**UNIT IV**

Basic Behavioral Modeling-I: Interactions, Interaction diagrams Use cases, Use case Diagrams, Activity Diagrams. Case Study: Web Application: Vacation Tracking System

**Basic Behavioral Modeling-I**

Basic Behavioral Modeling-I is a concept in software design and system development. It focuses on understanding and representing how a system behaves when interacting with users, other systems, or its environment. The goal is to ensure the system meets the functional needs effectively. This topic is part of object-oriented modeling and design, often used in programming and system architecture.

Behavioral modeling is a way to describe how a system acts or behaves in response to events. It helps answer questions like:

* What actions does the system take when something happens?
* How does the system respond to user input or other triggers?

Think of it as creating a "playbook" for the system to follow in different situations.

Concepts in Basic Behavioral Modeling

**1.Use Case Diagrams**

* A use case is a description of how a user interacts with the system to achieve a goal.
* It focuses on the steps involved in completing a specific task.

**2.Interaction Diagrams**

* These diagrams show how objects in a system communicate with each other.
* Two common types of interaction diagrams:
  + **Sequence Diagrams:** Show the order of messages between objects over time.
  + **Collaboration Diagrams:** Show relationships between objects and how they interact.

**3.State Machine Diagrams**

* These diagrams show how a system or object transitions between states based on events.
* A state is a condition or situation the system is in at a given time.

**4.Activity Diagrams**

* These diagrams represent workflows or processes within the system.
* They focus on actions and decisions.

**Basic Behavioral Modeling Importants**

* **Clarifies Requirements:** Helps designers and developers understand what the system needs to do.
* **Improves Communication:** Provides a clear picture for stakeholders, making it easier to discuss system behavior.
* **Enhances Testing:** Makes it easier to test the system by simulating various scenarios.
* **Facilitates Maintenance:** Models make it easier to understand and update the system later.

**Interactions**

An **interaction** refers to how different parts of a system (like objects, classes, or components) communicate or work together to achieve a specific task. This involves **messages** being sent between these parts, which are actions, commands, or information exchanged.

**In simple terms:**

* Imagine you’re ordering food at a restaurant.
* You interact with the waiter, the waiter interacts with the kitchen staff, and they interact with the chef.
* This series of actions is an **interaction** that leads to you getting your food.

In **UML (Unified Modeling Language)**, interactions are visualized using **interaction diagrams**.

These diagrams show:

1. Who is involved in the interaction? (participants/objects)
2. What messages are being exchanged?
3. The sequence or flow of communication.

**Use of Interaction Diagrams**

1. **Understand Communication:** Helps designers see how parts of the system talk to each other.
2. **Identify Dependencies:** Shows which components rely on others.
3. **Simplify Design:** Breaks down complex workflows into clear, visual steps.
4. **Improve Testing:** Makes it easy to simulate interactions for testing.

**Interaction Diagrams**

Interaction diagrams depict interactions of objects and their relationships. They also include the messages passed between them. There are two types of interaction diagrams

There are two main types of interaction diagrams:

**Interaction Diagrams**

**1.Sequence Diagrams**

**2.Collaboration Diagrams**

**1.Sequence Diagrams**

A Sequence Diagram is a type of interaction diagram in UML (Unified Modeling Language). It shows how objects interact with each other in a specific scenario by visualizing the sequence of messages exchanged over time.

It’s like a movie script that details:

* Who is involved? (objects or participants)
* What actions/messages are exchanged?
* In what order do the actions happen?

**Components of a Sequence Diagram**

1. **Participants (or Objects):**
   * Represented as rectangles with a name, such as "User" or "System."
   * These are the entities that interact in the process.
   * Each participant has a **lifeline** (a dashed vertical line).
2. **Lifelines:**
   * Vertical dashed lines drawn below each participant.
   * Represent the participant’s lifespan during the interaction.
3. **Messages:**
   * Represented as arrows between lifelines.
   * Indicate communication, such as method calls or responses.
   * Types of messages:
     + **Synchronous message:** Requires a response (solid arrow with a filled arrowhead).
     + **Asynchronous message:** Does not need a response (solid arrow with an open arrowhead).
     + **Reply message:** A dashed arrow returning a result.
4. **Activation Bars:**
   * Thin rectangles on lifelines.
   * Show when a participant is performing an action or process.
5. **Return Messages:**
   * Dashed arrows pointing back to the sender, representing a response.
6. **Optional Elements:**
   * **Loops:** Show repeated actions.
   * **Conditions:** Show decisions (e.g., "if-else" logic).
   * **Fragments:** Highlight specific scenarios like alternatives or parallel processes.

