Set up a development environment for smart contracts.
Experiment 3.4. Explore the use cases of smart contracts.
Experiment 3.5. Compare Solidity and Vyper.
Experiment 3.6. Document the smart contract development process.
Experiment 3.7. Present a seminar on the benefits of smart contracts.
Experiment 3.8. Research the challenges in deploying smart contracts.
Experiment 3.9. Analyze the impact of smart contracts on various industries.
Experiment 3.10. Create a report on smart contract development and deployment.

Module 4: Advanced Smart Contract Development (14 hours)

- **Topics:** Complex smart contract structures and logic; Testing and debugging smart contracts; Security considerations and best practices for smart contract development; Case studies of significant smart contracts and DApps.

Experiments:

Experiment 4.1. Develop a complex smart contract with business logic.
Experiment 4.2. Test and debug smart contracts.
Experiment 4.3. Implement security measures in smart contracts.
Experiment 4.4. Study case studies of significant smart contracts.
Experiment 4.5. Document the advanced smart contract development process.
Experiment 4.6. Present a seminar on smart contract security best practices.
Experiment 4.7. Research common vulnerabilities in smart contracts.
Experiment 4.8. Analyze the impact of smart contract security on adoption.
Experiment 4.9. Create a report on advanced smart contract structures.
Experiment 4.10. Investigate the challenges in testing and debugging smart contracts.

Module 5: Decentralized Applications (DApps) (14 hours)

- **Topics:** Introduction to DApps: Definition, characteristics, and benefits; Architecture of a DApp: Frontend and backend integration; Tools and frameworks for DApp development (e.g., Truffle, Ganache); Developing a simple DApp and deploying it on the blockchain.

Experiments:

Experiment 5.1. Develop a simple DApp.

Experiment 5.2. Integrate frontend and backend for a DApp.
Experiment 5.3. Use Truffle and Ganache for DApp development.
Experiment 5.4. Document the DApp development process.
Experiment 5.5. Present a seminar on the benefits of DApps.
Experiment 5.6. Research the impact of DApps on various industries.
Experiment 5.7. Analyze the architecture of a successful DApp.
Experiment 5.8. Investigate the challenges in DApp development.
Experiment 5.9. Implement security measures in DApps.
Experiment 5.10. Create a report on DApp development and deployment.

Module 6: Project: Smart Contract and DApp Development (14 hours)

- **Topics:** Project planning and requirement gathering; Designing and implementing a smart contract-based solution; Integrating the smart contract with a frontend interface to create a DApp; Testing, deploying, and presenting the DApp project.

Experiments:

Experiment 6.1. Plan and define requirements for a smart contract-based project.
Experiment 6.2. Design a smart contract solution for the project.
Experiment 6.3. Implement the smart contract and deploy it on Ethereum.
Experiment 6.4. Develop a frontend interface for the DApp.
Experiment 6.5. Integrate the smart contract with the frontend.
Experiment 6.6. Test the DApp for functionality and security.
Experiment 6.7. Debug and resolve any issues in the DApp.
Experiment 6.8. Deploy the DApp to a test network.
Experiment 6.9. Present the completed project to peers.
Experiment 6.10. Document the project development process and outcomes.

Blockchain Development (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUBD1093	Blockchain Development	3	0+1+2

Course Prerequisites:

- Completion of "Cryptocurrencies and Smart Contracts" or equivalent knowledge
- Proficiency in programming and software development

Course Objectives:

- To master the skills required for blockchain application development.
- To understand the architecture and tools used in blockchain development.
- To gain practical experience in developing and deploying blockchain applications.

Course Outcomes (COs):

- **CO1:** Develop proficiency in Solidity and smart contract development. (Apply, Create)
- **CO2:** Build, test, and deploy decentralized applications (DApps). (Create, Evaluate)
- **CO3:** Understand and utilize blockchain development frameworks and tools. (Understand, Apply)
- **CO4:** Implement security best practices in blockchain development. (Apply, Evaluate)
- **CO5:** Develop a full-stack blockchain application with frontend and backend integration. (Create, Apply)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	3	3	2	3	-	-	-	2	2	2	2	3	3	3
CO2	3	2	3	3	3	-	-	-	2	2	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	3	2	2	3	3	3
CO5	3	2	3	3	3	-	-	-	3	3	3	3	3	3	3

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

Course Syllabus:

Module 1: Ethereum and Solidity Programming (14 hours)

- **Topics:** Introduction to Ethereum blockchain; Detailed study of Solidity programming language; Writing, testing, and deploying smart contracts using Solidity; Practical exercises on Solidity basics.

Experiments:

Experiment 1.1. Write a basic smart contract in Solidity.
Experiment 1.2. Deploy a smart contract on the Ethereum blockchain.
Experiment 1.3. Test a smart contract using Remix IDE.
Experiment 1.4. Implement a simple token contract.
Experiment 1.5. Explore the use of events in Solidity.
Experiment 1.6. Study Solidity data types and structures.
Experiment 1.7. Develop a contract with conditional statements.
Experiment 1.8. Integrate Solidity with JavaScript for contract interaction.
Experiment 1.9. Analyze Solidity security features.

Experiment 1.10. Document the process of writing and deploying a smart contract.

Module 2: Development Tools and Environments (14 hours)

- **Topics:** Overview of blockchain development tools: Truffle, Ganache, Remix; Setting up and configuring a development environment; Using MetaMask for blockchain interactions; Practical sessions on using these tools for smart contract development.

Experiments:

Experiment 2.1. Set up a blockchain development environment with Truffle and Ganache.
Experiment 2.2. Create and migrate a smart contract using Truffle.
Experiment 2.3. Interact with a deployed contract using MetaMask.
Experiment 2.4. Debug smart contracts with Remix IDE.
Experiment 2.5. Implement contract testing using Truffle.
Experiment 2.6. Develop a simple DApp frontend using React.
Experiment 2.7. Connect the DApp frontend to smart contracts using Web3.js.
Experiment 2.8. Use Ganache to simulate blockchain transactions.
Experiment 2.9. Document the steps for setting up a blockchain development environment.
Experiment 2.10. Present a seminar on blockchain development tools and their usage.

Module 3: Developing Decentralized Applications (DApps) (14 hours)

- **Topics:** Architecture of DApps: Frontend and backend integration; Building a frontend interface for DApps using JavaScript frameworks (e.g., React); Connecting the frontend to smart contracts using Web3.js; Practical exercises on developing and deploying DApps.

Experiments:

Experiment 3.1. Design the architecture of a DApp.
Experiment 3.2. Implement a DApp frontend using React.
Experiment 3.3. Connect the frontend to smart contracts with Web3.js.
Experiment 3.4. Develop a simple voting DApp.
Experiment 3.5. Integrate MetaMask with the DApp.
Experiment 3.6. Implement state management in the DApp.
Experiment 3.7. Test the DApp on a local blockchain network.
Experiment 3.8. Deploy the DApp on a testnet.
Experiment 3.9. Document the DApp development process.
Experiment 3.10. Present the completed DApp project.

Module 4: Testing and Debugging Blockchain Applications (14 hours)

- **Topics:** Importance of testing in blockchain development; Writing unit tests for smart contracts using Truffle; Debugging techniques and tools; Practical session on testing and debugging a blockchain application.

