**MODEL Question**

**UNIT-I: Classes and UML Basics**

**Remembering (Knowledge Level)**

1. Define UML and explain its significance in software design.
2. List the building blocks of UML.
3. What are the different phases of the Rational Unified Process (RUP)?

**Understanding (Comprehension Level)**

1. Explain the importance of the 4+1 architecture view in system modeling.
2. Describe the difference between Generalization and Inheritance in UML.
3. Differentiate between Link and Association concepts in UML.

**Applying (Application Level)**

1. Draw a basic UML class diagram for a Library Management System.
2. Identify objects and classes for an Online Banking System and represent them in UML.
3. Apply the concept of abstraction to model a Student Registration System.

**Analyzing (Analysis Level)**

1. Compare and contrast MDA and XMI in the context of OMG standards.
2. Analyze the necessity of modeling in object-oriented thinking.
3. Examine how UML Meta Model influences software design.

**Evaluating (Evaluation Level)**

1. Justify the need for UML diagrams in large-scale software development.
2. Critically evaluate the benefits of using RUP in software development life cycle.
3. Assess the role of abstraction in software modeling.

**Creating (Synthesis Level)**

1. Design a UML class diagram for an E-commerce application.
2. Develop a UML model representing a Hospital Management System.
3. Formulate a set of class relationships for an ATM system and justify your design choices.

**UNIT-II: State and Use Case Modeling**

**Remembering (Knowledge Level)**

1. Define events, states, and transitions in state modeling.
2. List the components of a state diagram.
3. What are the different relationships in a Use Case diagram?

**Understanding (Comprehension Level)**

1. Explain the need and purpose of state modeling in software development.
2. Differentiate between Include and Extend relationships in Use Case diagrams.
3. Describe the role of actors in a Use Case diagram.

**Applying (Application Level)**

1. Draw a state diagram for an Online Food Ordering System.
2. Construct a Use Case diagram for an Online Examination System.
3. Identify actors and use cases for a Flight Reservation System.

**Analyzing (Analysis Level)**

1. Compare and contrast the roles of primary and secondary actors in Use Case diagrams.
2. Analyze how advanced class concepts impact object-oriented system modeling.
3. Examine the advantages of using state diagrams in real-time system development.

**Evaluating (Evaluation Level)**

1. Evaluate the effectiveness of Use Case diagrams in software requirement analysis.
2. Justify the necessity of formal Use Case documentation in software projects.
3. Assess the impact of event-driven modeling on software development.

**Creating (Synthesis Level)**

1. Design a Use Case diagram for an Online Shopping Portal with detailed actor interactions.
2. Develop a state diagram for a Mobile Banking Application.
3. Formulate a strategy to model real-time traffic light control using state diagrams.

**UNIT-III: Activity and Sequence Diagrams**

**Remembering (Knowledge Level)**

1. Define an activity diagram and list its components.
2. What is the purpose of swimlanes in an activity diagram?
3. Explain the significance of sequence diagrams in software design.

**Understanding (Comprehension Level)**

1. Differentiate between decision and merge points in activity diagrams.
2. Explain the role of activation and focus of control in sequence diagrams.
3. Describe how sequence diagrams help in modeling system interactions.

**Applying (Application Level)**

1. Construct an activity diagram for an Online Course Registration System.
2. Draw a sequence diagram for an E-Ticket Booking System.
3. Model a sequence diagram for a Customer Complaint Management System.

**Analyzing (Analysis Level)**

1. Compare activity diagrams and sequence diagrams in terms of system modeling.
2. Examine how sequence diagrams help in software testing and debugging.
3. Analyze the significance of parallel execution in activity diagrams.

**Evaluating (Evaluation Level)**

1. Justify the use of swimlanes in an activity diagram for business process modeling.
2. Evaluate the effectiveness of sequence diagrams in modeling real-time systems.
3. Assess the impact of modeling interactions on software scalability.

**Creating (Synthesis Level)**

1. Design an activity diagram for an Online Banking Transaction System.
2. Develop a sequence diagram for a Smart Home Automation System.
3. Formulate a set of interaction rules for designing robust sequence diagrams.

**UNIT-IV: Collaboration, Timing, and Design Patterns**

**Remembering (Knowledge Level)**

1. Define collaboration diagrams and explain their components.
2. List different types of design patterns in the GoF classification.
3. What is the purpose of GRASP patterns?

**Understanding (Comprehension Level)**

1. Explain the difference between collaboration diagrams and sequence diagrams.
2. Describe the role of responsibility assignment patterns in object-oriented design.
3. Differentiate between creational, structural, and behavioral design patterns.

**Applying (Application Level)**

1. Construct a collaboration diagram for a Chat Application.
2. Apply the Singleton pattern in designing a Logger class.
3. Implement the Adapter pattern to bridge incompatible interfaces.

**Analyzing (Analysis Level)**

1. Compare the benefits of collaboration diagrams over sequence diagrams.
2. Examine how GRASP patterns influence software architecture.
3. Analyze the role of polymorphism in software design patterns.

**Evaluating (Evaluation Level)**

1. Justify the need for Gang of Four (GoF) design patterns in software development.
2. Evaluate the effectiveness of protected variations in handling system complexity.
3. Assess the impact of applying controller design patterns in enterprise applications.

**Creating (Synthesis Level)**

1. Design a communication diagram for a Hotel Booking System.
2. Develop a software architecture using GRASP principles for a Smart Parking System.
3. Formulate a design pattern strategy for an E-Learning Platform.

**UNIT-V: Component and Deployment Diagrams**

**Remembering (Knowledge Level)**

1. Define component diagrams and list their elements.
2. What are the key components of a deployment diagram?
3. Explain the concept of forward and reverse engineering in UML.

**Understanding (Comprehension Level)**

1. Describe the purpose of interfaces and ports in component diagrams.
2. Explain the difference between two-tier and three-tier architectures.
3. Discuss the importance of domain class models in system design.

**Applying (Application Level)**

1. Construct a component diagram for a Banking System.
2. Develop a deployment diagram for a Cloud-Based File Storage System.
3. Apply UML principles to model a Ticket Reservation System.

**Analyzing (Analysis Level)**

1. Compare forward engineering and reverse engineering approaches in UML.
2. Examine how component diagrams aid in software modularity.
3. Analyze the role of deployment diagrams in system deployment planning.

**Evaluating (Evaluation Level)**

1. Evaluate the impact of system architecture on software performance.
2. Justify the use of UML domain models in software development.
3. Assess the importance of development life cycle modeling in large-scale applications.

**Creating (Synthesis Level)**

1. Design a component diagram for an IoT-based Smart Home System.
2. Develop a deployment strategy for a Distributed Healthcare Information System.
3. Formulate a UML model that integrates component and deployment diagrams for an AI-based Chatbot.