OOSE Module – IA-2 QB with Answers

[4 marks]

1] Define an Actor and a Use Case in the context of the Use Case Diagram.

1. Actor (in Use Case Diagram)

An Actor is an external entity that interacts with the system. It may be a user, another system, a device, or any external component that communicates with the system.

1. **Actors initiate use cases and receive results**. They are the source of interactions and represent the role played by a real-world entity.
2. **Actors are shown using stick-figure notation** (as seen on page 2 image where "Actor" is drawn as a stick figure).
3. They are **not part of the system**, but they influence and use its functionality.

2. Use Case (in Use Case Diagram)

A Use Case represents a specific function or action the system performs for an actor. Examples include "Place Order," "Track Delivery," or "Browse Products."

1. Use cases describe **functional requirements**—what the system should do from the user's perspective.
2. They are **represented by ovals**, as shown on page 2 where the "Use Case" oval symbol is illustrated.
3. A use case models a **goal-oriented scenario**, showing the steps or behavior the system executes to deliver value to the actor.

2] Explain the primary purpose of a Sequence Diagram.

1. A sequence diagram shows how objects or people interact step by step. It mainly focuses on the order in which messages are passed.

2. It helps explain the exact flow of a process inside the system. This makes it easy to see what happens first and what happens next.

3. It shows timing clearly by placing events from top to bottom. This helps understand how the system behaves over time.

4. It helps developers know which object is responsible for which action. This improves understanding of system roles and communication.

5. It is useful during design because it explains a use case in detail. This makes planning and building the system much easier.

3] List the main elements involved in a State Transition Diagram.

1. **States**
   A state shows the condition or situation of an object at a specific time. It represents what the object is doing or waiting for during the process.
2. **Transitions**
   A transition is the movement from one state to another. It shows how the system changes when something important happens.
3. **Events**
   An event is something that triggers a transition. It can be a user action, system signal, or any activity that causes change.
4. **Initial State**
   This is the point where the system or object begins. It is usually shown as a filled black circle.
5. **Final State**
   This represents the end of the process for that object. It is shown as a circle with another filled circle inside it.
6. **Actions**
   Actions are activities performed during a transition or inside a state. They help show what work the system does while moving through states.

4] Differentiate between a Sequence Diagram and a Collaboration Diagram.

1. **Focus of the Diagram**
   A sequence diagram focuses mainly on the **time order** of messages. It shows what happens first, next, and last in a vertical timeline.
2. **View of Interaction**
   A collaboration diagram focuses on **how objects are connected** and communicate. It highlights relationships between objects more than time order.
3. **Layout Style**
   A sequence diagram arranges objects **horizontally** and messages **vertically**. This makes it easier to see timing clearly.
4. **Message Representation**
   A collaboration diagram uses **numbered messages** to show order. Sequence diagrams use the **top-to-bottom position** to show order naturally.
5. **Use in Design**
   Sequence diagrams are better when you want to understand **exact timing and flow**. Collaboration diagrams are better when you want to understand **structure and communication links**.

5] Name the three primary types of Interaction Diagrams discussed in the syllabus.

1. **Sequence Diagram**
   A sequence diagram shows how objects interact step-by-step over time. It focuses on the order of messages, making the flow very clear.
2. **Collaboration (Communication) Diagram**
   A collaboration diagram shows how objects are linked and communicate with each other. It focuses more on relationships and structure rather than time order.
3. **Interaction Overview Diagram**
   An interaction overview diagram gives a high-level view of interactions. It combines features of activity diagrams and sequence diagrams to show the overall control flow of multiple interactions.

These three diagrams together help understand timing, communication, and overall behavior of a system clearly.

6] Explain how an Activity Diagram is used to model the flow of control in a system.

1. **Shows the starting point of the process**
An activity diagram begins with an initial node. This tells us where the process starts in the system.
2. **Represents each step as an activity**
Every action or task is shown as a rounded box. This helps us see what the system does at each stage.
3. **Uses arrows to show movement between steps**
Arrows indicate how control moves from one activity to the next. This makes the flow easy to follow.
4. **Includes decision points for conditions**
Diamonds show places where the system must choose between different paths. This helps model real-life choices and branching.
5. **Shows parallel work using fork and join**
Some tasks can happen at the same time, and forks show this clearly. Joins bring the paths back together.
6. **Ends with a final node**
The diagram finishes with an end symbol. This shows where the process stops.

7] Define the term Develop Use-case Model.

1. Develop Use-case Model means creating a clear picture of how users interact with a system. It focuses on understanding what the system should do from the user's point of view.
2. It includes identifying all the actors who use the system. These may be people, devices, or external systems that interact with the software.
3. It also involves listing all the use cases or functions the system must provide. Each use case represents a specific task or goal the user wants to achieve.
4. The process helps define the system's functional requirements in a simple and visual way. This makes communication easy between developers, designers, and stakeholders.
5. Developing a use-case model ensures that the system design matches user needs. It guides later steps like designing, coding, and testing.

8] Describe the role of a guard condition in an Activity Diagram.