Experiments:

Experiment 4.1. Write unit tests for smart contracts using Truffle.
Experiment 4.2. Debug a smart contract using Remix IDE.
Experiment 4.3. Implement integration tests for a DApp.
Experiment 4.4. Use Ganache for testing blockchain applications.
Experiment 4.5. Develop a test suite for a smart contract.
Experiment 4.6. Test a DApp frontend with Cypress.
Experiment 4.7. Analyze the results of unit tests and fix bugs.
Experiment 4.8. Document the testing and debugging process.
Experiment 4.9. Present a seminar on the importance of testing in blockchain development.
Experiment 4.10. Review and audit the test cases for a blockchain application.

Module 5: Security and Best Practices in Blockchain Development (14 hours)

- **Topics:** Common security vulnerabilities in smart contracts; Best practices for secure smart contract development; Code reviews and security audits; Case studies of security breaches and lessons learned.

Experiments:

Experiment 5.1. Identify common security vulnerabilities in smart contracts.
Experiment 5.2. Implement security best practices in a smart contract.
Experiment 5.3. Conduct a security audit for a smart contract.
Experiment 5.4. Review code for security issues.
Experiment 5.5. Analyze case studies of security breaches.
Experiment 5.6. Implement access control in smart contracts.
Experiment 5.7. Use static analysis tools for smart contract security.
Experiment 5.8. Document the security measures for a blockchain application.
Experiment 5.9. Present a seminar on smart contract security best practices.
Experiment 5.10. Create a report on the lessons learned from security breaches in blockchain.

Module 6: Project: Full-Stack Blockchain Application Development (14 hours)

- **Topics:** Project planning and requirement analysis; Developing a full-stack blockchain application; Integrating smart contracts with frontend and backend; Testing, deploying, and presenting the blockchain application project.

Experiments:

Experiment 6.1. Plan and define requirements for a blockchain application project.
Experiment 6.2. Design the architecture of a full-stack blockchain application.
Experiment 6.3. Implement the smart contracts for the application.
Experiment 6.4. Develop the frontend interface using React.
Experiment 6.5. Integrate the frontend with the smart contracts.
Experiment 6.6. Develop the backend services for the application.
Experiment 6.7. Test the complete application for functionality and security.
Experiment 6.8. Deploy the application on a blockchain network.
Experiment 6.9. Document the project development process and outcomes.
Experiment 6.10. Present the completed blockchain application project to peers and instructors.

Textbooks and References:

- "Mastering Blockchain" by Imran Bashir
- "Solidity Programming Essentials" by Ritesh Modi
- "Building Ethereum DApps" by Roberto Infante

Web3 and Decentralized Technologies (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUBD1094	Web3 and Decentralized Technologies	3	0+1+2

Course Prerequisites:

- Completion of "Blockchain Development" or equivalent knowledge
- Familiarity with web development concepts

Course Objectives:

- To understand the principles and technologies behind Web3.
- To learn about decentralized finance (DeFi) and other Web3 applications.
- To develop skills in building and interacting with Web3 applications.

Course Outcomes (COs):

- **CO1:** Understand and explain the principles and technologies behind Web3. (Understand, Explain)
- **CO2:** Develop and deploy Web3 applications. (Apply, Create)
- **CO3:** Implement and interact with decentralized finance (DeFi) protocols. (Apply, Analyze)
- **CO4:** Create and manage NFT-based projects. (Create, Manage)
- **CO5:** Explore and implement emerging Web3 technologies. (Apply, Evaluate)

CO-PO-PSO Mapping:

CO/PO/ PSO	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	3	2	2	2	3	-	-	-	2	2	2	2	3	2	3
CO2	3	3	3	2	3	-	-	-	3	2	2	2	3	3	3
CO3	3	3	3	3	3	-	-	-	3	3	3	2	3	3	3
CO4	3	3	3	3	3	-	-	-	3	3	3	2	3	3	3
CO5	3	3	3	3	3	-	-	-	3	3	3	2	3	3	3

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

Course Syllabus:

Module 1: Introduction to Web3 (14 hours)

- **Topics:** Definition and principles of Web3; Differences between Web3 and traditional web technologies; Overview of Web3 architecture and components; Practical session on setting up a Web3 development environment.

Experiments:

Experiment 1.1. Set up a Web3 development environment.
Experiment 1.2. Create a basic Web3 project.
Experiment 1.3. Compare Web3 with traditional web technologies.
Experiment 1.4. Explore the architecture of a Web3 application.
Experiment 1.5. Implement basic blockchain interactions in Web3.
Experiment 1.6. Analyze the components of a Web3 architecture.
Experiment 1.7. Set up a local Ethereum node.
Experiment 1.8. Deploy a simple smart contract.
Experiment 1.9. Document the setup and architecture of a Web3 environment.
Experiment 1.10. Present a seminar on the principles of Web3.

Module 2: Web3.js and Blockchain Interaction (14 hours)

- **Topics:** Introduction to Web3.js library; Connecting to the Ethereum blockchain using Web3.js; Writing scripts to interact with smart contracts and blockchain data; Practical exercises on using Web3.js for blockchain interactions.

Experiments:

Experiment 2.1. Set up Web3.js in a project.
Experiment 2.2. Connect to an Ethereum blockchain using Web3.js.
Experiment 2.3. Write scripts to interact with smart contracts.
Experiment 2.4. Retrieve blockchain data using Web3.js.
Experiment 2.5. Implement a simple DApp using Web3.js.
Experiment 2.6. Document and present a Web3.js interaction script.
Experiment 2.7. Create a user interface for blockchain interaction.
Experiment 2.8. Test Web3.js scripts for functionality.
Experiment 2.9. Analyze the security of Web3.js interactions.
Experiment 2.10. Present a seminar on Web3.js and its capabilities.

Module 3: Decentralized Finance (DeFi) (14 hours)

- **Topics:** Overview of DeFi: Definition, components, and benefits; Key DeFi protocols: Uniswap, Aave, Compound, etc.; Developing a simple DeFi protocol; Practical session on interacting with DeFi protocols.

Experiments:

Experiment 3.1. Study key DeFi protocols.
Experiment 3.2. Implement a basic DeFi protocol.
Experiment 3.3. Interact with Uniswap using Web3.js.
Experiment 3.4. Create a lending protocol using Aave.
Experiment 3.5. Develop a simple DEX (Decentralized Exchange).
Experiment 3.6. Analyze the benefits and challenges of DeFi.
Experiment 3.7. Test a DeFi protocol for functionality and security.
Experiment 3.8. Document the implementation of a DeFi protocol.
Experiment 3.9. Present a seminar on DeFi protocols.
Experiment 3.10. Create a report on the impact of DeFi in the financial sector.

Module 4: NFTs and Token Standards (14 hours)

- **Topics:** Introduction to non-fungible tokens (NFTs); Understanding token standards: ERC20, ERC721, ERC1155; Creating and deploying NFTs on the blockchain; Practical session on developing NFT-based projects.

Experiments:

Experiment 4.1. Study the ERC20, ERC721, and ERC1155 token standards.
Experiment 4.2. Create and deploy an ERC721 NFT.
Experiment 4.3. Develop a marketplace for NFTs.
Experiment 4.4. Implement smart contracts for NFT management.
Experiment 4.5. Analyze the economic model of NFTs.
Experiment 4.6. Document the process of creating and deploying NFTs.
Experiment 4.7. Test NFT transactions on a testnet.