**Simple Example: Online Shopping**

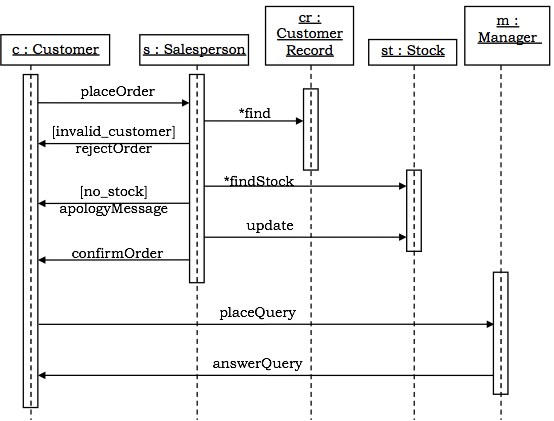
Steps:

1. The user places an order.
2. The system checks stock availability.
3. The payment is processed.
4. The order is confirmed.

**Sequence Diagram:**

* **User** sends a message to **System**: "Place Order."
* **System** sends a message to **Inventory System**: "Check Stock."
* **System** sends a message to **Payment Gateway**: "Process Payment."
* **System** replies to **User**: "Order Confirmed."

**Example** − A sequence diagram for the Automated Trading House System is shown in the following figure.



**2. Collaboration Diagrams**

Collaboration diagrams are interaction diagrams that illustrate the structure of the objects that send and receive messages

**Features**

* + Emphasize the **relationships** between objects rather than time order.
  + Show **links** (lines) between objects and the **messages** exchanged.

**Simple Example: ATM Withdrawal**

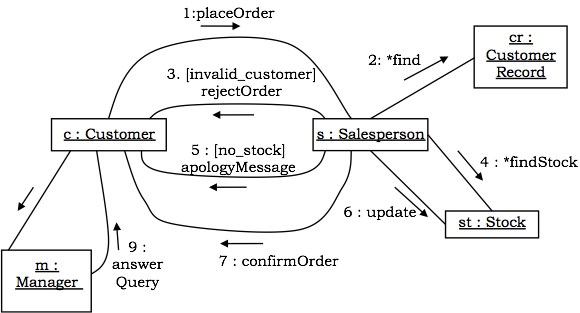
**Participants:**

* **Customer**
* **ATM**
* **Bank Server**

**Communication Diagram:**

* **Customer** → **ATM**: "Insert Card."
* **ATM** → **Bank Server**: "Verify PIN."
* **Bank Server** → **ATM**: "PIN Verified."
* **ATM** → **Customer**: "Dispense Cash."

**Example** − Collaboration diagram for the Automated Trading House System is illustrated in the figure below.



**Differences Between Sequence and Communication Diagrams**

|  |  |  |
| --- | --- | --- |
| Feature | Sequence Diagram | Communication Diagram |
| Focus | Order of messages over time | Relationships and object links |
| Structure | Vertical timeline | Network-like structure |
| Best For | Understanding message flow | Analyzing system connections |

**State – Chart Diagrams**

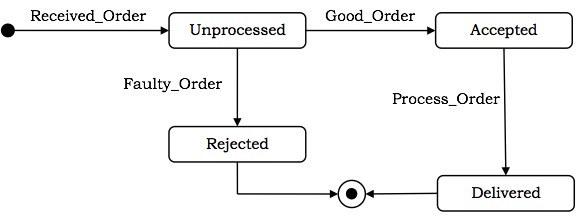
A state–chart diagram shows a state machine that depicts the control flow of an object from one state to another. A state machine portrays the sequences of states which an object undergoes due to events and their responses to events.