1. **A guard condition controls which path the process should follow.** It acts like a small rule written on an arrow to decide if that path is allowed.

2. **It is placed near a decision node to guide the flow.** This helps the system choose the correct direction based on the condition.

3. **A guard ensures that the next activity happens only if the condition is true.** If the condition is false, the system will follow another arrow with a different guard.

4. **It makes the diagram more accurate by showing real-life choices.** This helps represent situations where different outcomes depend on specific values or checks.

5. **Overall, guard conditions help model logical decisions clearly.** They prevent confusion and make the flow of control easy to understand for developers and users.

[5- marks]

1] Explain the process of Identifying Actors and Use Cases for a new software system

1. **Start by understanding who will use the system in real life.** Think about all groups of people or external systems that will interact with the software. These become your actors.

2. **Identify different roles, not individuals.** For example, "Customer," "Admin," and "Employee" are roles, even if many people play them.

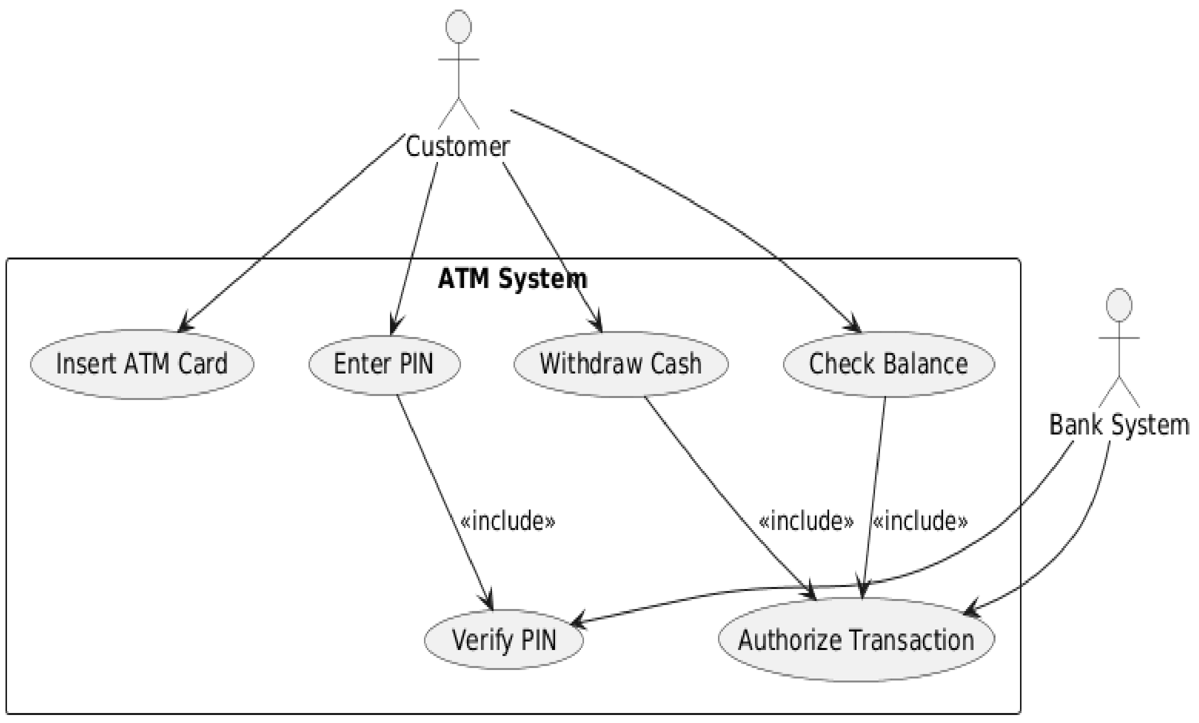
3. **Next, find out what each actor wants to achieve using the system.** These goals or tasks become your use cases.

4. **Talk to users, stakeholders, or clients to understand their needs.** This helps ensure that the system supports all important functions.

5. **List all actions the system must perform for each actor.** These actions form the main use cases like "Login," "Place Order," or "Generate Report."

6. **Finally, group and refine the use cases to remove duplicates.** This creates a clear and complete use-case model for the new software system.

2] Draw a simple Use Case Diagram for an ATM System, showing at least two actors and four use cases.



3] Discuss the different components of a detailed Description of Use Case Diagram (narrative documentation).

1 **Use Case Name**
This gives a clear title for the use case. It helps everyone understand what action or function the use case represents.

2 **Actor(s)**
These are the users or external systems that participate in the use case. They show who starts the process or receives the result.

3 **Brief Description**
This explains the main purpose of the use case in a short paragraph. It gives a simple overview of what the system will do.

4 **Preconditions**
These are conditions that must be true before the use case starts. They ensure the system is ready for the action.

5 **Postconditions**
These describe what must be true after the use case finishes. They help confirm that the process ended correctly.