Experiment 4.8. Present a seminar on the impact of NFTs.
Experiment 4.9. Create a report on a successful NFT project.
Experiment 4.10. Develop a simple NFT-based game.

Module 5: Emerging Web3 Technologies (14 hours)

- **Topics:** Overview of emerging Web3 technologies: DAOs, decentralized identity, etc.; Case studies of successful Web3 projects; Exploring the potential and challenges of Web3 adoption; Practical exercises on implementing emerging Web3 technologies.

Experiments:

Experiment 5.1. Study Decentralized Autonomous Organizations (DAOs).
Experiment 5.2. Implement a basic DAO.
Experiment 5.3. Explore decentralized identity solutions.
Experiment 5.4. Develop a project using emerging Web3 technologies.
Experiment 5.5. Analyze case studies of successful Web3 projects.
Experiment 5.6. Document the implementation of a Web3 technology.
Experiment 5.7. Test and deploy a Web3 solution.
Experiment 5.8. Present a seminar on emerging Web3 technologies.
Experiment 5.9. Create a report on the potential of Web3 adoption.
Experiment 5.10. Develop a proposal for a new Web3 project.

Module 6: Project: Web3 and DeFi Application Development (14 hours)

- **Topics:** Project planning and requirement gathering; Developing a Web3 application with blockchain interaction; Implementing a DeFi protocol or an NFT-based project; Testing, deploying, and presenting the Web3 application project.

Experiments:

Experiment 6.1. Plan and define requirements for a Web3 application project.
Experiment 6.2. Design the architecture of a Web3 application.
Experiment 6.3. Implement the smart contracts for the application.
Experiment 6.4. Develop the frontend interface using React.
Experiment 6.5. Integrate the frontend with the smart contracts.
Experiment 6.6. Develop the backend services for the application.
Experiment 6.7. Test the complete application for functionality and security.
Experiment 6.8. Deploy the application on a blockchain network.
Experiment 6.9. Document the project development process and outcomes.
Experiment 6.10. Present the completed Web3 application project to peers and instructors.

Textbooks and References:

- "Mastering Blockchain" by Imran Bashir

- "Solidity Programming Essentials" by Ritesh Modi
- "Building Ethereum DApps" by Roberto Infante

Advanced Blockchain Concepts and Development (84 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUBD1095	Advanced Blockchain Concepts and Development	3	0+1+2

Course Prerequisites:

- Completion of "Web3 and Decentralized Technologies" or equivalent knowledge
- Proficiency in blockchain development and smart contract coding

Course Objectives:

1. To explore advanced concepts and emerging trends in blockchain technology.
2. To develop expertise in scalability, security, and interoperability of blockchain systems.
3. To gain practical experience in implementing advanced blockchain solutions.

Course Outcomes (COs):

1. **CO1:** Master advanced blockchain concepts and techniques. (Analyze, Evaluate)
2. **CO2:** Develop scalable and secure blockchain applications. (Create, Apply)
3. **CO3:** Implement cross-chain communication and interoperability solutions. (Create, Evaluate)
4. **CO4:** Conduct security audits and enhance blockchain security. (Apply, Analyze)
5. **CO5:** Explore and implement emerging blockchain trends and technologies. (Analyze, Create)

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	2	2	2	3	2	3
CO2	3	3	3	2	3	-	-	-	3	2	2	2	3	3	3
CO3	3	3	3	3	3	-	-	-	3	3	3	2	3	3	3
CO4	3	3	3	3	3	-	-	-	3	3	3	2	3	3	3
CO5	3	3	3	3	3	-	-	-	3	3	3	2	3	3	3

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

Course Syllabus:

Module 1: Scalability Solutions in Blockchain (14 hours)

- **Topics:** Challenges of scalability in blockchain networks; Layer 1 solutions: Sharding, improved consensus algorithms; Layer 2 solutions: State channels, sidechains, rollups; Practical session on implementing a scalability solution.

Experiments:

Experiment 1.1. Implement a basic sharding protocol.
Experiment 1.2. Develop a state channel for off-chain transactions.
Experiment 1.3. Configure and deploy a sidechain.
Experiment 1.4. Test scalability improvements using rollups.
Experiment 1.5. Analyze the performance of Layer 1 vs. Layer 2 solutions.
Experiment 1.6. Optimize transaction throughput in a blockchain network.
Experiment 1.7. Compare different consensus algorithms for scalability.
Experiment 1.8. Implement a hybrid consensus mechanism.
Experiment 1.9. Simulate network conditions to test scalability solutions.
Experiment 1.10. Present a case study on a scalable blockchain application.

Module 2: Blockchain Security (14 hours)

- **Topics:** Common security issues in blockchain and smart contracts; Advanced cryptographic techniques for enhancing security; Security auditing and best practices; Practical session on conducting a security audit.

Experiments:

Experiment 2.1. Identify common vulnerabilities in smart contracts.
Experiment 2.2. Implement cryptographic hash functions.
Experiment 2.3. Use digital signatures for transaction security.
Experiment 2.4. Conduct a security audit of a smart contract.
Experiment 2.5. Develop a secure smart contract with input validation.
Experiment 2.6. Implement multi-signature wallets.
Experiment 2.7. Analyze the security of blockchain networks.
Experiment 2.8. Conduct a code review for security flaws.
Experiment 2.9. Simulate a blockchain attack and defense mechanism.
Experiment 2.10. Present a case study on blockchain security breaches.

Module 3: Interoperability and Cross-Chain Communication (14 hours)

- **Topics:** Importance of interoperability in blockchain ecosystems; Protocols for cross-chain communication: Polkadot, Cosmos, etc.; Implementing cross-chain communication solutions; Practical session on developing a cross-chain application.

Experiments:

Experiment 3.1. Implement a basic cross-chain communication protocol.
Experiment 3.2. Use Polkadot's Substrate to create a parachain.
Experiment 3.3. Develop a cross-chain bridge using Cosmos SDK.
Experiment 3.4. Test interoperability between different blockchain networks.
Experiment 3.5. Analyze the performance of cross-chain communication protocols.
Experiment 3.6. Implement token transfers across chains.
Experiment 3.7. Develop a cross-chain decentralized exchange (DEX).
Experiment 3.8. Simulate cross-chain data exchange.
Experiment 3.9. Present a case study on cross-chain applications.
Experiment 3.10. Document the implementation of a cross-chain solution.

Module 4: Privacy in Blockchain (14 hours)

- **Topics:** Privacy challenges in blockchain technology; Techniques for enhancing privacy: Zero-knowledge proofs, ring signatures; Case studies of privacy-focused blockchain projects; Practical session on implementing privacy-enhancing techniques.

Experiments:

Experiment 4.1. Implement zero-knowledge proofs (ZKPs) in a blockchain application.
Experiment 4.2. Use ring signatures for transaction privacy.
Experiment 4.3. Develop a privacy-focused smart contract.
Experiment 4.4. Test the effectiveness of privacy techniques.
Experiment 4.5. Analyze the privacy features of different blockchain platforms.
Experiment 4.6. Implement a private blockchain network.
Experiment 4.7. Simulate private transactions using Monero.
Experiment 4.8. Develop a confidential transaction protocol.
Experiment 4.9. Present a case study on privacy-focused blockchains.
Experiment 4.10. Document privacy enhancements in a blockchain project.