**State–Chart Diagrams comprise of −**

* **States: Simple or Composite**
* **Transitions between states**
* **Events causing transitions**
* **Actions due to the events**

State-chart diagrams are used for modeling objects which are reactive in nature.

**Example** In the Automated Trading House System, let us model Order as an object and trace its sequence. The following figure shows the corresponding state–chart diagram.



**Activity Diagram**

An Activity Diagram is a visual representation of a process or workflow in a system. It shows the steps involved in completing a task, the decisions made along the way, and the flow of actions.

Think of it like a flowchart that explains how a task progresses from start to finish. It’s used in software development to:

* **Understand workflows**
* **Identify decisions and parallel processes**

**Components of an Activity Diagram**

1. **Start Node:**
   * Represented by a filled black circle.
   * Indicates where the process begins.
2. **Activity:**
   * Represented by a rectangle with rounded corners.
   * Describes a task or action in the process.
   * Example: "Login," "Select Product," "Make Payment."
3. **Transition (Arrow):**
   * Shows the flow from one activity to another.
   * Indicates the sequence of steps.
4. **Decision Node:**
   * Represented by a diamond shape.
   * Shows a decision point where multiple paths are possible.
   * Example: "Is payment successful?" Yes → Continue, No → Retry.
5. **Merge Node:**
   * Also represented by a diamond.
   * Combines multiple paths back into one.
6. **Fork Node:**
   * Represented by a thick horizontal or vertical bar.
   * Indicates the start of parallel activities (tasks happening at the same time).
7. **Join Node:**
   * Represented by the same thick bar as the fork node.
   * Combines parallel activities into one flow.
8. **End Node:**
   * Represented by a circle with a black border and filled center.
   * Indicates the process is complete.

**Symbols in Activity Diagrams**

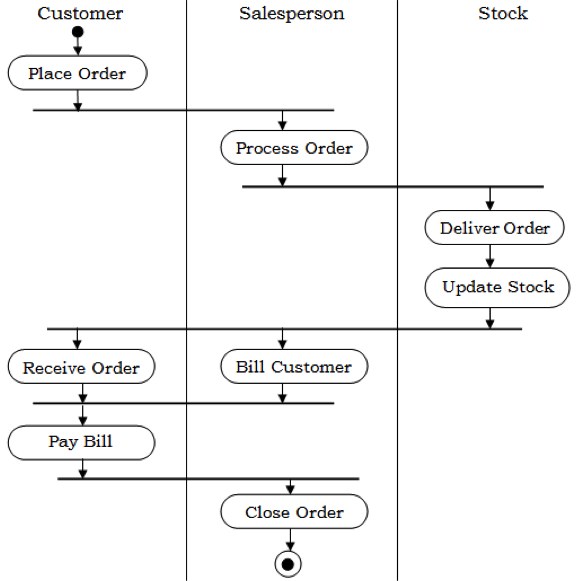
|  |  |
| --- | --- |
| Symbol | Meaning |
| Circle (Start Node) | Starting point of the process. |
| Rounded Rectangle | An activity (task or action). |
| Arrow | Flow of the process or transition. |
| Diamond (Decision) | A decision point (e.g., Yes/No). |
| Thick Bar (Fork/Join) | Start or end of parallel activities. |
| Circle with Border | Ending point of the process. |

**Advantages of Activity Diagrams**

1. **Simplify Complex Processes:**
   * Break down workflows into easy-to-understand steps.
2. **Highlight Decisions:**
   * Clearly show where choices need to be made.
3. **Support Parallel Tasks:**
   * Visualize tasks that can occur simultaneously.
4. **Communicate with Stakeholders:**
   * Easy to explain to both technical and non-technical audiences.