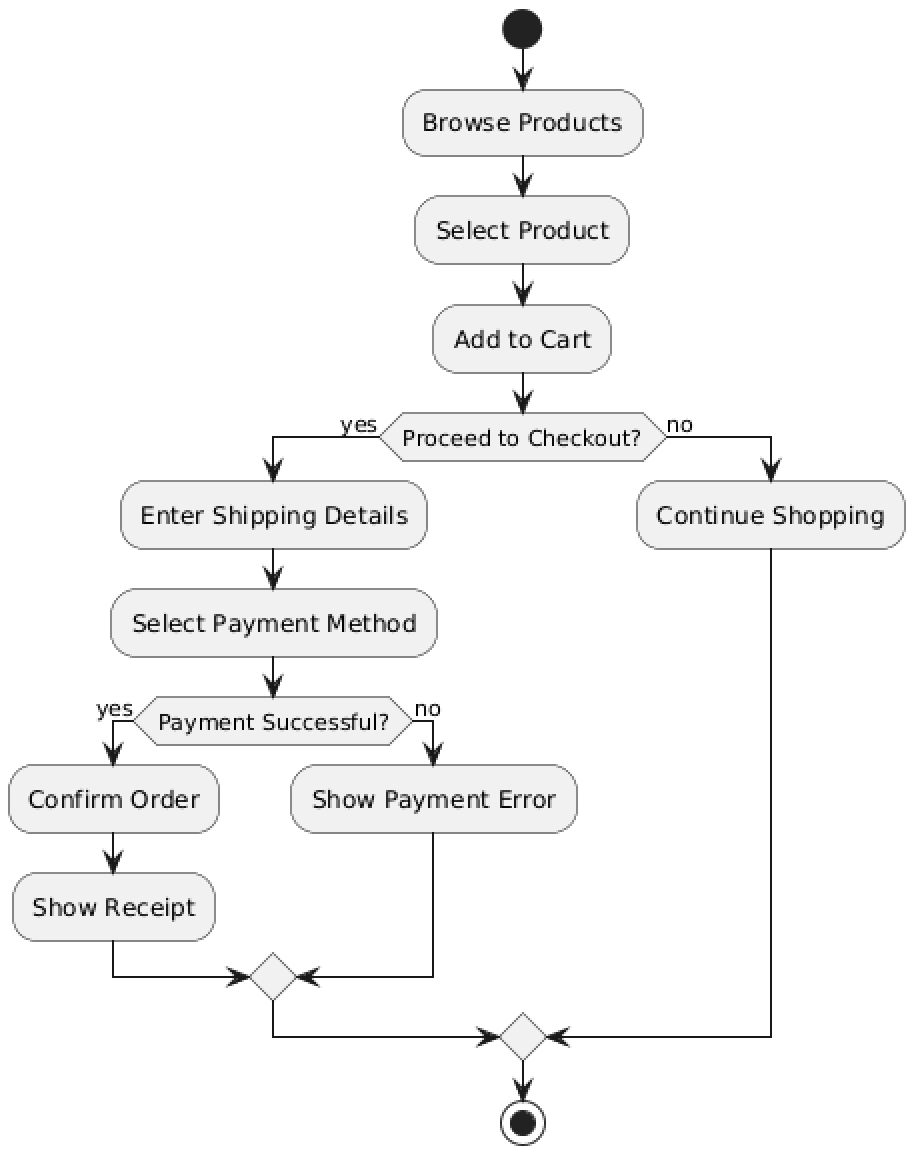
6 **Basic Flow (Main Success Scenario)**
This is the normal step-by-step sequence of events when everything works correctly. It explains how the actor and system interact successfully.

7 **Alternative Flows**
These describe different paths the use case may take when the user chooses another option. They show optional actions or different outcomes.

8 **Exception Flows**
These cover error situations like invalid input or system failure. They help the system handle unexpected problems safely.

9 **Triggers**
This states what event starts the use case. It could be a user action, system event, or time-based trigger.

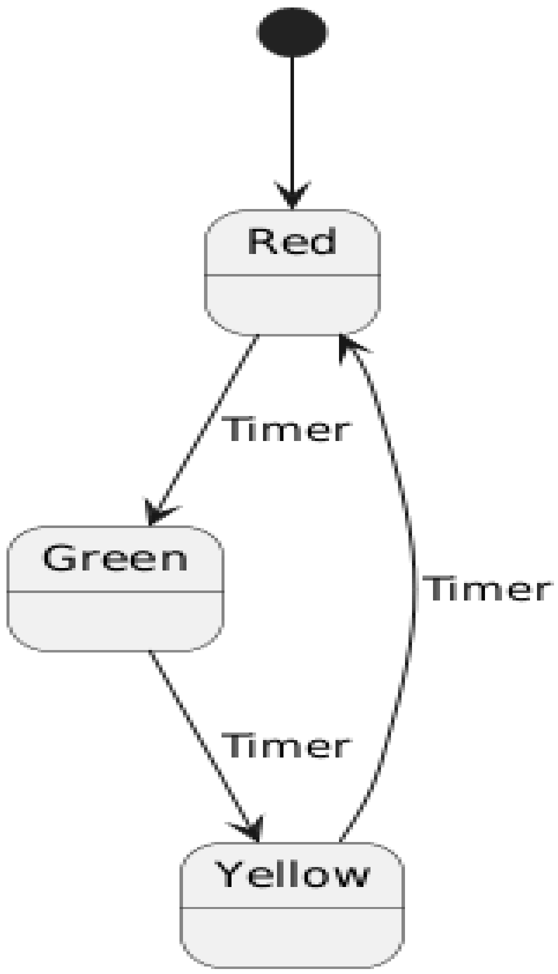
4] Draw a simple Activity Diagram to model the process of a customer making an Online Purchase.



