UML Diagrams

- A UML diagram is designed with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes in order to better understand, maintain or document information about the system.
- UML is a modern approach of modelling based on diagrammatic representation of software components.
- There are two categories of UML diagrams:

- Behavioural UML Diagram.
- Structural UML Diagram.

1. Behavioural Diagram

- Use-case Diagram
- Activity Diagram
- Sequence Diagram
- Timing Diagram

a. Use-case Diagram

- A use case diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that visually represents the interactions between users (actors) and a system, focusing on the functional requirements of the system from the users' perspective.
- It provides a high-level view of the system's functionality and the external entities that interact with it.
- Key components of a use case diagram include:

❖ **Actors:**

❑ An actor portrays any entity (or entities) that perform certain roles in a given system.

❑ The different roles the actor represents are the actual business roles of users in a given system.

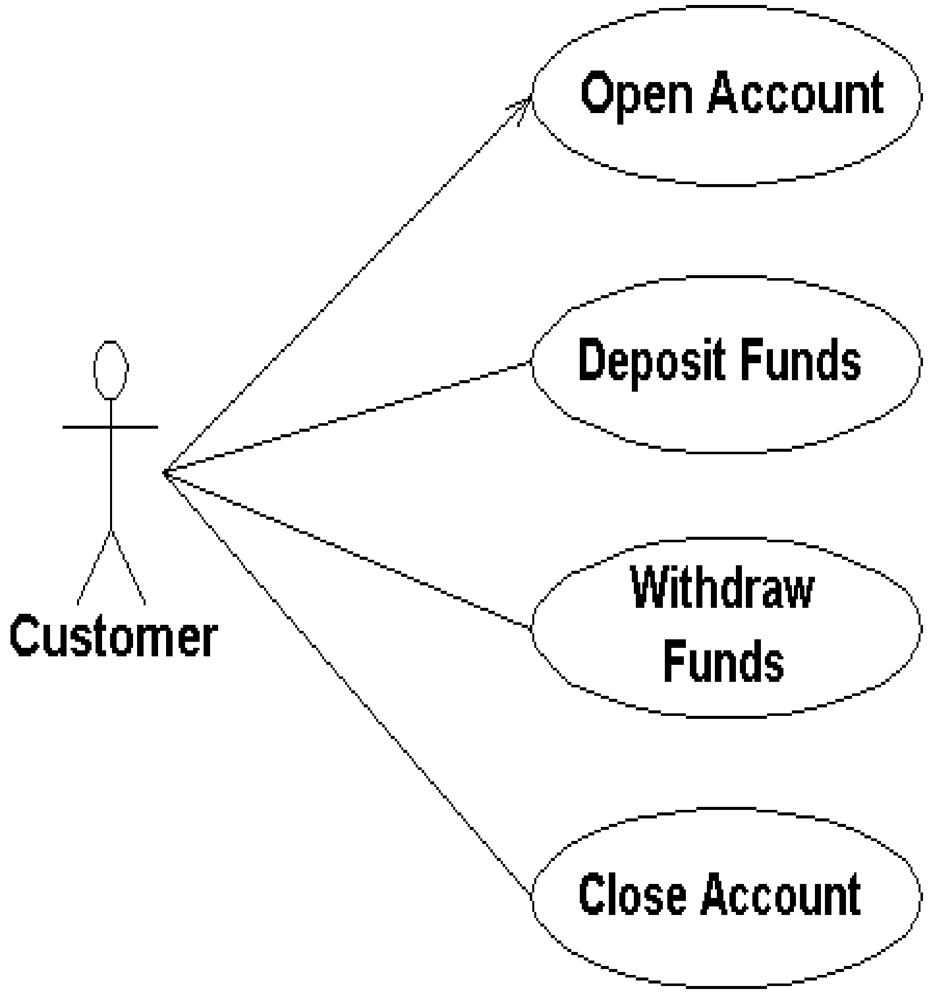
❑ An actor in a use case diagram interacts with a use case. For example, for modeling a banking application, a customer entity represents an actor in the application.

❑ Similarly, the person who provides service at the counter is also an actor. But it is up to you to consider what actors make an impact on the functionality that you want to model.

❑ If an entity does not affect a certain piece of functionality that you are modeling, it makes no sense to represent it as an actor.

❑ An actor is shown as a stick figure in a use case diagram depicted "outside" the system boundary, as shown in the below figure.