**Example**

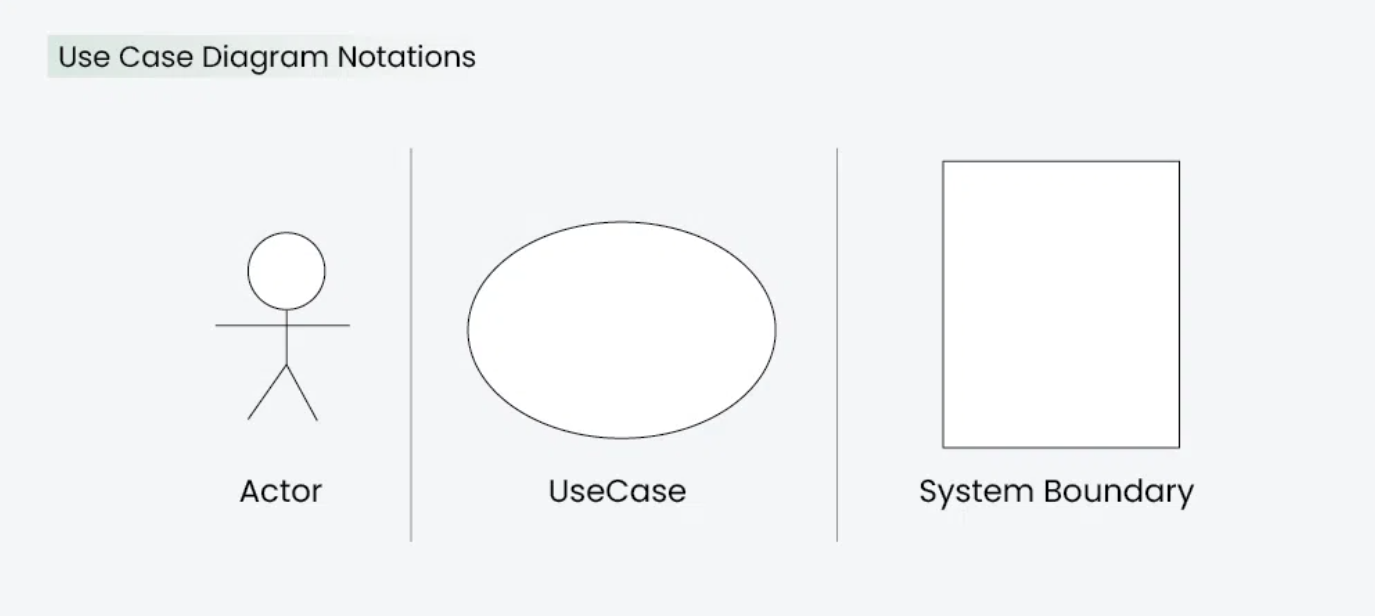
The following figure shows an activity diagram of a portion of the Automated Trading House System.



**Activity Diagrams** are powerful tools for visualizing workflows and processes. They are simple to understand, highly effective for communication, and essential for designing and analyzing systems. By showing step-by-step actions, decisions, and parallel tasks, they make it easier to understand and improve how a system operates.

**Use Case Diagram**

A Use Case Diagram is a type of Unified Modeling Language (UML) diagram that represents the interaction between actors (users or external systems) and a system under consideration to accomplish specific goals. It provides a high-level view of the system's functionality by illustrating the various ways users can interact with it.



**Use cases are useful for:**

* Understanding user needs: They focus on what users want to achieve.
* Defining system functionality: They provide a clear description of system behaviors.
* Communication: They act as a bridge between stakeholders (e.g., business, developers, and users)

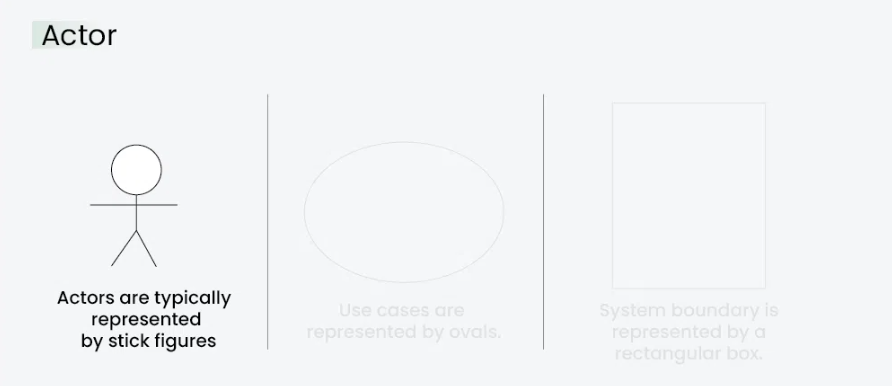
**Use Case Diagram Notations**

UML notations provide a visual language that enables software developers, designers, and other stakeholders to communicate and document system designs, architectures, and behaviors in a consistent and understandable manner.