5] Explain how the Sequence Diagram illustrates the time-ordered sequence of interactions between objects.

1. A sequence diagram shows interactions from **top to bottom**, which represents time moving forward. Anything drawn higher happens earlier, and anything lower happens later.
2. Each object has a **lifeline**, a vertical dotted line that shows how long the object exists during the interaction. This helps us see when the object starts acting and when it becomes idle.
3. **Messages are drawn as arrows** between lifelines. The position of each arrow on the diagram shows the exact order in which the messages happen.
4. When one message depends on another, the arrows show this clearly. This makes it easy to understand which actions must happen first.
5. Activation bars show when an object is **busy doing some work**. This helps explain how long an object spends processing a message.
6. Overall, the diagram gives a clear, step-by-step timeline of how objects communicate. It visually explains the complete flow of a use case over time.

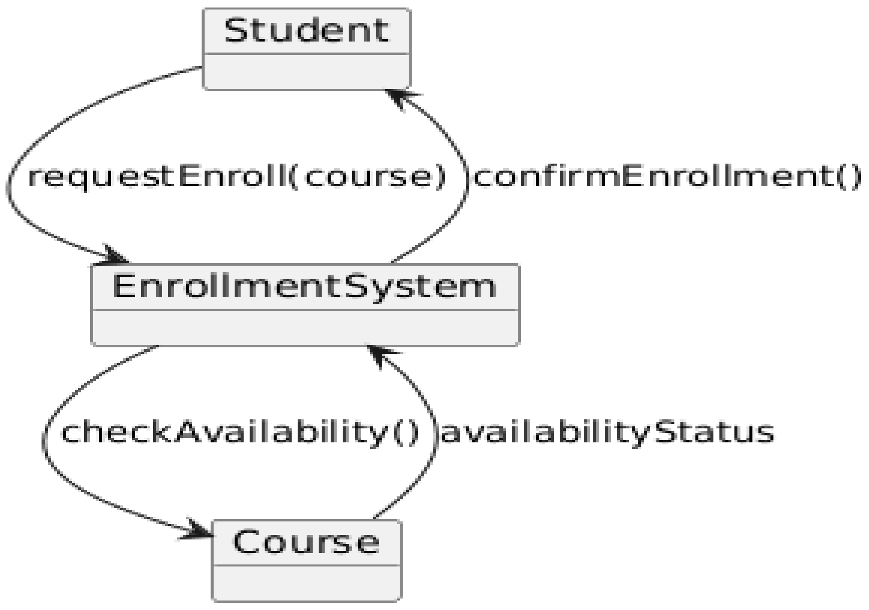
6] For a Traffic Light System, draw a simple State Transition Diagram showing the states (Red, Yellow, Green) and the transitions between them.



7] Differentiate between Behavioral Modeling and Structural Modeling in software engineering.

1. **Structural Modeling shows the static parts of a system.** It explains how classes, objects, and their relationships are arranged. These structures do not change during program execution.
2. **Behavioral Modeling shows how the system behaves over time.** It focuses on actions, events, and interactions that happen when the system runs. This helps understand the system's dynamic nature.
3. **Structural Modeling uses diagrams like Class Diagrams and Object Diagrams.** These diagrams describe the architecture and organization of the system.
4. **Behavioral Modeling uses diagrams like Activity, Sequence, and State Diagrams.** These show how objects communicate, how processes flow, and how states change.
5. **Structural Modeling answers "what the system contains."** Behavioral Modeling answers "how the system works."
6. Together, both models give a **complete understanding of the system**, covering design and behavior.

8] Draw a simplified Collaboration Diagram to show how a Student Object and Course Object interact to Enroll in a Course.



[10 marks]

1] Explain the different Interaction Diagrams (Sequence, Activity, State Transition) in detail. Analyze the specific type of information each diagram captures about system behavior.

Interaction diagrams help us understand how a system behaves when it is running. Each diagram focuses on a different view of the system's behavior, and together they provide a complete picture.

1. Sequence Diagram

A Sequence Diagram shows how objects interact over time. It uses lifelines and message arrows to display the exact order of communication. This diagram captures time-based behavior, meaning it tells us what happens first, what happens next, and how messages flow step-by-step. Sequence diagrams are useful for understanding the execution of a specific use case.