Module 5: Emerging Trends and Technologies in Blockchain (14 hours)

- **Topics:** Overview of emerging trends: DAOs, CBDCs, blockchain in IoT, etc.; Potential impact and future directions of blockchain technology; Exploring new blockchain platforms and protocols; Practical exercises on implementing emerging blockchain technologies.

Experiments:

Experiment 5.1. Develop a Decentralized Autonomous Organization (DAO) smart contract.
Experiment 5.2. Implement a Central Bank Digital Currency (CBDC) prototype.
Experiment 5.3. Explore blockchain integration with IoT devices.
Experiment 5.4. Test blockchain applications in supply chain management.
Experiment 5.5. Analyze the impact of blockchain in healthcare.
Experiment 5.6. Develop a blockchain-based voting system.
Experiment 5.7. Simulate blockchain applications in finance.
Experiment 5.8. Present a case study on emerging blockchain trends.
Experiment 5.9. Implement a new blockchain protocol.
Experiment 5.10. Document the potential impact of emerging technologies.

Module 6: Project: Advanced Blockchain Solution Development (14 hours)

- **Topics:** Project planning and requirement analysis; Developing an advanced blockchain solution with a focus on scalability, security, or interoperability; Testing, deploying, and presenting the advanced blockchain project; Documenting the project with detailed explanations of the advanced concepts and implementation.

Experiments:

Experiment 6.1. Define the project scope and objectives.
Experiment 6.2. Plan and design the architecture of the blockchain solution.
Experiment 6.3. Develop the core components of the blockchain solution.
Experiment 6.4. Integrate scalability solutions in the project.
Experiment 6.5. Implement security enhancements.
Experiment 6.6. Develop cross-chain communication protocols.
Experiment 6.7. Test the complete blockchain solution.
Experiment 6.8. Deploy the solution on a testnet or mainnet.
Experiment 6.9. Document the development and deployment process.
Experiment 6.10. Present the completed project to peers and instructors.

Textbooks and References:

- "Mastering Blockchain" by Imran Bashir
- "Solidity Programming Essentials" by Ritesh Modi
- "Building Ethereum DApps" by Roberto Infante

Capstone Project in Blockchain Development (112 Hours)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUBD1096	Capstone Project in Blockchain Development	4	0+0+4

Course Prerequisites:

- Completion of "Advanced Blockchain Concepts and Development" or equivalent knowledge
- Proficiency in blockchain development and smart contract coding

Course Objectives:

1. To apply theoretical and practical knowledge to solve real-world blockchain problems.
2. To develop a comprehensive blockchain project that showcases advanced development skills.
3. To demonstrate proficiency in blockchain project planning, implementation, and presentation.

Course Outcomes (COs):

1. **CO1:** Develop a comprehensive blockchain project from concept to deployment. (Create, Apply)
2. **CO2:** Implement advanced blockchain techniques and solutions in a real-world scenario. (Analyze, Apply)
3. **CO3:** Demonstrate proficiency in blockchain development tools and frameworks. (Apply, Evaluate)
4. **CO4:** Conduct thorough testing, security audits, and optimizations of a blockchain application. (Evaluate, Analyze)
5. **CO5:** Present and document a blockchain project effectively. (Create, Apply)

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	3	3	-	-	-	2	3	2	2	3	3	3
CO2	3	3	3	3	3	-	-	-	3	3	2	2	3	3	3
CO3	3	3	3	3	3	-	-	-	2	3	3	3	3	3	3
CO4	3	3	3	3	3	-	-	-	3	3	3	3	3	3	3
CO5	3	3	3	3	3	-	-	-	2	3	2	2	3	3	3

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

Course Syllabus:

Module 1: Project Scope and Objectives (10 hours)

- **Topics:** Project Concept; Scope of Work; Blockchain Domain; Project Feasibility.

Experiments:

Experiment 1.1. Define the project concept and problem statement.
Experiment 1.2. Develop a comprehensive project scope and objectives.
Experiment 1.3. Conduct a feasibility study, including resource and time assessment.
Experiment 1.4. Identify the blockchain topics covered by the project.
Experiment 1.5. Present the project proposal for feedback and refinement.
Experiment 1.6. Finalize the project plan and timeline.
Experiment 1.7. Document the project scope and objectives.
Experiment 1.8. Review and refine the project feasibility.
Experiment 1.9. Prepare a detailed project plan.
Experiment 1.10. Conduct a peer review of project proposals.

Module 2: Technology Stack and Tools (10 hours)

- **Topics:** Blockchain Platform; Development Tools; Frontend and Backend Technologies; Database Systems.

Experiments:

Experiment 2.1. Select the appropriate blockchain platform for the project.
Experiment 2.2. Set up development tools and frameworks.
Experiment 2.3. Configure the development environment.
Experiment 2.4. Integrate frontend technologies (e.g., React).
Experiment 2.5. Integrate backend technologies (e.g., Node.js, Express).
Experiment 2.6. Set up database systems and integrate them with the blockchain.
Experiment 2.7. Develop a basic prototype using the selected tools.
Experiment 2.8. Conduct initial tests of the development environment.
Experiment 2.9. Document the technology stack and tools used.
Experiment 2.10. Present a technical overview of the chosen tools and technologies.

Module 3: Project Design and Architecture (10 hours)

- **Topics:** System Architecture; Component Design; Data Flow and Integration.

Experiments:

Experiment 3.1. Design the overall system architecture of the project.

Experiment 3.2. Develop detailed designs for key components.
Experiment 3.3. Create data flow diagrams to illustrate system integration.
Experiment 3.4. Design the smart contract architecture.
Experiment 3.5. Integrate DApps with backend services.
Experiment 3.6. Develop frontend components for user interaction.
Experiment 3.7. Document the system architecture and component design.
Experiment 3.8. Review and refine design documentation.
Experiment 3.9. Present the project design and architecture for feedback.
Experiment 3.10. Conduct a design review session with peers.

Module 4: Smart Contract Development and Deployment (10 hours)

- **Topics:** Smart Contract Functionality; Smart Contract Deployment; Contract Security.

Experiments:

Experiment 4.1. Develop smart contracts based on the project requirements.
Experiment 4.2. Write unit tests for the smart contracts.
Experiment 4.3. Deploy smart contracts to a testnet.
Experiment 4.4. Implement security measures in the smart contracts.
Experiment 4.5. Conduct a security audit of the smart contracts.
Experiment 4.6. Document the smart contract development process.
Experiment 4.7. Review and refine the smart contract code.
Experiment 4.8. Deploy the smart contracts to a mainnet.
Experiment 4.9. Conduct integration tests with the deployed contracts.
Experiment 4.10. Present the smart contract development and deployment process.

Module 5: Decentralized Applications (DApps) (10 hours)

- **Topics:** DApp Design; DApp Interaction; Web3.js Integration.

