Department of CSE

Details of Lesson Plan

S.No.	Particulars	Details
1.	Course Name	OOAD
2.	Course Code	COM-702
3.	Academic Year	2024-2025
4.	Semester	7th
5.	Number of Lesson plans	33
6.	Faculty Assigned	Azra Ashraf

Azra

Faculty Signature

Lesson Plan No. 1	Course Name: OOAD Topic: Introduction to OOAD	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. articulate the concept of object oriented programming and analysis b. Reiterate how these principles contribute to the benefits of OOAD, such as enhanced code reusability and improved maintainability.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & Talk
Teaching Development	1. **Introduction** (5 minutes) - Introduce the primary benefits of using OOAD in software development compared to procedural programming approaches? 2. **Development** (30 minutes) a. Introduction to Object oriented programming - Introduce the concept of four main principles of object-oriented programming: encapsulation, inheritance, polymorphism, and abstraction. - Discuss the significance of OOAD in modern software development. - Give an example of an online shopping system that needs to manage products, customers, and orders. Implementing OOAD in this context offers several advantages - E-Commerce Platforms (Amazon, eBay) - Online banking platforms like Chase, HSBC - Electronic Health Records (EHR) systems - Video games like World of Warcraft, Fortnite b. Challenges in OOAD - Complexity in Design - Overhead of Abstraction - Inheritance and Its Misuse - Performance Concerns 3. Exercise (5 minutes) – - To practice defining basic classes and their attributes and methods.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading

	- Researchgate Paper on object oriented analysis https://www.researchgate.net/publication/262350168_Object-oriented_analysis_and_design_OOAD 3. Homework -Choose a real-world system (e.g., an online store, a school management system, a social media platform). Describe how OOAD concepts can be applied to model this system. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Quiz on OOAD Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	Course Name: OOAD Topic: Benefits of using object-oriented analysis and design	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. By the end of the lesson, students will understand the primary benefits of OOAD and how these benefits impact software development practices and outcomes.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & Talk
Teaching Development	1. **Introduction** (5 minutes) - To explore the key benefits of OOAD and understand its impact on software development. - What challenges have we faced in software development with traditional approaches? 2. **Development** (30 minutes) - Discuss how OOAD promotes code reusability through inheritance and encapsulation. - Show how reusable components can be created in a library or framework. - Explain how OOAD allows easy modifications and extensions, using examples like adding new features to an e-commerce system. - Present case studies where OOAD has significantly benefited the development process. - Examples: E-commerce platforms, online banking systems, and large-scale software projects. - a. Advantages of OOAD - Enhanced Code Reusability - Improved Maintainability - Better Scalability -Increased Flexibility - Enhanced Understandability - Give examples to illustrate the advantages from a user-perspective. 3. Exercise (5 minutes) – - Provide a simple code example that demonstrates reusable classes and objects. Have students discuss how this improves efficiency.

	Use Nearpod to collect responses and discuss the answers.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - Researchgate Paper on object oriented analysis https://www.researchgate.net/publication/262350168_Object-oriented_analysis_and_design_OOAD 3. Homework -Compare maintaining a procedural codebase vs. an OOAD-based system. Use a case study to highlight differences. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Facilitate a discussion on how OOAD benefits were observed in these cases. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	Course Name: OOAD Topic:Introduction to the Unified Modeling Language (UML)	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. By the end of this lesson, students will have an understanding of UML, including its purpose, key diagram types, and how to use them for modeling software systems.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & Talk
Teaching Development	1. **Introduction** (5 minutes) - To introduce UML and its importance in modeling software systems. - Provide a brief history of UML and its purpose in software engineering. **Development** (30 minutes) a. **Use Case Diagram**(Shows interactions between users (actors) and the system.) - **Class Diagram(** Describes the static structure of the system, including classes, attributes, methods, and relationships. Show how reusable components can be created in a library or framework). -**Sequence Diagram(** Shows how objects interact in a particular sequence of events). -**Activity Diagram(** Represents workflows or processes, showing the sequence of activities). -**State Diagram(**Shows the states of an object and the transitions between those states). b. Advantages ofUML Diagrams - Improved Communication - Enhanced Understanding of the System - Facilitates System Design -Aids in Requirement Analysis - Supports Documentation c. Exercise (5 minutes) – -Discuss real-world scenarios where UML diagrams might be useful in planning and designing software projects. Use Nearpod to collect responses and discuss the answers.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading

	- Researchgate Paper on uses and applications of UML Diagrams https://www.researchgate.net/publication/349973644_UML_Diagrams_in_Software_Engineering_Research_A_Systematic_Literature_Review - 3. Homework -Ask students to select a simple system or process (e.g., a library system or a ticket booking system) and create the following UML diagrams: ● Use Case Diagram ● Class Diagram ● Activity Diagram . Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Open the floor for questions and clarifications about UML and its application. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	Course Name: OOAD Topic:Software development life cycle (SDLC) and OOAD.	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. By the end of this lesson, students will understand the Software Development Life Cycle (SDLC) and how Object-Oriented Analysis and Design (OOAD) fits into the SDLC.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & Talk
Teaching Development	1. **Introduction** (5 minutes) - To introduce the SDLC phases and discuss how OOAD integrates into these phases. **Development** (30 minutes) -Discuss the significance of having a structured development process and how OOAD can enhance various phases of the SDLC. - Overview of SDLC Phases - Object-Oriented Analysis and Design (OOAD) and its role in software development. - Integrating OOAD into SDLC b. Exercise (5 minutes) – -Create a detailed plan for a hypothetical software project, including the application of SDLC phases and OOAD techniques. Use Nearpod to collect responses and discuss the answers.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - Researchgate Paper on uses and applications of UML Diagrams https://www.researchgate.net/publication/349973644_UML_Diagrams_in_Software_Engineering_Research_A_Systematic_Literature_Review - 3. Homework Review a case study where OOAD was used effectively within an SDLC framework. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Open the floor for questions . 2. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5	Course Name: OOAD Topic:Software development life cycle (SDLC)	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. By the end of this lesson, students will understand the Software Development Life Cycle (SDLC) and how Object-Oriented Analysis and Design (OOAD) fits into the SDLC.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & Talk
Teaching Development	1. **Introduction** (5 minutes) - To introduce the SDLC phases and discuss how OOAD integrates into these phases. **Development** (30 minutes) -Discuss the significance of having a structured development process and how OOAD can enhance various phases of the SDLC. - Overview of SDLC Phases - Object-Oriented Analysis and Design (OOAD) and its role in software development. - Integrating OOAD into SDLC b. Exercise (5 minutes) – -Create a detailed plan for a hypothetical software project, including the application of SDLC phases and OOAD techniques. Use Nearpod to collect responses and discuss the answers.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - Researchgate Paper on uses and applications of UML Diagrams https://www.researchgate.net/publication/349973644_UML_Diagrams_in_Software_Engineering_Research_A_Systematic_Literature_Review - 3. Homework Review a case study where OOAD was used effectively within an SDLC framework. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Open the floor for questions . 2. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 6	Course Name: OOAD Topic: Requirement gathering and analysis	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the process of gathering and analyzing requirements for a software project b. Learn techniques for eliciting requirements, documenting them, and ensuring they are complete, clear, and testable..
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Introduce the topic and its importance. - Definition of requirements gathering and analysis. - Importance of accurate requirements for project success. 2. **Development** (30 minutes) a. Introduction to Functional requirements and Non-functional requirements - Understand various techniques for gathering requirements - Interviews - Surveys and Questionnaires - Observation - Document Analysis - Workshops b. Identify and engage stakeholders in the requirements gathering process. c. Exercise (5 minutes) – - Role-play: Conduct a mock interview to gather requirements.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - https://www.geeksforgeeks.org/requirements-gathering-introduction-processes-benefits-and-tools/ - 3. Homework