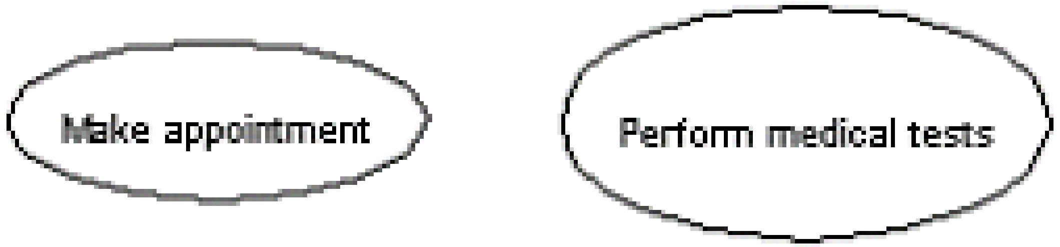
Doctor
Patient


Customer
Open Account
Deposit Funds
Withdraw Funds
Close Account

a. Use-case Diagram

❖Use case:

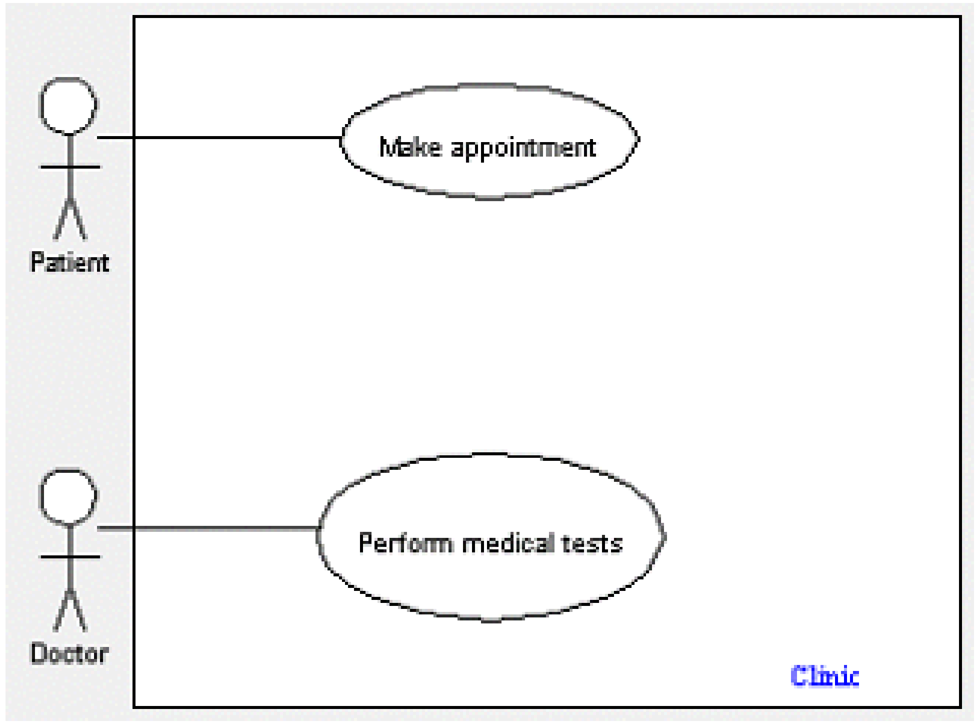
- ❑A use case in a use case diagram is a visual representation of a distinct business functionality/operations in a system.
- ❑Example In a bank system, the operations are: withdraw, deposit, balance enquiry etc.
- ❑In online shopping system, the operations are: order, payment, view product etc.
- ❑A use case is shown as an ellipse in a use case diagram.



- ❑The above figure shows two uses cases: "Make appointment" and "Perform medical tests" in the use case diagram of a clinic system.

❖System boundary:

- ❑A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases also need to have definitive limits defined. A system boundary of a use case diagram defines the limits of the system.
- ❑The system boundary is shown as a rectangle spanning all the use cases in the system.