Experiments:

Experiment 5.1. Design the frontend interface of the DApp.
Experiment 5.2. Develop the backend services for the DApp.
Experiment 5.3. Integrate the frontend and backend components.
Experiment 5.4. Use Web3.js to interact with the blockchain.
Experiment 5.5. Conduct user testing of the DApp interface.
Experiment 5.6. Optimize the user experience of the DApp.
Experiment 5.7. Document the DApp development process.
Experiment 5.8. Review and refine the DApp components.
Experiment 5.9. Present the DApp development and user interaction.
Experiment 5.10. Conduct a usability review session with peers.

Module 6: Security and Privacy Measures (10 hours)

- **Topics:** Security Threats and Mitigation; Smart Contract Security; User Data Privacy; Regulatory Compliance.

Experiments:

Experiment 6.1. Identify and document common security threats.
Experiment 6.2. Implement security measures in smart contracts.
Experiment 6.3. Conduct a security audit of the DApp.
Experiment 6.4. Implement data privacy measures.
Experiment 6.5. Ensure compliance with regulations (e.g., KYC, AML, GDPR).
Experiment 6.6. Document security and privacy measures.
Experiment 6.7. Review and refine security implementations.
Experiment 6.8. Conduct a security and privacy review session.
Experiment 6.9. Present the security and privacy measures of the project.
Experiment 6.10. Conduct a compliance audit of the project.

Module 7: Scalability and Performance Optimization (10 hours)

- **Topics:** Scalability Solutions; Gas Optimization; Performance Testing.

Experiments:

Experiment 7.1. Implement scalability solutions (e.g., state channels, sidechains).
Experiment 7.2. Optimize gas costs for smart contracts.
Experiment 7.3. Conduct performance testing of the DApp.
Experiment 7.4. Document scalability and performance optimization techniques.
Experiment 7.5. Review and refine performance optimization strategies.
Experiment 7.6. Present the scalability and performance optimizations.
Experiment 7.7. Conduct a performance review session with peers.
Experiment 7.8. Implement additional scalability solutions based on feedback.
Experiment 7.9. Test the scalability solutions under different conditions.
Experiment 7.10. Conduct a final performance evaluation of the project.

Module 8: Testing and Quality Assurance (10 hours)

- **Topics:** Unit Testing; Integration Testing; User Testing and Feedback; Code Review and Auditing.

Experiments:

Experiment 8.1. Write unit tests for smart contracts and DApps.
Experiment 8.2. Conduct integration testing of blockchain and non-blockchain components.
Experiment 8.3. Perform user testing and gather feedback.
Experiment 8.4. Conduct a thorough code review.

Experiment 8.5. Perform a security audit.
Experiment 8.6. Document testing and quality assurance procedures.
Experiment 8.7. Review and refine testing methodologies.
Experiment 8.8. Present the testing and quality assurance results.
Experiment 8.9. Conduct a testing review session with peers.
Experiment 8.10. Implement feedback from testing and quality assurance.

Module 9: Project Documentation and Presentation (12 hours)

- **Topics:** Project Documentation; Project Presentation; Feedback and Improvements.

Experiments:

Experiment 9.1. Prepare comprehensive project documentation.
Experiment 9.2. Create a project presentation, including a live demonstration.
Experiment 9.3. Conduct a code walkthrough and explain design choices.
Experiment 9.4. Gather feedback from instructors and peers.
Experiment 9.5. Implement suggested improvements.
Experiment 9.6. Review and refine the project documentation.
Experiment 9.7. Present the final project to a panel of reviewers.
Experiment 9.8. Conduct a final project review session.
Experiment 9.9. Document the feedback and improvements process.
Experiment 9.10. Submit the final project documentation and presentation.

Module 10: Real-World Applications and Use Cases (12 hours)

- **Topics:** Real-World Use Cases; Industry Relevance; Impact and Benefits.

Experiments:

Experiment 10.1. Identify real-world use cases for the project.
Experiment 10.2. Assess the project's relevance to industry trends.
Experiment 10.3. Analyze the potential impact and benefits of the project.
Experiment 10.4. Document the real-world applications and use cases.
Experiment 10.5. Review and refine the industry relevance analysis.
Experiment 10.6. Present the project's real-world applications.
Experiment 10.7. Conduct a case study review session.
Experiment 10.8. Implement feedback on the real-world applications.
Experiment 10.9. Prepare a final report on the project's industry impact.
Experiment 10.10.

Experiment 10.11. Submit the real-world applications and use cases documentation.

Textbooks and References:

- "Mastering Blockchain" by Imran Bashir

- "Solidity Programming Essentials" by Ritesh Modi
- "Building Ethereum DApps" by Roberto Infante

Domain Track: Cybersecurity

Linux Server Management and Security (112 hrs)

Course Code	Course Title	Credits	Type (T+P+Pj)
CUCS1101	Linux Server Management and Security	4	2+2+0

Course Description: This course provides an in-depth understanding of Linux server management and security. Students will learn essential Linux commands, manage users and files, configure networks, and implement security measures.

Course Objectives:

1. Master essential Linux commands and server management tasks.
2. Implement and manage Linux security features.
3. Configure and optimize Linux server performance.

Course Outcomes (COs):

1. CO1: Access and manage the Linux command line for server management. (Apply, Analyze)
2. CO2: Manage local users, groups, and file system permissions. (Apply, Create)
3. CO3: Implement software installation, updates, and file system management. (Apply, Evaluate)
4. CO4: Configure Linux networking and secure services. (Apply, Create)
5. CO5: Perform system auditing, manage security policies, and ensure compliance. (Evaluate, Create)

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	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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**

Module Breakdown:

Module 1: Command Line and File Management (16 hours)

- **Topics:** Access the command line; Linux Basic and advanced commands; Recovery of the root user password; Managing files from the command line; Creating, Viewing, and Editing Text File.

Experiments:

Experiment 1.1. Navigate the Linux file system using command line.
Experiment 1.2. Execute basic Linux commands for file management.
Experiment 1.3. Recover the root user password.
Experiment 1.4. Create, view, and edit text files using nano and vim.
Experiment 1.5. Manage directories and file permissions.
Experiment 1.6. Use advanced commands like find, grep, and awk.
Experiment 1.7. Write and execute shell scripts.
Experiment 1.8. Document and present file management techniques.
Experiment 1.9. Implement a backup and restore procedure.
Experiment 1.10. Compare different text editors and their features.

Module 2: User Management and Process Monitoring (16 hours)

- **Topics:** Managing Local Linux Users and Groups; Using Virtualized Systems; Linux File System Permissions; Monitoring and Managing Linux Processes.

Experiments:

Experiment 2.1. Create and manage local users and groups.
Experiment 2.2. Configure file system permissions.
Experiment 2.3. Monitor processes using top, htop, and ps.
Experiment 2.4. Use virtualization tools like VirtualBox or VMware.
Experiment 2.5. Manage system resources and process priorities.
Experiment 2.6. Implement disk quotas for users.
Experiment 2.7. Use cron and at for scheduling tasks.
Experiment 2.8. Document and present user management best practices.
Experiment 2.9. Conduct a peer review of process management techniques.
Experiment 2.10. Compare virtualization tools and their features.

Module 3: Software and File System Management (16 hours)

- **Topics:** Archiving and Copying Files Between Systems; Installing and Updating Software Packages; Accessing Linux File Systems; Mounting and Unmounting File System.