1. **Actors**: People or systems interacting with the system.
2. **System**: The software or process under consideration.
3. **Use Case**: The specific function or task that fulfills a user goal.
4. **Relationships**: How actors and the system interact.

**1. Actors**

Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes. Proper identification and understanding of actors are crucial for accurately modeling system behavior.



**2. Use Cases**

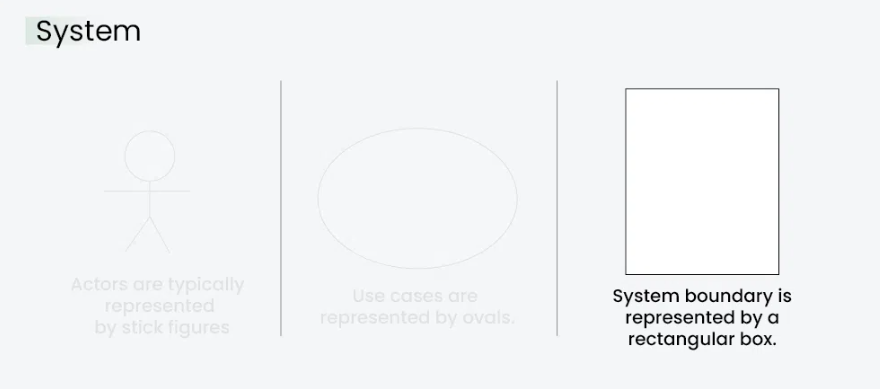
Use cases are like scenes in the play. They represent specific things your system can do. In the online shopping system, examples of use cases could be "Place Order," "Track Delivery," or "Update Product Information".Use cases are represented by ovals.



**3. System Boundary**

The system boundary is a visual representation of the scope or limits of the system you are modeling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it. The system boundary is typically represented by a rectangular box that surrounds all the use cases of the system.

* The purpose of system boundary is to clearly outlines the boundaries of the system, indicating which components are internal to the system and which are external actors or entities interacting with the system.



**Use Case Diagram Relationships**

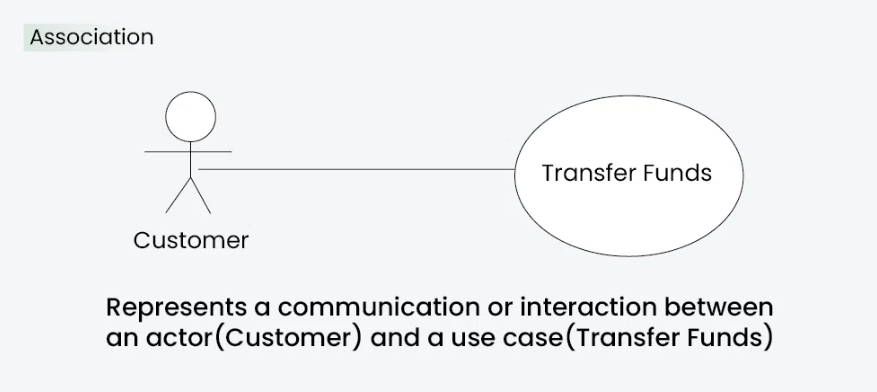
In a Use Case Diagram, relationships play a crucial role in depicting the interactions between actors and use cases. These relationships provide a comprehensive view of the system's functionality and its various scenarios. Let's delve into the key types of relationships and explore examples to illustrate their usage.

**1. Association Relationship**

TheAssociation Relationship represents a communication or interaction between an actor and a use case. It is depicted by a line connecting the actor to the use case. This relationship signifies that the actor is involved in the functionality described by the use case.

**Example: Online Banking System**

* **Actor:** Customer
* **Use Case:** Transfer Funds
* **Association:** A line connecting the "Customer" actor to the "Transfer Funds" use case, indicating the customer's involvement in the funds transfer process.