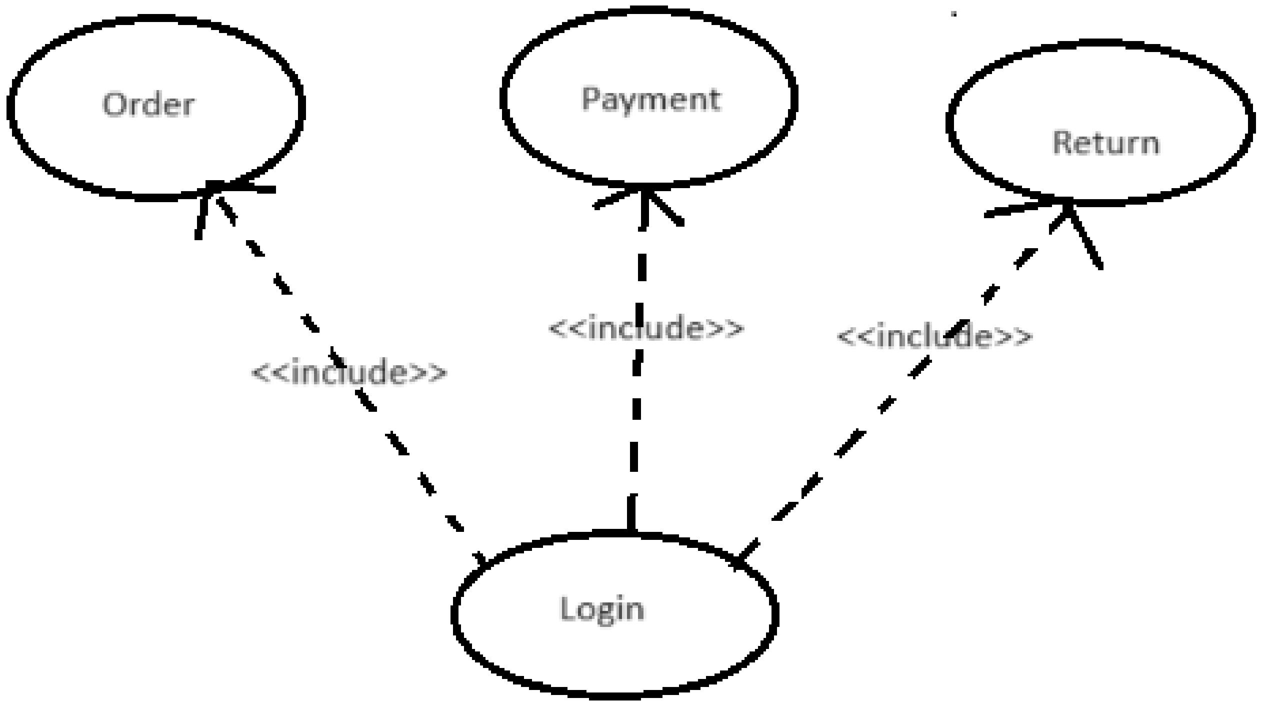
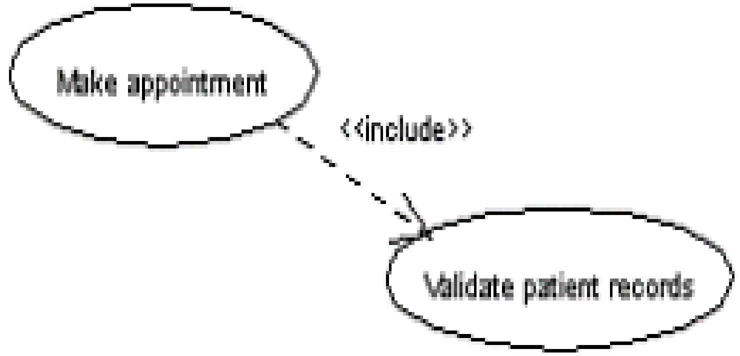
❖Relationships in Use Cases