Experiments:

Experiment 3.1. Archive files using tar and gzip.
Experiment 3.2. Copy files between systems using scp and rsync.
Experiment 3.3. Install and update software packages using apt and yum.
Experiment 3.4. Access and manage different file systems.
Experiment 3.5. Mount and unmount file systems.
Experiment 3.6. Use package managers for software maintenance.
Experiment 3.7. Configure software repositories.
Experiment 3.8. Document and present software management techniques.
Experiment 3.9. Implement a system update policy.
Experiment 3.10. Compare different file systems and their features.

Module 4: Networking and Security (16 hours)

- **Topics:** Linux Networking; Connecting Network Defined User and Groups; Analyzing and Storing Logs; Configuring and Securing OpenSSH Service; Using Regular Expressions with grep.

Experiments:

Experiment 4.1. Configure network interfaces and settings.
Experiment 4.2. Manage network users and groups.
Experiment 4.3. Analyze system logs using journalctl and logrotate.
Experiment 4.4. Secure SSH service with key-based authentication.
Experiment 4.5. Use grep with regular expressions for log analysis.
Experiment 4.6. Configure network services like DNS and DHCP.
Experiment 4.7. Implement firewall rules using iptables.
Experiment 4.8. Document and present network security best practices.
Experiment 4.9. Conduct a security audit of network configurations.
Experiment 4.10. Compare different logging and monitoring tools.

Module 5: Advanced Security and Storage Management (16 hours)

- **Topics:** Scheduling Future Linux Tasks; ACLs; SELinux Security; Adding Disks, Partitions, and File Systems to a Linux System; Managing Logical Volume Management (LVM) Storage.

Experiments:

Experiment 5.1. Schedule tasks using cron and at.
Experiment 5.2. Configure Access Control Lists (ACLs) for file permissions.
Experiment 5.3. Implement SELinux policies.
Experiment 5.4. Add and manage disks and partitions.
Experiment 5.5. Configure Logical Volume Management (LVM).
Experiment 5.6. Document and present advanced storage management techniques.
Experiment 5.7. Implement disk encryption.

Experiment 5.8. Conduct a peer review of SELinux configurations.
Experiment 5.9. Compare different storage management tools.
Experiment 5.10. Implement a secure backup and restore policy.

Module 6: Network Storage and Service Management (16 hours)

- **Topics:** Network Storage with NFS and SMB; Boot Process; Managing different services using systemctl; Planning and Configuring Security Updates.

Experiments:

Experiment 6.1. Configure NFS and SMB for network storage.
Experiment 6.2. Manage the Linux boot process.
Experiment 6.3. Use systemctl to manage services.
Experiment 6.4. Plan and implement security updates.
Experiment 6.5. Document and present network storage solutions.
Experiment 6.6. Configure network file sharing.
Experiment 6.7. Conduct a boot process analysis.
Experiment 6.8. Implement service monitoring and restart policies.
Experiment 6.9. Compare different network storage protocols.
Experiment 6.10. Implement a security update schedule.

Module 7: System Auditing and Security (16 hours)

- **Topics:** Basics of System Auditing; Security guidelines during installation; Configuring firewalld; Compliance Policy and Vulnerability Scanning.

Experiments:

Experiment 7.1. Perform a system audit using auditd.
Experiment 7.2. Implement security guidelines during installation.
Experiment 7.3. Configure firewalld for firewall management.
Experiment 7.4. Conduct vulnerability scanning using tools like Nessus.
Experiment 7.5. Document and present compliance policies.
Experiment 7.6. Implement a system hardening procedure.
Experiment 7.7. Conduct a security audit.
Experiment 7.8. Implement best practices for system security.
Experiment 7.9. Compare different auditing and scanning tools.
Experiment 7.10. Present findings from a security audit.

Textbooks and References:

1. Soyinka Wale, "Linux Administration: A Beginner's Guide," McGraw-Hill HED, Sixth Edition

2. Lucas Barnes, "Linux Server Management: A Deep Dive into Configuration and Optimization," Kindle Edition

Offensive Security (112 hrs)

Course Objective:

Subject Code	Subject Name	T-P-Pj (Credit)	Pre-Requisite
CUCS1102	Offensive Security	4 (2-2-0)	Nil

1. Equip students with the skills and knowledge necessary to conduct comprehensive penetration tests on networks, applications, and systems using industry-standard tools and methodologies.
2. Ensure students understand the legal and ethical implications of offensive security practices and can apply ethical hacking principles to identify and mitigate vulnerabilities.
3. Provide students with advanced knowledge of exploitation techniques, including the ability to discover, exploit, and document vulnerabilities

Course Outcomes (COs):

1. CO1: Explain the legal and ethical aspects of red teaming
2. CO2: Use network scanning and enumeration tools to gather information about network assets and services.
3. CO3: Interpret and analyze vulnerability assessment and penetration test results.
4. CO4: Evaluate the performance and security implications of different web protocols, frameworks, and server software, discerning best practices for optimizing web application development and deployment.
5. CO5: Develop comprehensive documentation and presentations for red team activities.

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	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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 1: 8 hours theory & 8 hours labs
Read teaming fundamentals

- Understanding Red Teaming
- Legal and Ethical Aspects
- Reconnaissance and Information Gathering
- Threat Modelling and Attack Vector

Module 2: 8 hours theory & 8 hours labs
Network Vulnerability Assessment & Penetration Testing:

- Network Scanning
- Enumeration
- Network Vulnerability Assessment Process & Phases
- Network Penetration Testing Process & Phases
- Advanced Network Vulnerability Assessment and Penetration Testing Techniques
- Tools and Frameworks for Network Vulnerability Assessment and Penetration Testing

Module 3: 8 hours theory & 8 hours labs
System Vulnerability Assessment & Penetration Testing

- System VAPT Tools Overview
- System Vulnerability Assessment Process & Phases
- System Penetration Testing Process & Phases
- Advanced System Vulnerability Assessment and Penetration Testing Techniques
- Case Studies and Practical Applications

Module 4: 8 hours theory & 8 hours labs
Post-Exploitation and Advanced Techniques

- Post-Exploitation Fundamentals
- Persistence, Evasion, and Clearing Tracks
- Data Exfiltration and Exfiltration Techniques
- Clearing Tracks After Exploitation
- MITRE Framework Integration
- Advanced Post-Exploitation Techniques

Module 5: 8 hours theory & 8 hours labs
Web Fundamentals

- How the Web Works
- Web Architecture & Web Protocols
- Web Requests, Client & Server Model

- Web Request Lifecycle
- Web App Coding and Frameworks - HTML, CSS, JS, PHP
- Web App Layers & SDLC Basics

Module 6: 8 hours theory & 8 hours labs
Web Application Vulnerability Assessment & Penetration Testing

- Basics of Web Vulnerability Assessment
- Introduction to Web Penetration Testing
- Web Application Vulnerabilities & Exploitation
- Web Application VAPT Tools & OWASP
- Advanced Web Application Vulnerabilities

Module 7: 8 hours theory & 8 hours labs
Communication Track - Red Teaming and Penetration Testing Reports and Documents

Subject Code	Subject Name	T-P-Pj (Credit)	Pre-Requisite
CUCS1103	Defensive Security	4 (2-2-0)	Nil

Training

- Effective Communication for VAPT and Red Teaming Reports
- Crafting Proposals for Red Teaming and VAPT Engagements
- Effective Presentation Skills for Red Teaming and VAPT Findings
- Writing Comprehensive Red Teaming and VAPT Reports

Text Books:

1."Red Team: How to Succeed By Thinking Like the Enemy" by Micah Zenko
2."The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy" by Patrick Engebretson

Defensive Security (112)

Course Objective:

1. Understanding of the foundational principles of defensive security, including risk management and threat modeling.
2. Design and deploy effective defensive security strategies, such as intrusion detection systems (IDS), firewalls, antivirus solutions, and encryption techniques.

