UNIFIED MODELING LANGUAGE LAB (22CSL38A)

For III SEMESTER

Department of Computer Science & Engineering

By

PROF. NARENDRA N, PROF. SWATHI N

.

UML

What is UML?

"The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems".— OMG UML Specification

"UML is a graphical notation for modeling various aspects of software systems." — whm

Why use UML?

Two questions, really:

1) Why use a graphical notation of any sort?
Facilitates construction of models that in turn can be used to:
Reason about system behavior.
Present proposed designs to others.
Document key elements of design for future understanding.

2) Which graphical notation should be used?
UML has become the de-facto standard for modeling object oriented systems. UML is extensible and method-independent.
UML is not perfect, but it's good enough.

The Origins of UML

Object-oriented programming reached the mainstream of programming in the late 1980's and early 1990's. The rise in popularity of object-oriented programming was accompanied by a profusion of object-oriented analysis and design methods, each with its own graphical notation.

Three OOA/D gurus, and their methods, rose to prominence Grady Booch — The Booch Method, James Rumbaugh, et al. — Object Modeling Technique, Ivar Jacsobson — Objectory In 1994, Booch and Rumbaugh, then both at Rational, started working on a unification of their methods. A first draft of their Unified Method was released in October 1995. In 1996, (+/-) Jacobson joined Booch and Rumbaugh at Rational; the name UML was coined. In 1997 the Object Management Group (OMG) accepted UML as an open and industry standard visual modeling language for object oriented systems. Current version of UML is 2.0.

UML Diagram Types

There are several types of UML diagrams:
Use-case Diagram
Shows actors, use-cases, and the relationships between them.
Class Diagram
Shows relationships between classes and pertinent information about classes themselves.
Object Diagram
Shows a configuration of objects at an instant in time.
Interaction Diagrams

Show an interaction between a group of collaborating objects.
Two types: Collaboration diagram and sequence diagram

Package Diagram

Shows system structure at the library/package level.

State Diagram

Describes behavior of instances of a class in terms of states, stimuli, and transitions.

Activity Diagram

Very similar to a flowchart—shows actions and decision points, but with the ability to accommodate concurrency.

Deployment Diagram

Shows configuration of hardware and software in a distributed system.

UML Modeling Types

It is very important to distinguish between the UML models. Different diagrams are used fordifferent type of UML modeling. There are three important type of UML modeling:

Structural modeling:

Structural modeling captures the static features of a system. They consist of the followings:

- Classes diagrams
- Objects diagrams
- Deployment diagrams
- Package diagrams
- Component diagrams

Structural model represents the framework for the system and this framework is the place where all other components exist. So the class diagram, component diagram and deployment diagrams are the part of structural modeling. They all represent the elements and the mechanism to assemble them.

But the structural model never describes the dynamic behavior of the system. Class diagram is the most widely used structural diagram.

Behavioral Modeling

Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modeling shows the dynamic nature of the system. They consist of the following:

- Activity diagrams
- Interaction diagrams
- Use case diagrams

All the above show the dynamic sequence of flow in a system.

Architectural Modeling

Architectural model represents the overall framework of the system. It contains both structural and behavioral elements of the system. Architectural model can be defined as the blue print of the entire system. Package diagram comes under architectural modeling.

UML Basic Notations

UML is popular for its diagrammatic notations. We all know that UML is for visualizing, specifying, constructing and documenting the components of software and non software systems. Here the Visualization is the most important part which needs to be understood and remembered by heart.

UML notations are the most important elements in modelling. Efficient and appropriate use of notations is very important for making a complete and meaningful model. The model is useless unless its purpose is depicted properly.

So learning notations should be emphasized from the very beginning. Different notations are available for things and relationships. And the UML diagrams are made using the notations of things and relationships. Extensibility is another important feature which makes UML more powerful and flexible.