2. Activity Diagram

An Activity Diagram models the flow of control in a system. It focuses on actions, decisions, loops, and parallel tasks. It captures the workflow, showing how the system moves from one activity to another. This diagram is helpful for analyzing business processes, complex logic, and user flows. It gives a clear picture of how a process starts, how it branches, and how it ends.

3. State Transition Diagram

A State Transition Diagram explains how an object changes its state based on events. It captures the possible states, transitions, triggers, and conditions. This diagram is useful for systems where behavior depends on internal states, such as ATMs, vending machines, and traffic lights.

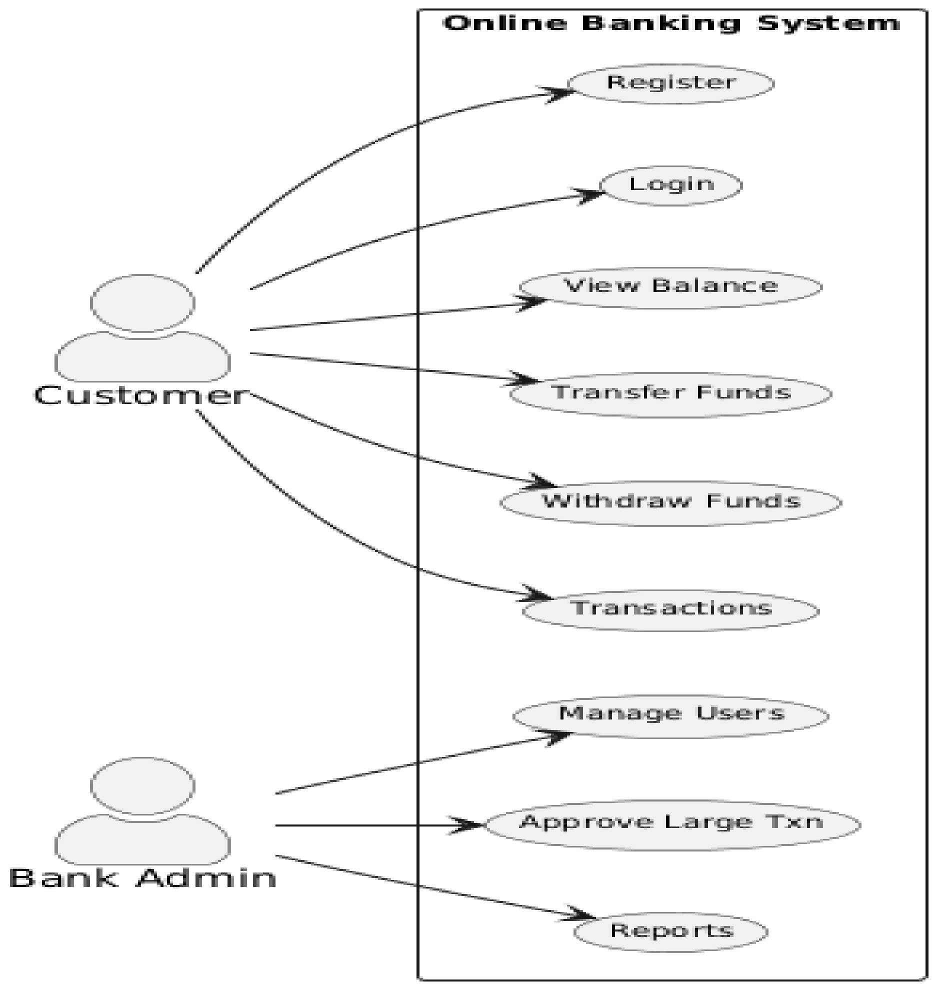
Together, these diagrams show sequence, workflow, and state-based behavior, giving a complete understanding of system dynamics.

2] Design a Use Case Model for an Online Banking System. Include the Use Case Diagram, Actors, and a detailed textual description (narrative) for the "Withdraw Funds" use case

1. Actors

Customer (C) - The Customer is the main actor who interacts with the system to perform personal banking tasks. They can register, log in, view their balance, transfer funds, withdraw funds, and view their transaction history. This actor represents all normal users of the banking application.

Bank Admin (A) - The Bank Admin is responsible for managing backend activities in the system. The Admin performs tasks such as managing user accounts, approving large or suspicious transactions, and generating reports for banking analysis or auditing.



Detailed Narrative – “Withdraw Funds” Use Case

Use Case Name: Withdraw Funds

Primary Actor: Customer

Supporting Actor: Bank Admin (only for large withdrawals)

Goal:

Allow the customer to withdraw money from their bank account securely through the online banking system.

Preconditions:

1. Customer must be registered and successfully logged in.
2. Customer must have sufficient balance in their account.

Postconditions:

1. The withdrawal amount is deducted from the customer’s account.
2. A transaction record is added to the **Transactions** history.

Basic Flow:

1. Customer selects **Withdraw Funds** from the online banking menu.
2. System asks for the withdrawal amount.
3. Customer enters the amount and confirms the request.
4. System verifies available balance and daily withdrawal limit.
5. System processes the withdrawal and updates the account balance.
6. System displays a confirmation message and stores the transaction.

