**Practical No. 4**

**Aim:** [Modeling UML Use Case Diagrams and Capturing Use Case Scenarios](http://vlabs.iitkgp.ernet.in/se/3/)

**Theory:**

1. **Use case diagrams**

Use case diagrams belong to the category of behavioural diagram of UML diagrams. Use case diagrams aim to present a graphical overview of the functionality provided by the system. It consists of a set of actions (referred to as use cases) that the concerned system can perform one or more actors, and dependencies among them.

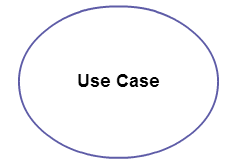
1. **Actor**

Actor in a use case diagram is any entity that performs a role in one given system. This could be a person, organization or an external system and usually drawn like skeleton shown below.



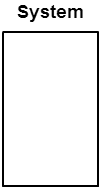
1. **Use Case**

A use case is simply a functionality provided by a system. A use case **represents a function or an action within the system**. It’s drawn as an oval and named with the function.



1. **Subject**

Subject is simply the system under consideration. Use cases apply to a subject. The system is used to define the scope of the use case and drawn as a rectangle.



1. **Graphical Representation**

An actor is represented by a stick figure and name of the actor is written below it. A use case is depicted by an ellipse and name of the use case is written inside it. The subject is shown by drawing a rectangle. Label for the system could be put inside it. Use cases are drawn inside the rectangle, and actors are drawn outside the rectangle, as shown in figure 01

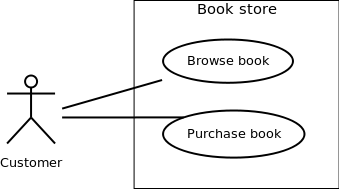


Figure - 01: A use case diagram for a book store

1. **Association between Actors and Use Cases**

A use case is triggered by an actor. Actors and use cases are connected through binary associations indicating that the two communicates through message passing.

An actor must be associated with at least one use case. Similarly, a given use case must be associated with at least one actor. Associations among the actors are usually not shown. However, one can depict the class hierarchy among actors.

1. **Use Case Relationships**

Three types of relationships exist among use cases:

**Include relationship**

**Extend relationship**

**Use case generalization**

**Include Relationship**

Include relationships are used to depict common behaviour that are shared by multiple use cases. This could be considered analogous to writing functions in a program in order to avoid repetition of writing the same code. Such a function would be called from different points within the program.

Example

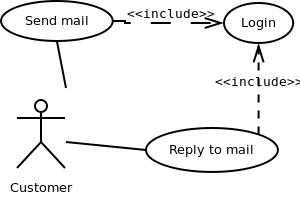


Figure - 02: Include relationship between use cases

Include relationship is depicted by a dashed arrow with a «include» stereotype from the including use case to the included use case.

**Extend Relationship**

Use case extensions are used to depict any variation to an existing use case. As the name implies it extends the base use case and adds more functionality to the system.

**Example**:

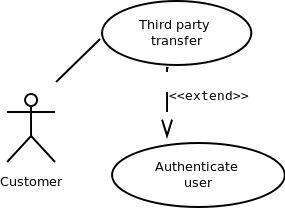


Figure - 03: Extend relationship between use cases

Extend relationship is depicted by a dashed arrow with a «extend» stereotype from the extending use case to the extended use case.

**Generalization Relationship**

Generalization relationship are used to represent the inheritance between use cases. A derived use case specializes some functionality it has already inherited from the base use case.

Example:

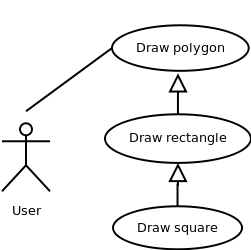


Figure - 04: Generalization relationship among use cases

Generalization relationship is depicted by a solid arrow from the specialized (derived) use case to the more generalized (base) use case.

1. **Identifying Actors**

Given a problem statement, the actors could be identified by asking the following questions :

* Who gets most of the benefits from the system? (The answer would lead to the identification of the primary actor)
* Who keeps the system working? (This will help to identify a list of potential users)
* What other software / hardware does the system interact with?
* Any interface (interaction) between the concerned system and any other system?