Structural Things

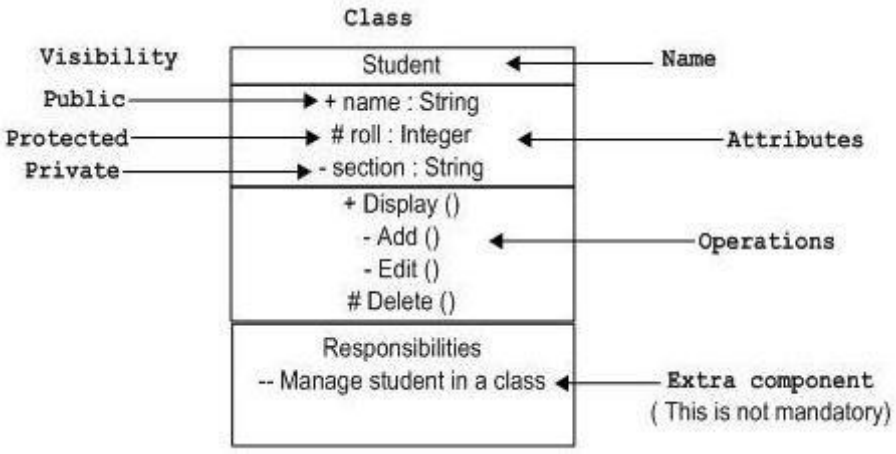
Graphical notations used in structural things are the most widely used in UML. These are consideredas the nouns of UML models. Following are the list of structural things.

- Classes
- Interface
- Collaboration
- Use case
- Active classes
- Components
- Nodes

Class Notation:

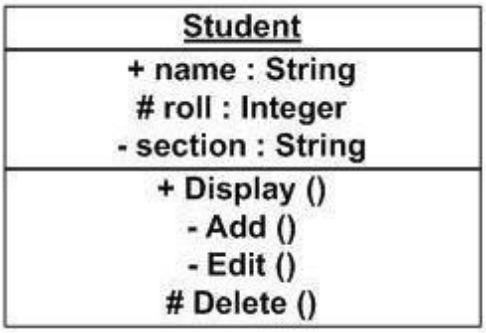
UML class is represented by the diagram shown below. The diagram is divided into four parts.

- The top section is used to name the class.
- The second one is used to show the attributes of the class.
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.



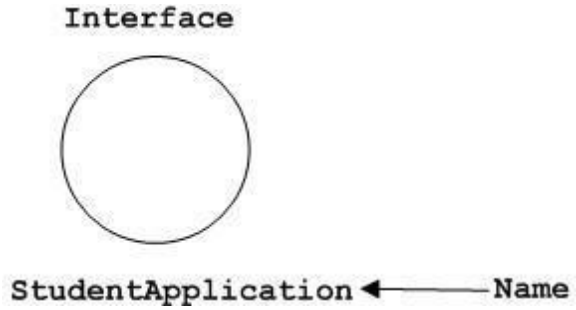
Classes are used to represent objects. Objects can be anything having properties and responsibility.

Object Notation:
The object is represented in the same way as the class. The only difference is the name which is underlined as shown below:



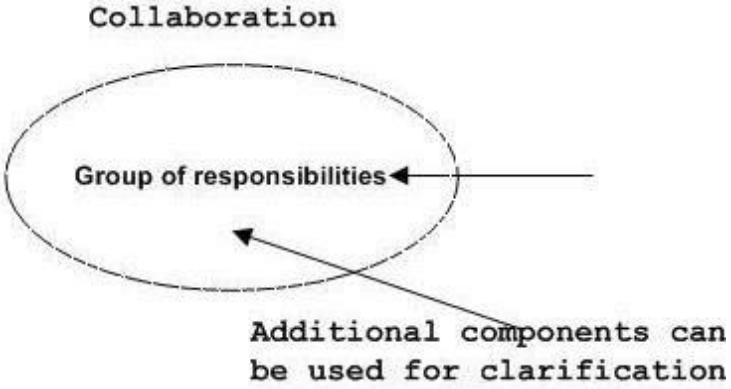
As object is the actual implementation of a class which is known as the instance of a class. So it has the same usage as the class.