3. Effectively respond to security incidents, including the identification, containment, eradication, and recovery from cyber attacks

Course Outcomes (COs):

1. CO1: Explain the role of operating systems in cybersecurity and the importance of secure file transfer protocols CO2: Use network scanning and enumeration tools to gather information about network assets and services.
2. CO2: Configure and manage IP addressing, DHCP, DNS, and routing protocols
3. CO3: Analyzing network traffic and identify anomalies using packet sniffers and network scanners
4. CO4: Assess the effectiveness of different defense tactics against APTs
5. CO5: Develop comprehensive compliance reports and automated reporting strategies

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	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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 1: 8 hours theory & 8 hours labs

Operating Systems from Security Perspective

- Operating System Fundamentals
- Windows Fundamentals
- Linux Fundamentals
- Secure File Transfer
- SSH Configuration Security
- Linux and Windows Hardening.

Experiment 1.1. Exploring File System Permissions in Linux – Experiment with file permissions using chmod to secure files.

Experiment 1.2. Configuring Secure SSH on a Linux Server – Set up and harden SSH access with public key authentication.

Experiment 1.3. Hardening a Windows Server – Implement security policies, disable unnecessary services, and enforce password complexity.

Experiment 1.4. Using Secure File Transfer (SFTP) – Configure and test SFTP for secure file transmission over a network.

Experiment 1.5. Logging and Auditing in Linux – Set up and analyze system logs using tools like syslog or journalctl.

Experiment 1.6. Windows Security Policies – Configure and manage local security policies on Windows for system hardening.

Experiment 1.7. Enforcing User Privileges in Linux – Implement sudo privileges to restrict root access.

Experiment 1.8. Security Patches and Updates – Test OS patching mechanisms and automate security updates.

Experiment 1.9. Secure Boot Configuration in Windows – Configure and test secure boot on a Windows OS.

Experiment 1.10. User and Group Management in Linux and Windows – Create users/groups and apply different security policies for each.

Module 2: 8 hours theory & 8 hours labs

Networking from Security Perspective

- Introduction to Networks
- Understanding Network Components
- Network Configuration, Administration, and Management
- Network Troubleshooting
- Network Architecture
- Securing Network Devices

Experiment 2.1. Setting Up a Secure Network Environment – Build a small network with secure protocols and firewall rules.

Experiment 2.2. Analyzing TCP/IP Traffic Using Wireshark – Capture and analyze basic network traffic.

Experiment 2.3. Configuring a Virtual Private Network (VPN) – Set up a VPN for secure remote access.

Experiment 2.4. Network Hardening Techniques – Disable unused network services and implement security on routers and switches.

Experiment 2.5. Network Troubleshooting Using Ping and Traceroute – Diagnose network issues using tools like ping, traceroute, and netstat.

Experiment 2.6. Securing Wireless Networks (WPA3) – Configure a wireless network with the latest security standards.

Experiment 2.7. Firewall Setup and Configuration – Install and configure a firewall for network protection.

Experiment 2.8. Monitoring Network Traffic – Use tools like Nagios or Zabbix to monitor network performance and security.

Experiment 2.9. Securing Routers and Switches – Implement access control lists (ACLs) on routers to block unauthorized traffic.

Experiment 2.10. Simulating a Network Attack – Perform basic penetration tests to expose vulnerabilities in a network.

Module 3: 8 hours theory & 8 hours labs

Blue Team Fundamentals

- Introduction to Blue Teaming
- Components of Blue Teaming
- Tools for Blue Teaming
- Defense Frameworks and Methodologies

Experiment 3.1. Setting Up Intrusion Detection Systems (IDS) – Install and configure tools like Snort for network defense.

Experiment 3.2. Incident Response Simulation – Simulate a security breach and carry out incident response steps.

Experiment 3.3. Implementing Honeypots – Deploy a honeypot to trap and monitor attackers in a controlled environment.

Experiment 3.4. Monitoring with Security Information and Event Management (SIEM) – Set up and analyze logs in a SIEM tool like Splunk.

Experiment 3.5. Analyzing Network Logs for Anomalies – Use log analysis tools to detect unusual activity.

Experiment 3.6. Using Firewalls and IDS for Blue Teaming – Implement firewall and IDS rules to protect systems from attacks.

Experiment 3.7. Cyber Defense Methodology – Simulate defense mechanisms using blue team tools against red team attacks.

Experiment 3.8. Patch Management and Vulnerability Scanning – Use tools like OpenVAS to scan and manage system vulnerabilities.

Experiment 3.9. Network Segmentation for Security – Simulate network segmentation to isolate critical systems.

Experiment 3.10. Conducting Forensic Analysis Post-Breach – Analyze and trace the source of a security breach in a lab setting.

Module 4: 8 hours theory & 8 hours labs

Network Security

- Introduction to Network Security
- Network Security Tools Overview
- Installation and Configuration of Network Security Tools
- Network Security Lab Exercises
- Defense Tactics and Techniques
- Advanced Network Security Topics

Experiment 4.1. Configuring Intrusion Prevention Systems (IPS) – Install and configure IPS to block malicious traffic.

Experiment 4.2. Network Firewall Configuration – Set up a firewall and configure rules for different network scenarios.

Experiment 4.3. Penetration Testing on Network Security Tools – Conduct penetration tests on the tools implemented to detect and analyze vulnerabilities.

Experiment 4.4. DoS Attack Simulation – Simulate a Denial of Service (DoS) attack and implement countermeasures.

Experiment 4.5. Installing and Configuring VPNs – Set up and configure VPNs for secure remote communication.

Experiment 4.6. Using Wireshark for Packet Analysis – Analyze network traffic and detect anomalies.

Experiment 4.7. Intrusion Detection with Snort – Configure Snort to detect and alert on malicious traffic.

Experiment 4.8. Setting Up Security on Network Devices – Secure routers and switches by disabling unused ports and configuring ACLs.

Experiment 4.9. Implementing and Analyzing Firewall Logs – Capture firewall logs and analyze for unusual traffic.

Experiment 4.10. Network Traffic Encryption with TLS/SSL – Configure and test encrypted traffic over the network.

Module 5: 8 hours theory & 8 hours labs
Network Analysis & Deep Packet Inspection

- Introduction to Network Analysis
- Introduction to Wireshark
- Understanding Packet Types and Basics
- Structure of Different Kinds of Packets
- Manual Packet Inspection Basics
- Advanced Network Analysis Techniques

Experiment 5.1. Packet Capture with Wireshark – Capture live network traffic and analyze different packet types.

Experiment 5.2. Examining HTTP and HTTPS Traffic – Analyze the differences in encrypted and unencrypted web traffic.

Experiment 5.3. Inspecting DNS Packets – Capture and analyze DNS traffic to understand how queries and responses work.

