

Name of Student: Pushkar Sane		
Roll Number: 45		Tutorial Number: 4
Title of Tutorial: UML Diagrams (Use Case Diagram)		
DOP: 25-09-2023		DOS: 25-09-2023
CO Mapped: CO1, CO2, CO3	PO Mapped: PO1, PO3, PO6	Signature:

Tutorial No. 4

Aim: UML Diagrams (Use Case Diagram).

Description:

- **Introduction**

- a. **The Role of Diagrams in System Modelling:**

System modeling is an essential part of software engineering and other disciplines. Diagrams play a pivotal role in system modeling as they provide a visual representation of complex systems, making it easier to understand, analyze, and communicate. Among these diagrams, UML Use Case Diagrams stand out as powerful tools for capturing and visualizing how a system interacts with its external entities.

- b. **Purpose and Significance of Use Case Diagrams:**

Use Case Diagrams serve to understand, specify, and document the functional requirements of a system. They focus on capturing the system's interactions with external entities called actors, and the specific actions or use cases that the system performs to achieve its goals. Use Case Diagrams are not just valuable during system design but also serve as a foundation for communication between stakeholders, developers, and designers.

- **Benefits of Using UML**

Unified Modelling Language (UML) provides a standardized approach to creating and interpreting diagrams like Use Case Diagrams. Its advantages include a common notation system, improved communication among teams, and the ability to model complex systems effectively.

- a. **Key Concepts:**

Understanding the fundamental concepts behind Use Case Diagrams is crucial for effective modeling.

- b. **Actors: Who Interacts with the System?**

Actors represent external entities that interact with the system. They can be users, other systems, or even physical devices. Actors are depicted in Use Case Diagrams as stick figures. Each actor has a specific role or responsibility within the system.

c. Use Cases: What the System Does

Use Cases represent specific functionalities or behaviors of the system. They describe the actions that the system performs to achieve a particular goal. Use Cases are represented as ovals in Use Case Diagrams, and they typically have clear and descriptive names.

d. Relationships: How Actors and Use Cases Connect



Relationships in Use Case Diagrams show how actors and use cases are connected. The primary relationship types include association, generalization, include, and extend. These relationships depict how actors interact with the system and under what conditions.

e. System Boundary: Defining the Scope

The system boundary encloses all actors and use cases, delineating the scope of the system. Anything within the boundary is considered part of the system, while entities outside the boundary are external to the system.

• Notations and Symbols:

Understanding the notations and symbols used in Use Case Diagrams is essential for creating and interpreting them accurately. Here, The use cases are represented by either circles or ellipses.

Name	Symbol	Description
Actor		Actor represents a user or another system that will interact with the system you are modeling.
Use Case		A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Association	_____	Association between use cases.
Include Relationship	----->	Include relationship between the use cases

● Creating Use Case Diagrams

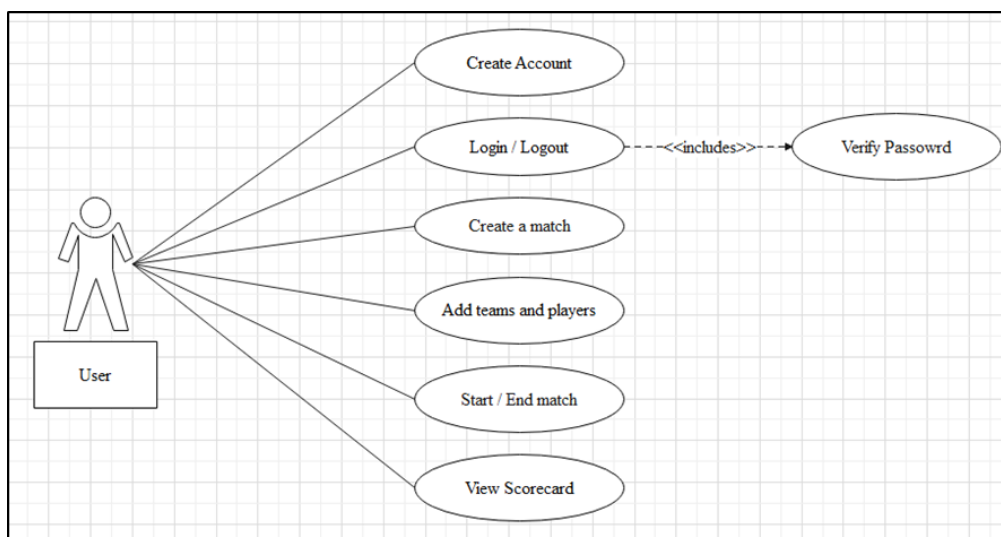
Creating a Use Case Diagram involves several steps, which can be illustrated through an example.

Steps to Create a Use Case Diagram

1. **Identify Actors:** Begin by identifying all the external entities (actors) that interact with the system.
2. **Define Use Cases:** Determine the specific functionalities (use cases) that the system offers to its actors.
3. **Establish Relationships:** Connect actors to the relevant use cases using appropriate relationship notations.
4. **Draw the System Boundary:** Enclose all actors and use cases within the system boundary to define the scope.

● Problem Statement

To create a live cricket scoring app. Here, various types of functions are provided such as ball-by-ball scoring, professional scorecard etc.



- **Best Practices and Guidelines**

To ensure the effectiveness of Use Case Diagrams, consider the following best practices:

1. **Simplicity and Clarity:** Keep your diagrams simple and focused. Avoid overcomplicating them with unnecessary details. The goal is to enhance understanding, not confuse stakeholders.
2. **Consistent Naming Conventions:** Use clear and consistent naming conventions for actors and use cases. Descriptive names make it easier for stakeholders to understand the diagram's content.
3. **Proper Use of Relationships:** Select the appropriate relationship type (association, generalization, include, or extend) to accurately represent the interactions between actors and use cases. Ensure that relationships reflect real world interactions.

- **Applications Across Industries**

Use Case Diagrams find applications across various industries beyond software development:

1. **Software Development:** In software engineering, Use Case Diagrams are invaluable for capturing and documenting system requirements. They guide the design and development process by providing a clear understanding of user interactions.
2. **Business Process Modelling:** Business analysts use Use Case Diagrams to model and analyze complex business processes. They help identify interactions between stakeholders, systems, and processes, leading to improved business strategies.
3. **System Design and Evaluation:** Use Case Diagrams are also used in system design beyond software. For example, they aid in designing hardware systems or evaluating the effectiveness of organizational processes.

Conclusion:

In conclusion, UML Use Case Diagrams are powerful tools for visualizing and understanding how a system interacts with its external entities. They facilitate effective communication, help gather and document requirements, and serve as a foundation for system design and development. Whether used in software engineering, business analysis, or other fields, Use Case Diagrams remain highly relevant for modelling and designing complex systems. Understanding their principles and best practices can greatly enhance the quality and efficiency of system development projects.