	- Choose a real-world project and identify at least five stakeholders. Write a brief report on how you would gather and document requirements for this project. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 7	Course Name: OOAD Topic:Understanding and Documenting User Requirements	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Students will learn the importance of understanding user requirements, techniques for eliciting these requirements, and methods for documenting them effectively to ensure successful project outcomes.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Definition of user requirements. - The role of user requirements in the software development lifecycle. 2. **Development** (30 minutes) a. Understand why user requirements are critical. - Impact on project success and user satisfaction. - Common pitfalls of poor requirement gathering b. Hershey's ERP Implementation - The FBI Virtual Case File (VCF) System
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - https://www.geeksforgeeks.org/requirements-gathering-introduction-processes-benefits-and-tools/ - 3. Homework - Choose a real-world application (e.g., a mobile app or website) and document at least five user requirements. Write a brief report on how you would refine and validate these requirements. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 8	Course Name: OOAD Topic:Understanding and Documenting User Requirements	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Students will learn the importance of understanding user requirements, techniques for eliciting these requirements, and methods for documenting them effectively to ensure successful project outcomes.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Definition of user requirements. - The role of user requirements in the software development lifecycle. 2. **Development** (30 minutes) a. Understand why user requirements are critical. - Impact on project success and user satisfaction. - Common pitfalls of poor requirement gathering b. Hershey's ERP Implementation - The FBI Virtual Case File (VCF) System
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - https://www.geeksforgeeks.org/requirements-gathering-introduction-processes-benefits-and-tools/ - 3. Homework - Choose a real-world application (e.g., a mobile app or website) and document at least five user requirements. Write a brief report on how you would refine and validate these requirements. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 9	Course Name: OOAD Topic:Introduction to different behavioural diagrams	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Students will understand the role and purpose of behavioral diagrams in software design and learn how to model dynamic aspects of a system using UML.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Brief Recap of UML (Structural vs. Behavioral Diagrams) - Importance of Behavioral Diagrams in Capturing System Behavior 2. **Development** (30 minutes) - Overview of the Key Behavioral Diagrams in UML: **Use Case Diagram** - Definition**:** Shows the system's functionality from the user's perspective. - Elements: Actors, Use Cases, System Boundary, Relationships (association, include, extend) - Example: Use case diagram for an online shopping system **Sequence Diagram** - Definition: Focuses on object interactions over time. - Elements: Lifelines, Messages, Activation Bars, Actors, and Objects. - Example: Sequence diagram for an online banking system.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - https://www.tutorialspoint.com/object_oriented_analysis_design/pdf/ooad_uml_behavioural_diagrams.pdf - 3. Homework

	- Design a behavioral model for a real-world system (e.g., hospital management system, flight booking system). - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 10	Course Name: OOAD Topic:Introduction to Interaction diagrams	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. To introduce students to interaction diagrams in UML and their significance. b. To explain different types of interaction diagrams: Sequence and Communication diagrams.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Overview of UML Behavioral Diagrams (brief recap). - Diagrams that describe interactions between objects to achieve a particular goal. - Types of Interaction Diagrams in UML: - Sequence Diagram - Communication Diagram 2. **Development** (30 minutes) **Sequence Diagram** - Definition: Represents the time-ordered sequence of messages between objects. - Key Elements: Lifelines, Actors, Objects, Messages, Activation Bars, Synchronous and Asynchronous Messages. - Example: Sequence diagram for the "Login" process in a web application. - Diagram Walkthrough: ○ Describe the components and explain how to interpret them. ○ Emphasize the flow of control (top-down message flow). **Communication Diagram** - Definition: Focuses on the interaction between objects and their relationships. - Key Elements: Objects, Links (connections), Messages (numbered to show sequence).