Experiment 5.4. TCP/IP Flow Analysis – Analyze packet flows for TCP handshake, data transmission, and termination.

Experiment 5.5. Analyzing Malicious Packets – Capture and analyze packets containing potential malware or exploits.

Experiment 5.6. Network Traffic Filtering with Wireshark – Set up filters to isolate specific types of traffic (e.g., ICMP, ARP).

Experiment 5.7. Deep Packet Inspection (DPI) Setup – Implement DPI tools to inspect packet contents for malicious activity.

Experiment 5.8. Understanding ARP Spoofing – Simulate an ARP spoofing attack and analyze traffic with Wireshark.

Experiment 5.9. Packet Fragmentation and Reassembly – Analyze fragmented packets and their reassembly in a network.

Experiment 5.10. Examining IPv6 Traffic – Capture and analyze IPv6 packets and compare them with IPv4 traffic.

Module 6: 8 hours theory & 8 hours labs

End Point & Advanced Persistent Threat

- Introduction to Endpoint Protection
- Advanced Persistent Threats (APT)
- Endpoint Protection Tools
- Installation and Configuration of Endpoint Protection Tools
- Managing Endpoint Protection Solutions
- Defense Tactics Against APTs

Experiment 6.1. Endpoint Protection Tool Installation – Install endpoint protection tools like Symantec or Bitdefender and test them.

Experiment 6.2. Configuring Endpoint Protection Policies – Set up and configure security policies on endpoint devices.

Experiment 6.3. Analyzing APT Behavior – Simulate an APT attack and monitor endpoint defense mechanisms.

Experiment 6.4. Implementing Patch Management on Endpoints – Automate patch deployment across endpoints and monitor for vulnerabilities.

Experiment 6.5. Simulating Malware Attacks on Endpoints – Simulate malware attacks and evaluate the endpoint protection tool's response.

Experiment 6.6. Endpoint Logging and Monitoring – Analyze endpoint logs to detect unusual or malicious activity.

Experiment 6.7. Installing Anti-APT Solutions – Implement specialized anti-APT solutions and simulate APT behavior.

Experiment 6.8. Encryption on Endpoint Devices – Configure encryption on endpoints for secure data storage.

Experiment 6.9. Mobile Device Endpoint Protection – Test mobile device management (MDM) for endpoint protection.

Experiment 6.10. Remote Endpoint Monitoring – Set up remote monitoring for endpoint devices and detect potential threats.

Module 7: 8 hours theory & 8 hours labs
Compliance, Technical Controls, Tools & Data Protection

- Introduction to Compliance Frameworks
- Different Compliance Standards
- Data Protection Tools
- Compliance Tools
- Technical Controls for Compliance
- Compliance Reporting Requirements

Experiment 7.1. Implementing GDPR Compliance – Simulate data protection measures to comply with GDPR regulations.
Experiment 7.2. Automating Compliance Reporting – Configure tools like Splunk to automate compliance reporting for audits.
Experiment 7.3. Implementing Data Loss Prevention (DLP) Tools – Set up and configure DLP tools to monitor and block unauthorized data access.
Experiment 7.4. Encryption Techniques for Data Protection – Use encryption tools like OpenSSL to secure sensitive data.
Experiment 7.5. Auditing Security Configurations for Compliance – Run security audits to check compliance with standards like ISO 27001.
Experiment 7.6. Managing Compliance Frameworks – Implement and simulate a compliance management system (e.g., HIPAA, PCI-DSS).

Subject Code	Subject Name	T-P-Pj(Credit)	Prerequisite

Experiment 7.7. Monitoring Data Access Logs – Set up and analyze data access logs to detect compliance violations.
Experiment 7.8. Using Compliance Tools for Vulnerability Management – Use tools like Nessus to scan systems for compliance gaps.
Experiment 7.9. Configuring Role-Based Access Control (RBAC) – Simulate the implementation of RBAC to meet compliance standards.
Experiment 7.10. Compliance Dashboard Setup – Set up a compliance monitoring dashboard to track the status of various controls and frameworks.

Text Book:

1."Operating System Concepts" by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne

2."Blue Team Handbook: Incident Response Edition: A condensed field guide for the Cyber Security Incident Responder" by Don Murdoch GSE

3."Defensive Security Handbook: Best Practices for Securing Infrastructure" by Lee Brotherston and Amanda Berlin

Security Analytics (112)

CUCS1104	Security Analytics	4(2-2-0)	NIL

Course Objective:

1. To equip students with the foundational knowledge and skills to understand and utilize various security analytics tools and techniques for monitoring and securing information systems.
2. To analyze and interpret security data, identify patterns of malicious activity, and assess vulnerabilities within an organization's network and systems.
3. To prepare students to develop and implement effective security strategies and response plans based on data-driven insights

Course Outcomes (COs):

1. CO1: Describe the purpose and importance of a Security Operations Center (SOC)
2. CO2: Utilize SIEM tools to collect, analyze, and correlate security data from various sources to identify potential threats.
3. CO3: Investigate and analyze security incidents using forensic tools and techniques to determine the cause and impact.
4. CO4: Assess the quality and relevance of threat intelligence data and its effectiveness in enhancing security measures.
5. CO5: Develop comprehensive threat hunting strategies, including the formulation of hunting hypotheses, selection of tools, and implementation of hunting campaigns.

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	-	-	-	1	2	2	2	3	2	3
CO2	3	3	3	3	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	3	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 1: 8 hours theory & 8 hours labs
Fundamentals of Purple Teaming & Tools

- Introduction to Purple Teaming
- Key Terminologies
- Introduction to SOC (Security Operations Center)
- Roles in a SOC
- Components to Build a SOC
- Purple Team Tools Overview

Module 2: 8 hours theory & 8 hours labs
SIEM (Security Information and Event Management)

- Introduction to SIEM
- SIEM Setup and Configuration
- Use Cases
- Custom Rules

Module 3: 8 hours theory & 8 hours labs
Incident Response

- SOAR (Security Orchestration, Automation, and Response)
- Incident Response Process
- Incident Response Tools
- Simulated Attack Scenarios

Module 4: 8 hours theory & 8 hours labs
Threat Intelligence

- Threat Intelligence Gathering & Sharing
- Mitigation Strategies
- Process & Phase-wise Approach
- Tools for Threat Intelligence

Module 5: 8 hours theory & 8 hours labs
Threat Hunting

- Adversary Tactics, Techniques, and Procedures (TTPs)
- Hunting for Indicators of Compromise (IoCs)
- Threat Hunting Methodologies

Module 6: 8 hours theory & 8 hours labs
Vulnerability Management

- Vulnerability Management Features and Process
- Tools for Vulnerability Management
- Asset Discovery and Inventory
- Vulnerability Scanners
- Patch Management

Module 7: 8 hours theory & 8 hours labs

Digital Forensics

- Forensic Methods & Investigations
- Tools for Digital Forensics
- Evidence Collection
- Hashing
- Linux & Windows Forensics
- Evidence Searching

Text Book:

1."Security Operations Center: Building, Operating, and Maintaining Your SOC" by Joseph Muniz, Gary McIntyre, and Nadhem AlFardan

2."Applied SIEM: Threat Hunting and Detection with Graylog" by Anthony Critelli

3."Security Information and Event Management (SIEM) Implementation" by David Miller, Shon Harris, and Allen Harper