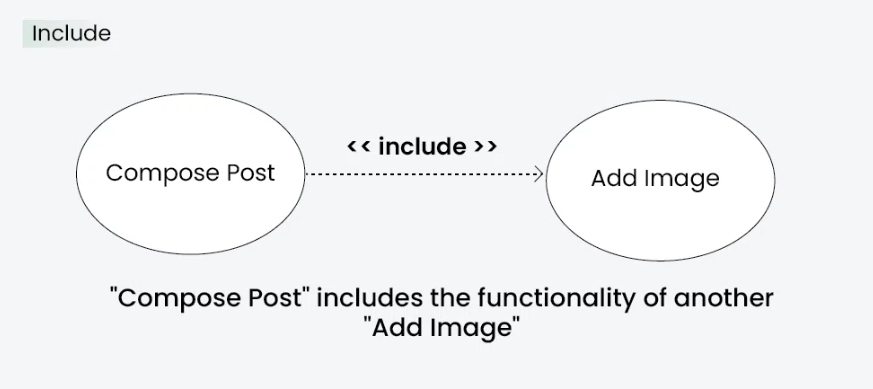
****

**2. Include Relationship**

The Include Relationship indicates that a use case includes the functionality of another use case. It is denoted by a dashed arrow pointing from the including use case to the included use case. This relationship promotes modular and reusable design.

**Example: Social Media Posting**

* **Use Cases:** Compose Post, Add Image
* **Include Relationship:** The "Compose Post" use case includes the functionality of "Add Image." Therefore, composing a post includes the action of adding an image.

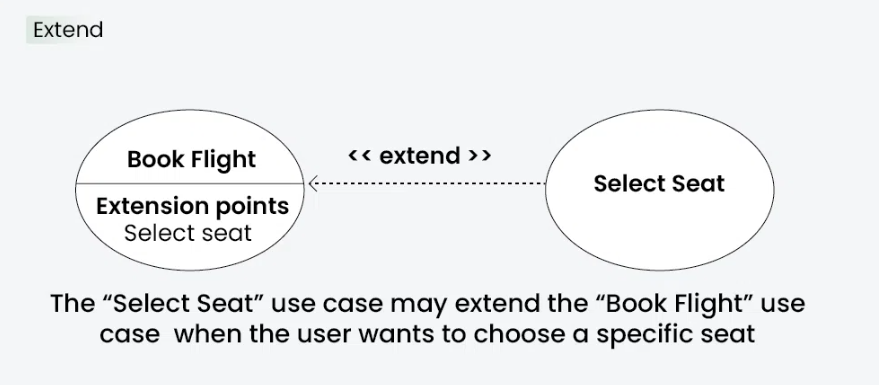


**3. Extend Relationship**

The Extend Relationship illustrates that a use case can be extended by another use case under specific conditions. It is represented by a dashed arrow with the keyword "extend." This relationship is useful for handling optional or exceptional behavior.

**Example: Flight Booking System**

* **Use Cases:** Book Flight, Select Seat
* **Extend Relationship:** The "Select Seat" use case may extend the "Book Flight" use case when the user wants to choose a specific seat, but it is an optional step.