Alternative / Exception Flow:

- If the amount exceeds the daily limit, the system triggers **Approve Large Txn**, requiring Bank Admin approval before processing.
- If funds are insufficient, the system displays “Insufficient Balance” and stops the withdrawal process.

3] Compare and contrast the Activity Diagram and the State Transition Diagram. Provide scenarios where each diagram is the most appropriate tool for modeling behavior.

1. Focus of Each Diagram

- **Activity Diagram** focuses on the *flow of activities* and shows how a process moves from one action to the next. It highlights decisions, parallel tasks, and the overall workflow. Your source explains that activity diagrams show the steps in how a system works and the flow of control between activities.
- **State Transition Diagram** focuses on *how an object changes from one state to another*. It shows states, events, and transitions that cause state changes.

2. What They Represent

- **Activity Diagram** represents tasks like *Browse items → Add to cart → Checkout*, including decision points and branching paths. It is similar to a flowchart and helps describe business or system processes.
- **State Transition Diagram** represents states like *Idle → Processing → Completed* and shows how an event triggers movement from one state to another.

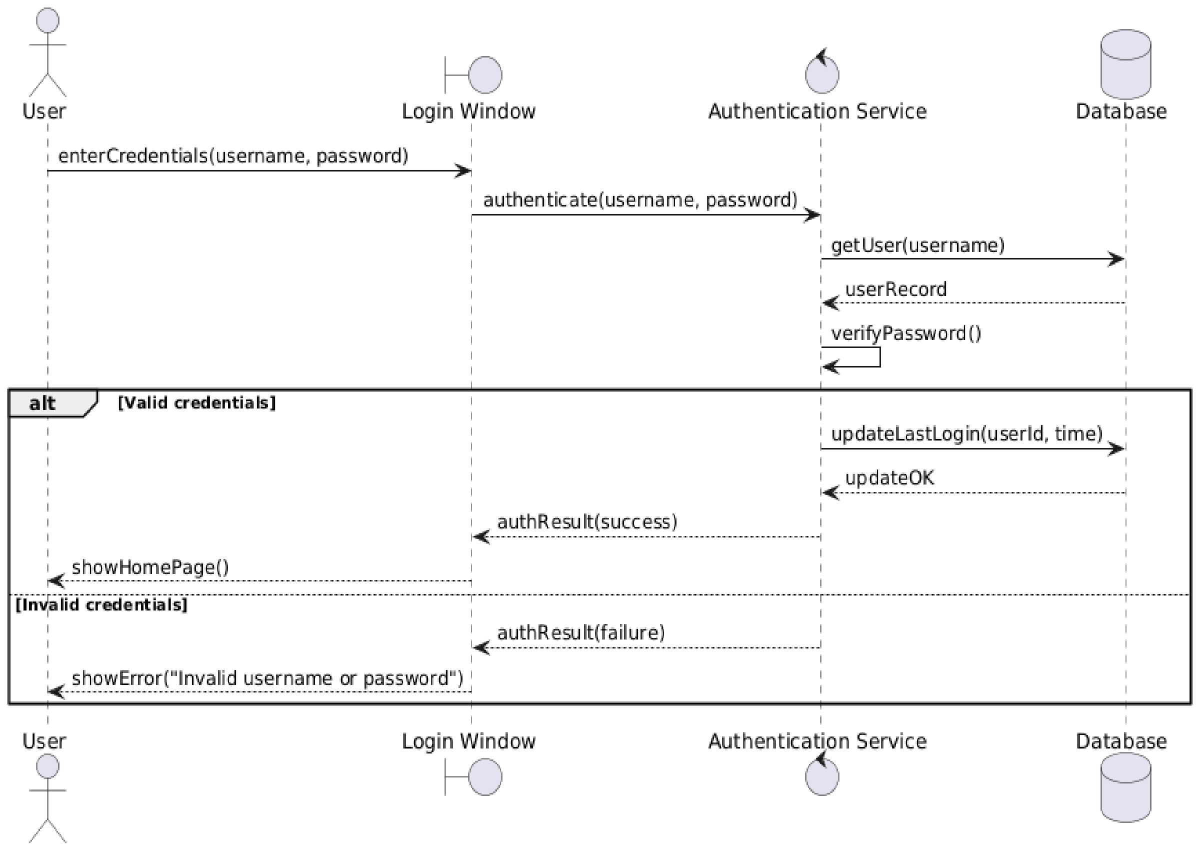
3. How They Model Behavior

- **Activity Diagram** models *dynamic workflow behavior* across multiple steps or users. It is ideal when you want to see the entire flow and how control moves through actions.
- **State Transition Diagram** models *object-centered behavior*, showing how a single object's condition changes over time.

4. Scenarios Where Each Is Appropriate

- **Use Activity Diagram when:**
  - Modeling business workflows (e.g., online purchase flow).
  - Showing processes with decisions, loops, or parallel actions.
  - Explaining user journeys step-by-step (e.g., login → verify → dashboard).
- **Use State Transition Diagram when:**
  - Modeling systems with strict states (ATM states, traffic lights, elevator behavior).
  - Describing object life cycles (order status: placed → shipped → delivered).
  - Showing how events cause changes in internal state.