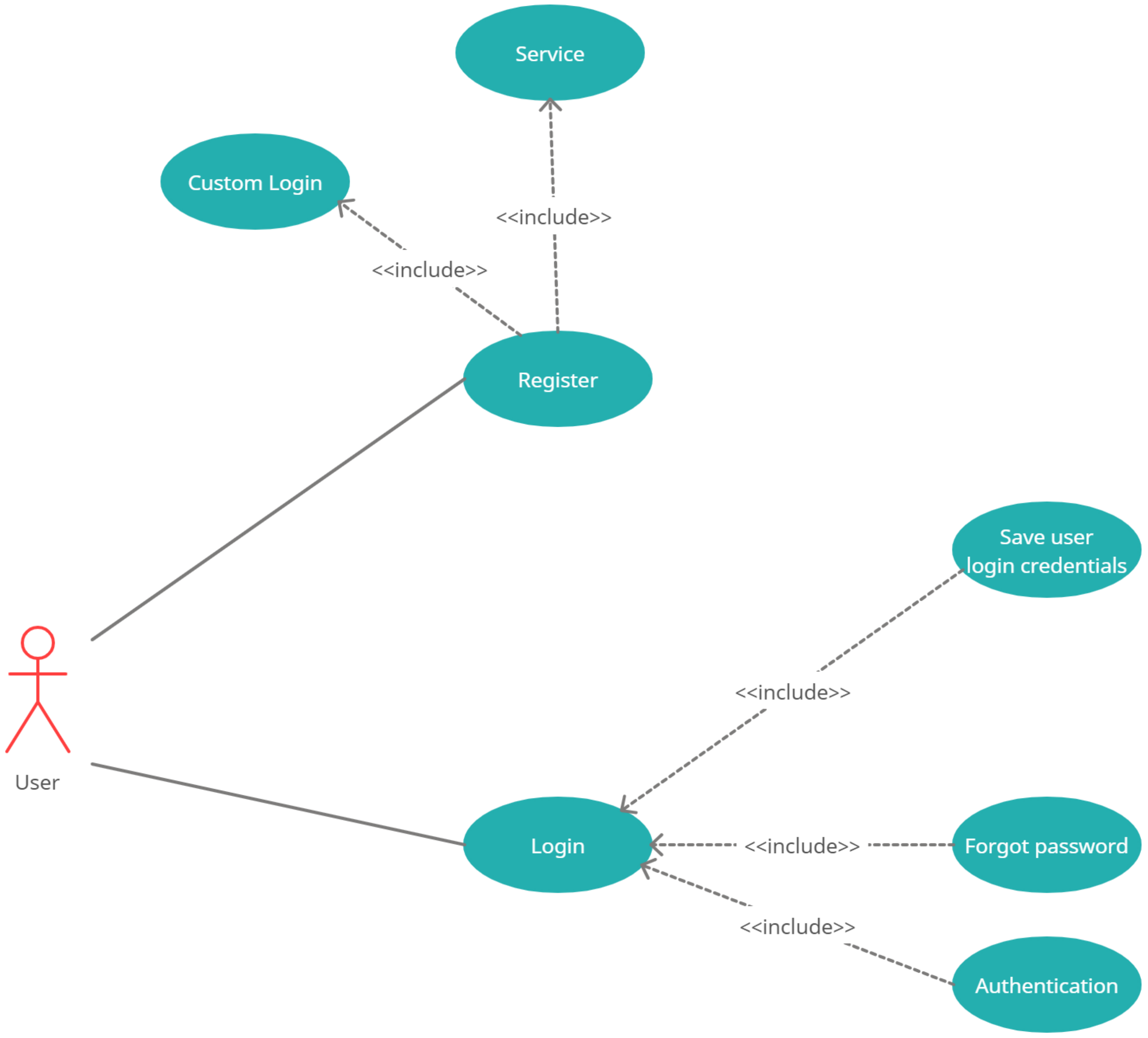
❑Use cases share different kinds of relationships. A relationship between two use cases is basically a dependency between the two use cases.

1.Include and Extend Relationships:

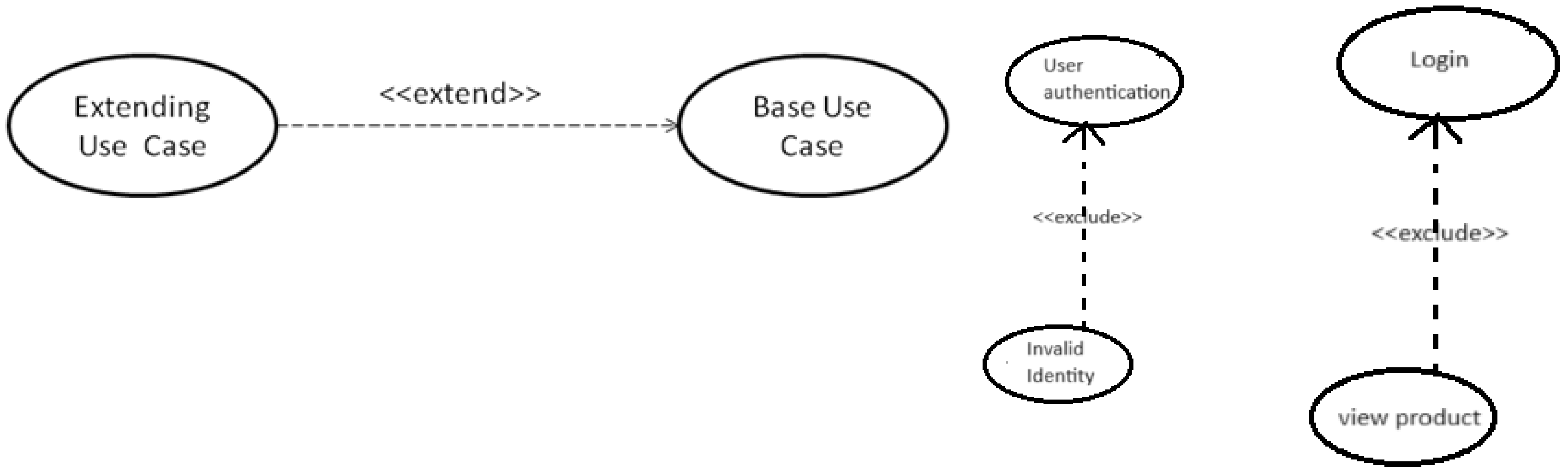
❖Use case diagrams may also include include and extend relationships between use cases. An include relationship indicates that one use case includes the behavior of another use case(mandatory/ compulsory), while an extend relationship indicates that one use case can optionally extend another use case under certain conditions. In an *extend* relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case.

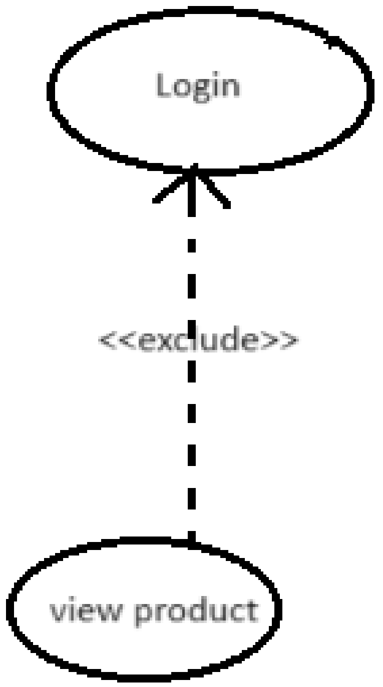
❖An include relationship is depicted with a directed arrow having a dotted shaft. The tip of the arrowhead points to the child use case and the parent use case is connected at the base of the arrow. A key here is that the included use case cannot stand alone, i.e., one would not validate the patient record without making an appointment. The stereotype "<<include>>" identifies the relationship as an include relationship.

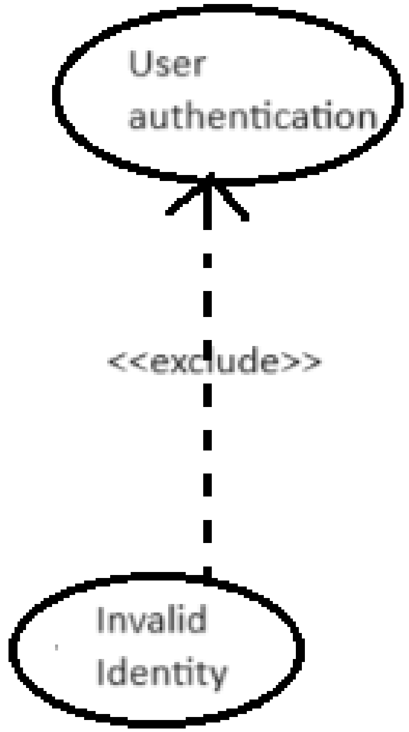

Order
Payment
Return
<<include>>
<<include>>
<<include>>
Login

Make appointment
<<include>>
Validate patient records


Service
Custom Login
<<include>>
<<include>>
Register
User
Save user login credentials
<<include>>
Login
<<include>>
Forgot password
<<include>>
Authentication

- An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow. The stereotype "<<extend>>" identifies the relationship as an extend relationship, as shown:




Login
<<exclude>>
view product


User authentication
<<exclude>>
Invalid Identity

❖ Generalization:

generalization is a relationship between two classes where one class (the child or subclass) inherits attributes, operations, and behaviors from another class (the parent or superclass). This relationship is often represented using inheritance in programming languages.

- **Note : Remaining UML diagrams we did in the class(on white board)**