	- Difference from Sequence Diagram: While sequence diagrams focus on time and order, communication diagrams emphasize the structure and flow of messages between objects. **Example**: Communication diagram for a "ticket booking" system. **Diagram Walkthrough**: - Explain the use of objects, links, and message numbering. - Discuss how it models the interaction architecture of the system. - Discuss how interaction diagrams help in documenting requirements, testing, and system maintenance.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.tutorialspoint.com/uml/uml_interaction_diagram.htm 3. Homework - Provide a case study (e.g., a banking system or e-commerce website). Students choose one scenario and develop both a sequence diagram and a communication diagram to model the interactions. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 12	Course Name: OOAD Topic:Introduction to Component diagram	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. To understand the purpose and role of component diagrams in UML. b. To introduce the key elements and notations of component diagrams. c. To explain how component diagrams help in visualizing the physical structure of software systems. d. To engage students in designing component diagrams for real-world applications.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - What is a Component Diagram? - Definition: A diagram that represents the physical architecture of a software system, focusing on components and their relationships. - Importance: Models the high-level structure of the system, focusing on its physical components and how they interact. 2. **Development** (30 minutes) - Key Elements of a Component Diagram (20 minutes) - **Components**: Logical units of the system (e.g., databases, UI components, external services). - **Interfaces**: The entry and exit points for components. - **Connectors**: Dependencies and relationships between components (assembly connectors and delegation connectors). - **Ports**: Interface connectors used to show how components interact externally. - **Example**: Create a simple component diagram for an e-commerce platform, showing components like User Interface, Payment Gateway, Order Management System, and Database. - Component Diagram Notation
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading

	https://www.geeksforgeeks.org/component-based-diagram/ 3. Homework - **Case Study**: Example of a large-scale software system (e.g., online banking) where component diagrams are crucial for modularity and system scalability. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 11	Course Name: OOAD Topic:Introduction to Class and object diagrams	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. To understand the purpose and significance of class and object diagrams in UML. b. To introduce the key elements and notations of class and object diagrams. c. To explain the difference between class diagrams and object diagrams. d. To guide students through the process of designing class and object diagrams.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - **Class Diagrams**: Static diagrams that describe the structure of a system by showing its classes, attributes, operations, and relationships. - **Object Diagrams**: Instances of class diagrams, representing the system's state at a particular moment. 2. **Development** (30 minutes) **-Class Diagram in UML** - Key Elements: - Classes: Represent entities with attributes and methods. -Attributes: Properties or data elements associated with a class. -Methods/Operations: Functions associated with a class. -Relationships: Associations, generalization, aggregation, composition, and dependency. **Class Notation**: - **Class Box**: Name of the class, attributes (private, public, protected), methods (operations). - **Relationships**: Lines, arrows, and their types. **Definition**: Shows a snapshot of the system at a specific point in time, with instantiated objects and their relationships. **Key Elements**: - Objects: Instances of classes with specific values for their attributes.

	- Links: Connections between objects that represent relationships. - **Object Notation**: - Use of object names and their attributes (with values), and links between objects. -
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.irjet.net/archives/V8/i8/IRJET-V8I8224.pdf 3. Homework - Students design a class diagram for a "Student Information System" with classes such as Student, Course, Faculty, and Department.. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 12	Course Name: OOAD Topic:Object-oriented principles: inheritance,polymorphism,encapsulation, and abstraction	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the four fundamental object-oriented principles: **Inheritance**, **Polymorphism**, **Encapsulation**, and **Abstraction**. b. Apply these principles in real-world programming scenarios.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) -Briefly explain how OOP models real-world entities using objects and classes. 2. **Development** (30 minutes) - Explain that **inheritance** allows one class (subclass) to inherit fields and methods from another class (superclass). - Superclass, subclass - Method overriding - Explain **polymorphism** as the ability of objects of different classes related by inheritance to respond to method calls of the same name, each one according to its own behavior. - Compile-time polymorphism - Runtime polymorphism - Explain encapsulation as the bundling of data (variables) and methods that operate on the data into a single unit, or class, with controlled access via access modifiers. - Access Modifiers - Explain **abstraction** as hiding implementation details while exposing only the essential features of an object.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.geeksforgeeks.org/understanding-encapsulation-inheritance-polymorphism-abstraction-in-oops/

	3. Homework - Recap how these principles enhance **modularity**, **maintainability**, and **reusability** in software development. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 13	Course Name: OOAD Topic:Design patterns and their applications	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of **design patterns** and their significance in software development. b. Identify different types of design patterns, including **creational**, **structural**, and **behavioral** patterns. c. Apply common design patterns, such as **Factory**, **Singleton**, **Observer**, and **Decorator**, in software solutions.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) **-Categories of Design Patterns**: - Creational Patterns: Handle object creation mechanisms. - Structural Patterns: Focus on how classes and objects are composed. - Behavioral Patterns: Concerned with communication between objects. . 2. **Development** (30 minutes) **Factory Pattern:** - Definition:A pattern that provides a way to create objects without specifying the exact class of the object that will be created. - **Singleton Pattern** - Definition: Ensures a class has only one instance and provides a global point of access to that instance. **Structural Patterns:** - **Decorator Pattern** - **Definition**: Allows behavior to be added to individual objects, either statically or dynamically, without affecting the behavior of other objects from the same class. **Adapter Pattern:** - **Definition**: Converts the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.(Bridging two incompatible interfaces)