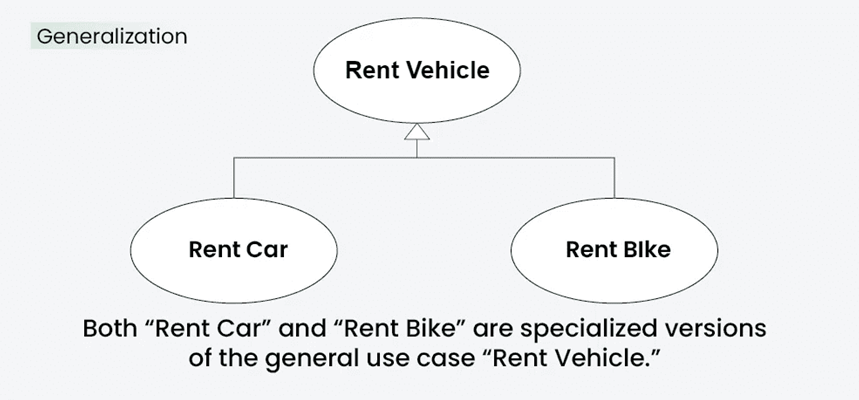


**4. Generalization Relationship**

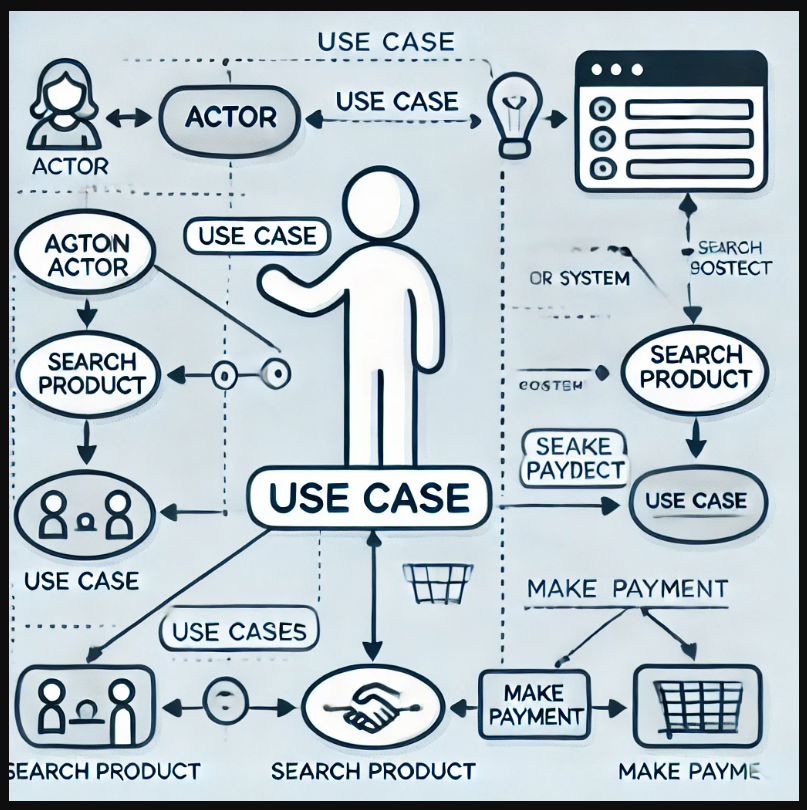
The Generalization Relationship establishes an "is-a" connection between two use cases, indicating that one use case is a specialized version of another. It is represented by an arrow pointing from the specialized use case to the general use case.

**Example: Vehicle Rental System**

* **Use Cases:** Rent Car, Rent Bike
* **Generalization Relationship:** Both "Rent Car" and "Rent Bike" are specialized versions of the general use case "Rent Vehicle."



Think of a use case as a list of things someone might do with a system. The diagram is like a map showing who does what. For example, in an online store, the "Customer" can search for products, buy them, or check their orders. The "Payment System" only helps process payments. This keeps things clear and easy to understand for everyone



**Case Study: Vacation Tracking System (Web Application)**

A Vacation Tracking System is a web application that helps organizations manage employee leave requests, approvals, and records. Below is a detailed explanation of the system in simple way

**Purpose of the System**

The Vacation Tracking System is designed to:

* Let employees request leave.
* Help managers review and approve or reject leave requests.
* Keep a record of employee leave balances and history.

**Main Actors**

1. **Employee**:
   * Requests leave.
   * Views leave balance and request status.
2. **Manager**:
   * Approves or rejects leave requests.
   * Views team leave schedules.
3. **Administrator (Admin)**:

* Sets up employee accounts.
* Configures leave policies (e.g., annual leave limit).

**Use Cases (Tasks or Functions)**

1. **Submit Leave Request**:  
   Employees fill out a form to request leave, specifying dates and leave type (e.g., vacation, sick leave).
2. **View Leave Balance**:  
   Employees check their remaining leave balance.
3. **Approve or Reject Leave**:  
   Managers review leave requests and approve or reject them based on team schedules and policies.
4. **View Team Schedule**:  
   Managers see who is on leave to avoid overlaps.
5. **Configure Leave Policies**:  
   Admin sets rules like annual leave limits or holidays.
6. **Generate Reports**:  
   Admins create reports on employee leave usage.