1. **Identifying Use cases**

Once the primary and secondary actors have been identified, we have to find out their goals i.e. what are the functionality they can obtain from the system. Guidelines for drawing Use Case diagrams

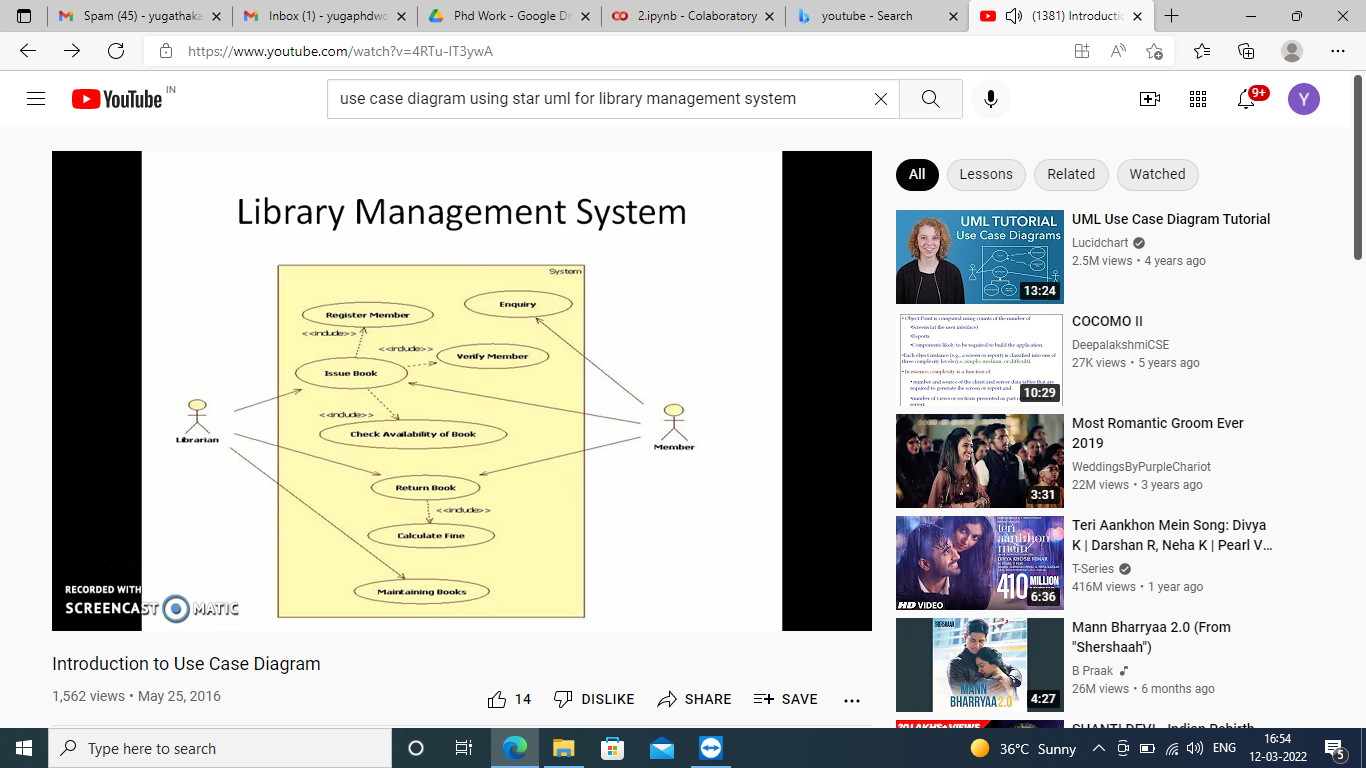
Following general guidelines could be kept in mind while trying to draw a use case diagram:

* Determine the system boundary
* Ensure that individual actors have well-defined purpose
* Use cases identified should let some meaningful work done by the actors
* Associate the actors and use cases -- there shouldn't be any actor or use case floating without any connection
* Use include relationship to encapsulate common behaviour among use cases , if any

**Case Study:**

* Draw a use case diagram for the following problem

Consider a library, where a member can perform two operations: issue book return book and mainlining books. A book is issued to a member only after verifying his credentials. Draw a use case diagram for the problem.



Conclusion: Thus we have studied different use case scenarios and Modeling Use Case Diagram for library management system.