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://stackify.com/introduction-to-design-patterns-in-software-development/ 3. Homework - Discuss how patterns help avoid redundant work, improve communication among developers, and enhance the scalability of software systems - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 13	Course Name: OOAD Topic:Design patterns and their applications	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of **design patterns** and their significance in software development. b. Identify different types of design patterns, including **creational**, **structural**, and **behavioral** patterns. c. Apply common design patterns, such as **Factory**, **Singleton**, **Observer**, and **Decorator**, in software solutions.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) **-Categories of Design Patterns**: - Creational Patterns: Handle object creation mechanisms. - Structural Patterns: Focus on how classes and objects are composed. - Behavioral Patterns: Concerned with communication between objects. . 2. **Development** (30 minutes) **Factory Pattern:** - Definition:A pattern that provides a way to create objects without specifying the exact class of the object that will be created. - **Singleton Pattern** - Definition: Ensures a class has only one instance and provides a global point of access to that instance. **Structural Patterns:** - **Decorator Pattern** - **Definition**: Allows behavior to be added to individual objects, either statically or dynamically, without affecting the behavior of other objects from the same class. **Adapter Pattern:** - **Definition**: Converts the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.(Bridging two incompatible interfaces)

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://stackify.com/introduction-to-design-patterns-in-software-development/ 3. Homework - Discuss how patterns help avoid redundant work, improve communication among developers, and enhance the scalability of software systems - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 15	Course Name: OOAD Topic:Component-based design and modularization.	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a) Understand the concepts of **component-based design** and **modularization** in software development. b) Identify the benefits of using components and modules in designing scalable and maintainable systems. c) Apply component-based design principles to create modular software applications.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) -Explain that component-based design involves building software systems from reusable components that encapsulate functionality and are independently deployable. **Characteristics of Components**: - Encapsulation of functionality. - Reusability across different systems. - Interoperability with other components. **Benefits of Component-Based Design**: - Improved maintainability and scalability. - Faster development due to reusability. - Easier testing and integration. **Real-world Examples**: Discuss examples like microservices architecture and the use of libraries (e.g., React components) in web development. 2. **Development** (30 minutes) - Introduction to Modularization - Cohesion: Modules should have a single, well-defined purpose. - Coupling: Minimize dependencies between modules. **Benefits of Modularization**:

	- Improved readability and understandability of code. - Enhanced collaboration among development teams. **Designing Components and Modules** **Identifying Components**: Explain how to identify potential components in a system based on functionality. Use an example of an e-commerce application to identify components like **Product**, **Shopping Cart**, **Payment**, etc. **Component Interfaces**: Discuss the importance of defining clear interfaces for components to allow interaction without tight coupling.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://stackify.com/introduction-to-design-patterns-in-software-development/ 3. Homework - Summarize the importance of component-based design and modularization in modern software development. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 14	Course Name: OOAD Topic:Component-based design and modularization.	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a) Understand the concepts of **component-based design** and **modularization** in software development. b) Identify the benefits of using components and modules in designing scalable and maintainable systems. c) Apply component-based design principles to create modular software applications.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) -Explain that component-based design involves building software systems from reusable components that encapsulate functionality and are independently deployable. **Characteristics of Components**: - Encapsulation of functionality. - Reusability across different systems. - Interoperability with other components. **Benefits of Component-Based Design**: - Improved maintainability and scalability. - Faster development due to reusability. - Easier testing and integration. **Real-world Examples**: Discuss examples like microservices architecture and the use of libraries (e.g., React components) in web development. 2. **Development** (30 minutes) - Introduction to Modularization - Cohesion: Modules should have a single, well-defined purpose. - Coupling: Minimize dependencies between modules. **Benefits of Modularization**:

	- Improved readability and understandability of code. - Enhanced collaboration among development teams. **Designing Components and Modules** **Identifying Components**: Explain how to identify potential components in a system based on functionality. Use an example of an e-commerce application to identify components like **Product**, **Shopping Cart**, **Payment**, etc. **Component Interfaces**: Discuss the importance of defining clear interfaces for components to allow interaction without tight coupling.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://stackify.com/introduction-to-design-patterns-in-software-development/ 3. Homework - Summarize the importance of component-based design and modularization in modern software development. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 16	Course Name: OOAD Topic:System Architecture and Design	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of system architecture in the context of object-oriented systems. b. Apply object-oriented principles (e.g., encapsulation, inheritance) to design system architecture. c. Identify and describe key components of object-oriented system architecture.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Discuss how architecture handles complexity, supports scalability, and ensures maintainability in OO systems. - Briefly recap the object-oriented principles—encapsulation, inheritance, polymorphism, and abstraction—and their relevance to system architecture. 2. **Development** (30 minutes) -Object-Oriented System Architecture Components - Layers in OO Architecture - Presentation Layer - Business Logic Layer - Data Access Layer - UML Diagrams for System Architecture: - Class Diagrams - Component Diagrams - Interaction Diagrams **Case Study**: Present a simple object-oriented case study, such as a banking system, and explain how these layers interact.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading 3. Homework - Design an object-oriented architecture for a given system (e.g., an online ticket booking system). Students must include class diagrams and explain how their design supports scalability and maintainability. - Spend 5 minutes to wrap up and consolidate the learnings

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 17	Course Name: OOAD Topic:Architectural Styles in Object-Oriented Systems	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of system architecture in the context of object-oriented systems. b. Apply object-oriented principles (e.g., encapsulation, inheritance) to design system architecture. c. Identify and describe key components of object-oriented system architecture.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Discuss how architecture handles complexity, supports scalability, and ensures maintainability in OO systems. - Briefly recap the object-oriented principles—encapsulation, inheritance, polymorphism, and abstraction—and their relevance to system architecture. 2. **Development** (30 minutes) -Layered Architecture - Client-Server and Distributed Architectures in OOAD - Design Principles in Object-Oriented Architecture -SOLID Principles -Scalability and Extensibility **Case Study Discussion**: Revisit the banking system example and discuss how SOLID principles can improve the architecture. -Application of Design Patterns in OOAD Architecture - Factory Pattern - Observer Pattern - Singleton Pattern
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading 3. Homework - Design an object-oriented architecture for a given system (e.g., an online ticket booking system). Students must include class diagrams and explain how their design supports scalability and maintainability. - Spend 5 minutes to wrap up and consolidate the learnings

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 18	Course Name: OOAD Topic:Architectural patterns in OOAD	Course No.: COM-702

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of system architecture in the context of object-oriented systems. b. Apply object-oriented principles (e.g., encapsulation, inheritance) to design system architecture. c. Identify and describe key components of object-oriented system architecture.
Teaching Aids (if any)	a. Powerpoint Presentation b. Chalk & talk
Teaching Development	1. **Introduction** (5 minutes) - Discuss how architecture handles complexity, supports scalability, and ensures maintainability in OO systems. - Briefly recap the object-oriented principles—encapsulation, inheritance, polymorphism, and abstraction—and their relevance to system architecture. 2. **Development** (30 minutes) -Layered Architecture - Client-Server and Distributed Architectures in OOAD - Design Principles in Object-Oriented Architecture -SOLID Principles -Scalability and Extensibility **Case Study Discussion**: Revisit the banking system example and discuss how SOLID principles can improve the architecture. -Application of Design Patterns in OOAD Architecture - Factory Pattern - Observer Pattern - Singleton Pattern
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading 3. Homework - Design an object-oriented architecture for a given system (e.g., an online ticket booking system). Students must include class diagrams and explain how their design supports scalability and maintainability. - Spend 5 minutes to wrap up and consolidate the learnings

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. 2. Open floor for questions. Spend 5 minutes to evaluate student assimilation of the lesson contents