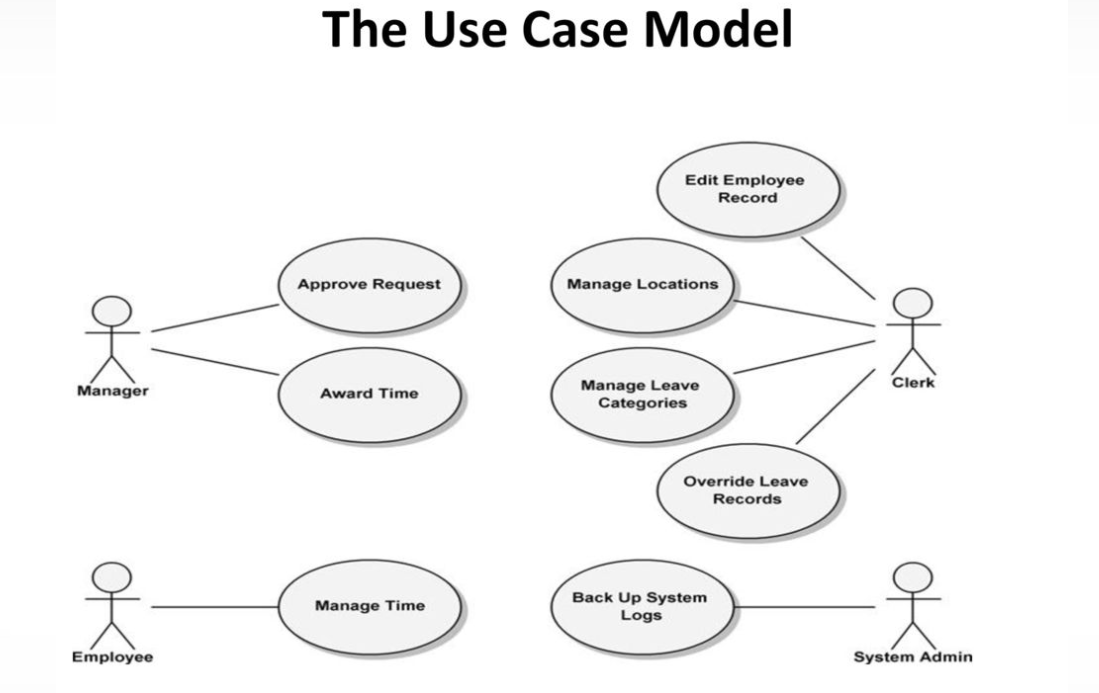
**Use Case Diagram**

**Actors and Use Cases**:

1. **Employee** (Stick Figure):
   * Submits Leave Request (Oval).
   * Views Leave Balance (Oval).
2. **Manager** (Stick Figure):
   * Approves or Rejects Leave (Oval).
   * Views Team Schedule (Oval).
3. **Admin** (Stick Figure):
   * Configures Leave Policies (Oval)
   * Generates Reports (Oval).

**4.Clerk**

A person from the HR department who can view employees' personal data and is responsible for keeping all employee information in the HR systems accurate and updated. An HR clerk can add or remove almost any record in the system.



**Relationships**:

* Lines connect employees, managers, and admins to the use cases they perform.

**System Boundary**:  
A rectangle encloses all use cases to show they belong to the Vacation Tracking System.

**Workflow Example**

1. **Employee Workflow**:
   * John logs in and requests vacation from Jan 15 to Jan 20.
   * The system checks if John has enough leave days available.
   * The request is sent to John’s manager.
2. **Manager Workflow**:
   * Sarah (the manager) logs in and sees John’s request.
   * She checks the team schedule to ensure no conflicts.
   * Sarah approves the leave request.
3. **Admin Workflow**:
   * The admin updates company holiday rules for the new year.
   * The system automatically adjusts leave balances for all employees.

**Benefits of the System**

1. **Efficiency**: Automates leave request and approval processes.
2. **Transparency**: Employees and managers can easily track leave balances and statuses.
3. **Accuracy**: Reduces errors in tracking leave balances.
4. **Centralization**: Keeps all leave records in one place.

This case study simplifies the key components of a Vacation Tracking System and how it supports different users.