4] Draw a detailed Sequence Diagram for the scenario where a user logs into a system (User → Login Window → Authentication Service → Database).



5] Evaluate the importance of the Use Case Diagram in the early stages of software development. How does it serve as a bridge between requirements and design?

1. **Use Case Diagrams help gather and clarify user requirements.**
   Your source explains that use case diagrams show how different users interact with the system and help define system behavior clearly.
   This makes it easier to understand what the system must do before any design or coding starts.
2. **They provide a simple visual picture for non-technical stakeholders.**
   The source states that use case diagrams are useful when working with stakeholders who may not understand technical language.
   Because of this, everyone can agree on the system's basic functions early in development.
3. **Use Case Diagrams define the system's boundary.**
   The system boundary in the diagram shows what is part of the system and what is external.
   This prevents misunderstandings and ensures the team knows exactly what features they must build.
4. **They identify actors and their goals, which become functional requirements.**
   Actors such as customers, admins, or external systems help developers understand user goals.
   These goals later convert into specific use cases and detailed requirements.
5. **They act as the foundation for later design diagrams.**
   Each use case becomes the basis for sequence diagrams, activity diagrams, and class diagrams.
   In this way, the use case diagram forms a bridge between what the user needs and how the system will be designed internally.
6. **They support planning and prioritization.**
   Because every use case represents a feature, teams can decide which features are important, which are optional, and which must be built first.

6] Discuss the steps involved in Developing the Use-case Model. Explain the critical difference between Use Cases and Use Case Description.

A. Steps in Developing the Use-Case Model

1. Identify the Actors

Your source explains that actors are external entities such as users or other systems. Identifying them helps us understand **who interacts with the system**.

2. Identify the Main Use Cases

A use case represents a **specific function** of the system (e.g., Login, Transfer Funds). The source states that use cases show how the system behaves for each actor.

3. Define Relationships Between Actors and Use Cases

We connect each actor to the use cases they initiate. This shows **which user performs which function**.

4. Add the System Boundary

The system boundary box defines what is **inside the system** and what is external. This helps clarify the project scope.

5. Refine Use Cases With Include/Extend (If Needed)

Include and extend relationships help separate common or optional behaviors. This makes the model more accurate and modular.

6. Review and Validate With Stakeholders

The source notes that use case diagrams are useful for non-technical users.

B. Critical Difference Between Use Cases and Use Case Descriptions

1. Use Cases (in the diagram)

- Use cases are **ovals** in the diagram showing system functions.
- They give a **high-level view** of what the system does.
- Example: “Login,” “Withdraw Funds,” “View Balance.”

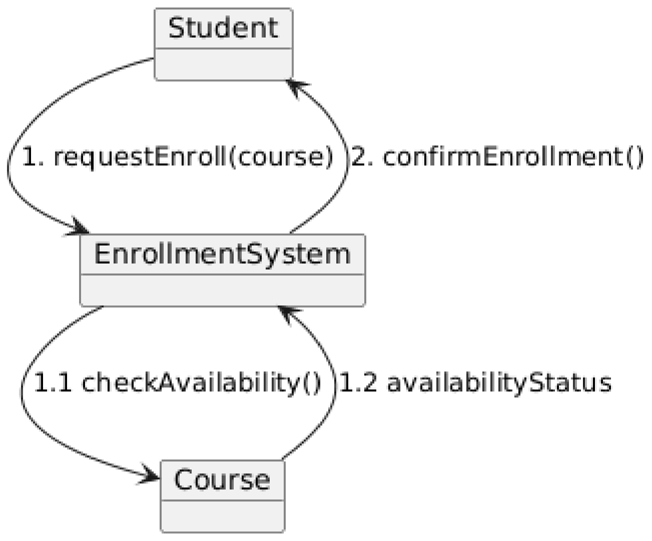
2. Use Case Description (textual narrative)

- A Use Case Description is a **detailed document** that explains how the use case works internally.
- It includes **preconditions, postconditions, triggers, basic flow, alternate flows**, and error handling.
- It shows **step-by-step interaction** between actors and the system.

Critical Difference

- A **Use Case** is a *name* of a function in the diagram.
- A **Use Case Description** explains the *full story* of how that function works.

7] Explain the primary elements and rules for drawing a Collaboration Diagram. Illustrate with a comprehensive example showing object roles and message flow.



Objects (or Object Roles)

Objects represent the parts of the system that interact with each other.
Each object is written with a name such as *Student*, *Course*, *PaymentService*, etc.
Objects are arranged around the page, and their **positions show communication links**, not time.

2. Links Between Objects

A link is a line between two objects showing that they can communicate.
If two objects share a link, they are allowed to exchange messages.
These links show **structural relationships** but do not show any time order.

