Experiment no. 3

Aim: Sketch a UML Use-Case Diagram for any Software Management System (e.g., Library Management System).

Tools: Lucidchart, Draw.io, MS Visio, or StarUML

Theory:

A Use-Case Diagram is a behavioral diagram defined by UML that captures the functional requirements of a system. It shows interactions between actors (users or external systems) and the system's use cases (functionalities). Use-case diagrams are helpful in understanding what the system should do and who interacts with it.

Notations:

• Actor: Represented as a stick figure (e.g., User, Admin, System)

• Use Case: Represented as an oval (e.g., Login, Search Book)

• System Boundary: Represented as a rectangle enclosing all use cases

• Relationships: Association (solid line), Include/Extend (dashed arrows with <<include>> or <<extend>>)

UML (Unified Modeling Language) is a standardized modeling language widely used in software engineering. It helps in visualizing, specifying, constructing, and documenting the components of a software system. Think of UML as a blueprint language for designing and describing a system before (and even while) it is built.

Why UML Diagrams are Important

- They provide a visual representation of the system.
- Help in understanding complex systems easily.
- Serve as a common communication medium among developers, designers, testers, and stakeholders.
- Useful in software documentation and maintenance.

Types of UML Diagrams

UML diagrams are broadly divided into Structural and Behavioral diagrams.

1. Structural Diagrams

These show the static structure of the system – the elements and how they relate to each other. Examples:

- Class Diagram: Shows classes, attributes, methods, and relationships.
- Object Diagram: Shows instances of classes at a particular moment.
- Component Diagram: Depicts how components are connected.
- Deployment Diagram: Shows how software is deployed on hardware.

2. Behavioral Diagrams

These show the dynamic behavior of the system – how it acts and reacts over time. Examples:

- Use Case Diagram: Shows interactions between users (actors) and the system.
- Sequence Diagram: Shows how objects interact through messages in a sequence.
- Activity Diagram: Shows workflow or activities in a system.
- State Diagram: Shows states of an object and transitions triggered by events.

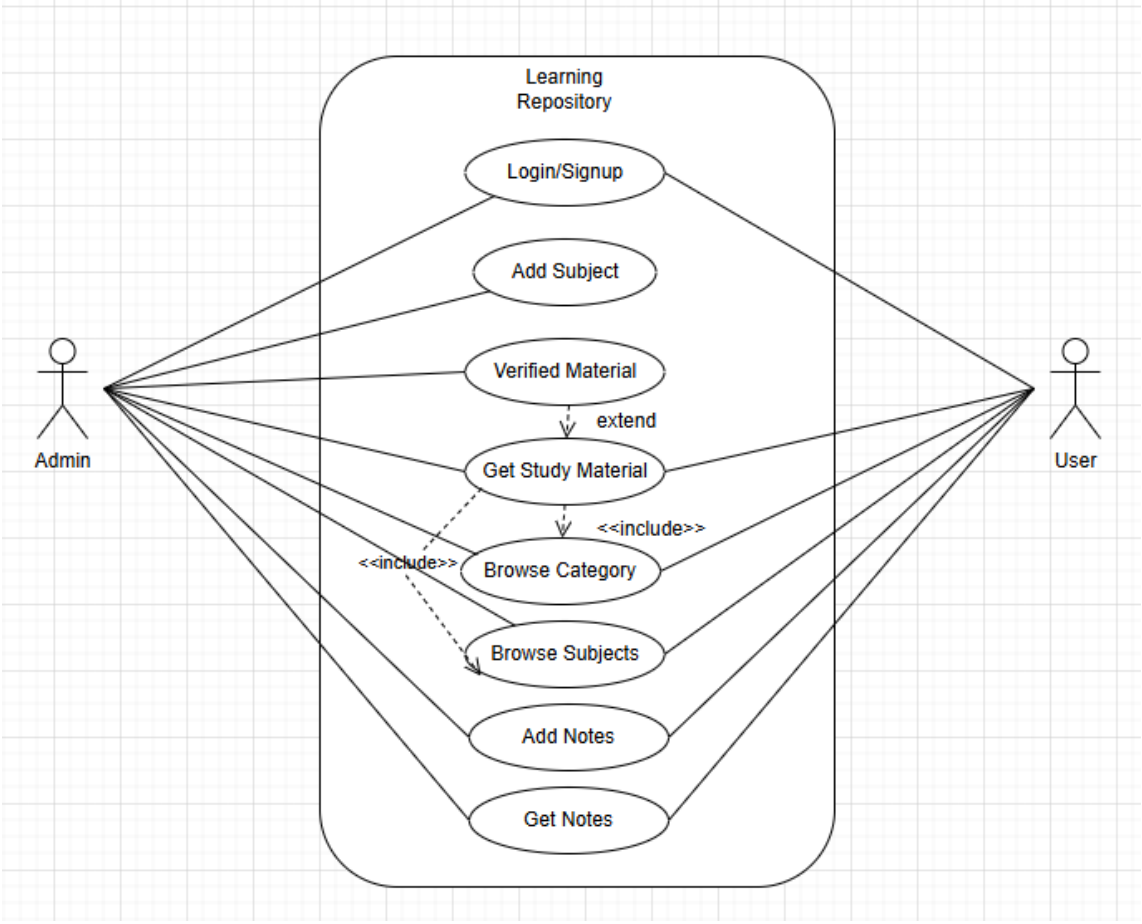
Key Features

- UML is language-independent and platform-independent.
- It is standardized by OMG (Object Management Group).
- It supports iterative and incremental development.

Procedure:

- Understand the system requirements and identify main functionalities.
- Identify actors interacting with the system.
- List use cases (functional requirements) of the system.
- Draw the system boundary and place use cases inside.
- Connect actors with their corresponding use cases using associations.
- Highlight any include or extend relationships if applicable.

UML Use Case Diagram for Learning Repository



Learning Outcomes:

LO1: Identify actors and use cases for a given system.

LO2: Draw a UML Use-Case Diagram to represent system functionality.

LO3: Understand how actors interact with system use cases.

Course Outcomes:

Students will be able to design a UML Use-Case Diagram representing the functional requirements of a system.

Conclusion:

Successfully designed a UML Use-Case Diagram for a chosen Software Management System, clearly representing the interactions between actors and the system's use cases.

Theory Questions:

1. What is the purpose of a UML Use-Case Diagram?

2. Differentiate between an actor and a use case with examples.

Case-Studies/Open-Ended Questions:

1. How would you update the Use-Case Diagram if new functionality (e.g., online payment) is added?

2. How can a Use-Case Diagram help in validating requirements with a client?

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				