3. Messages

Collaboration diagrams use **numbered messages** (1, 1.1, 1.2...) to show the order of communication.
Each message has a name like *checkAvailability()* or *confirmEnrollment()*.
Arrows are used to show the **direction of the message flow**.

4. Message Ordering Rules

Numbering messages is required because collaboration diagrams do not use vertical time flow.
Primary messages use whole numbers (1, 2, 3).
Sub-messages use decimal numbering (1.1, 1.2) to show nested calls.

5. Purpose of Collaboration Diagrams

They show **how objects work together** to complete a task.
They emphasize **relationships** between objects more than timing, making them useful for analyzing system structure.

8] Critically discuss the challenges in accurately modeling complex concurrent behavior using Activity Diagrams. What notations are used to handle parallelism?

1. Difficulty in Visualizing Multiple Parallel Tasks Clearly

When many actions happen at the same time, the diagram can become crowded and confusing. Complex flows are hard to read, and users may misunderstand how tasks run together. Your source shows that Activity Diagrams handle parallel actions, but large workflows make clarity difficult.

2. Managing Synchronization Between Concurrent Activities

Activities that start at the same time often need to finish before the next step begins. Synchronizing them correctly can be challenging because the designer must ensure all branches merge properly. Your source explains that “Join” nodes are used for convergence, but incorrect placement can cause logical errors.

3. Representing Complex Decision Logic Alongside Parallel Paths

When decisions and parallel flows appear together, the diagram becomes complicated. Mixing decision nodes with fork/join nodes often leads to mistakes in modeling the actual behavior.

4. Overlapping Workflows Create Visual Complexity

Real-world systems may have many long-running tasks, short tasks, and interruptions. Representing all this in a single diagram makes the model visually heavy and harder for stakeholders to understand.

5. Tool Limitations and Interpretation Differences

Different modeling tools may interpret concurrency differently. Stakeholders may also misunderstand the meaning of parallel paths without proper explanation.

Notations Used to Handle Parallelism

1. Fork Node

A fork splits one flow into **two or more parallel flows**. Your source shows a fork as a **thick horizontal/vertical bar** that creates concurrent activities.

2. Join Node

A join merges **two or more concurrent flows** back into one. All incoming activities must finish before the next step continues. Your source uses the same bar notation for joins.

9] Discuss the process of developing a Use Case Model. Explain identification of actors, identification of use cases, writing use case descriptions, and drawing the use case diagram with an example.

Developing a Use Case Model is one of the first steps in understanding how a new system should work. It helps the development team understand **who will use the system** and **what tasks they need to perform**. The process has four main parts.

1. Identifying the Actors

- Actors are **users or external systems** that interact with the software.
- An actor can be a human (Customer), another system (Payment Gateway), or a device (ATM).
- The goal is to understand **who depends on the system** and **who receives value from it**.

2. Identifying the Use Cases

- A Use Case represents a **task or function** the system must perform for an actor.
- Use Cases answer the question: *What does each actor want to do?*
- Examples: Login, View Balance, Submit Order, Withdraw Funds.
- Use Cases become the **functional requirements** of the system.

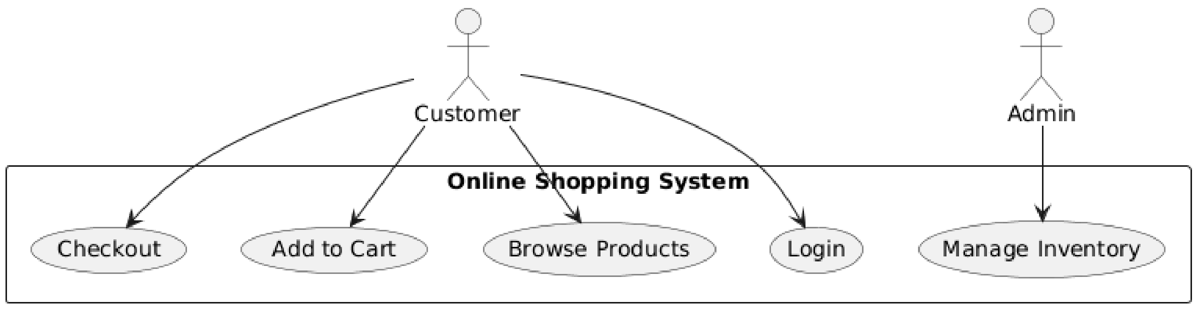
3. Writing Use Case Descriptions

- A Use Case Description explains the **complete story** of how the system performs a function.
- It includes:
    - Use Case Name , Actors , Goal , Preconditions , Postconditions , Basic Flow (normal steps) , Alternative/Exception flows
- This gives developers a **step-by-step explanation** of the feature.

4. Drawing the Use Case Diagram

- The diagram shows actors, use cases, and their relationships using simple symbols (stick figures and ovals).
- The system boundary box separates **what is inside the system** from what is outside.
- It gives a **visual summary** of the system’s functions.

Example : - Online Shopping system


Customer
Admin
Online Shopping System
Checkout
Add to Cart
Browse Products
Login
Manage Inventory