Interface Notation:
Interface is represented by a circle as shown below. It has a name which is generally written below the circle.



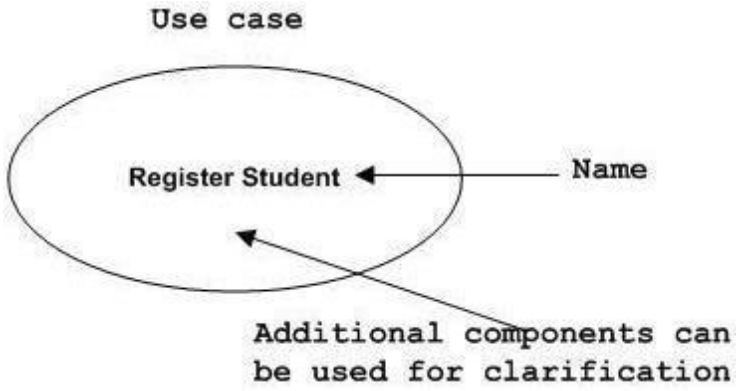
Interface is used to describe functionality without implementation. Interface is the just like a template where you define different functions not the implementation. When a class implements the interface it also implements the functionality as per the requirement.

Collaboration Notation:
Collaboration is represented by a dotted eclipse as shown below. It has a name written inside the eclipse.



Collaboration represents responsibilities. Generally responsibilities are in a group.

Use case Notation:
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.

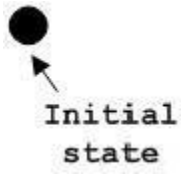


Use case is used to capture high level functionalities of a system.

Actor Notation:
An actor can be defined as some internal or external entity that interacts with the system.

Actor is used in a use case diagram to describe the internal or external entities.

Initial State Notation:
Initial state is defined show the start of a process. This notation is used in almost all diagrams.

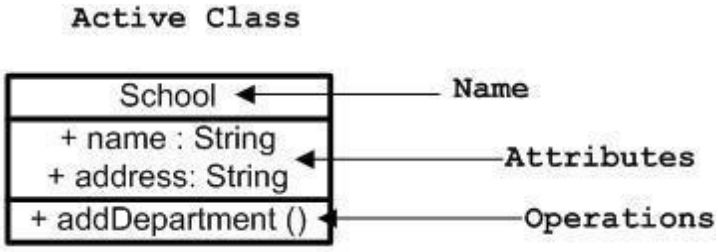


The usage of Initial State Notation is to show the starting point of a process.

Final State Notation:
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.

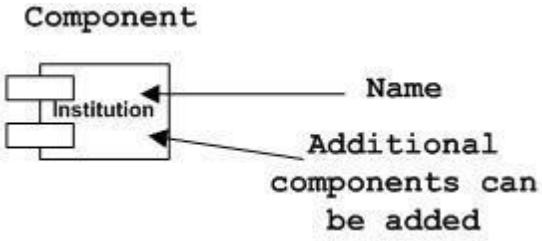
The usage of Final State Notation is to show the termination point of a process.

Active class Notation:
Active class looks similar to a class with a solid border. Active class is generally used to describe concurrent behavior of a system.



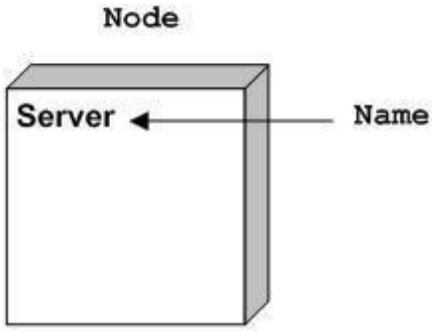
Active class is used to represent concurrency in a system.

Component Notation:
A component in UML is shown as below with a name inside. Additional elements can be added wherever required.



Component is used to represent any part of a system for which UML diagrams are made.

Node Notation:
A node in UML is represented by a square box as shown below with a name. A node represents a physical component of the system.



Node is used to represent physical part of a system like server, network etc.

Behavioural Things:

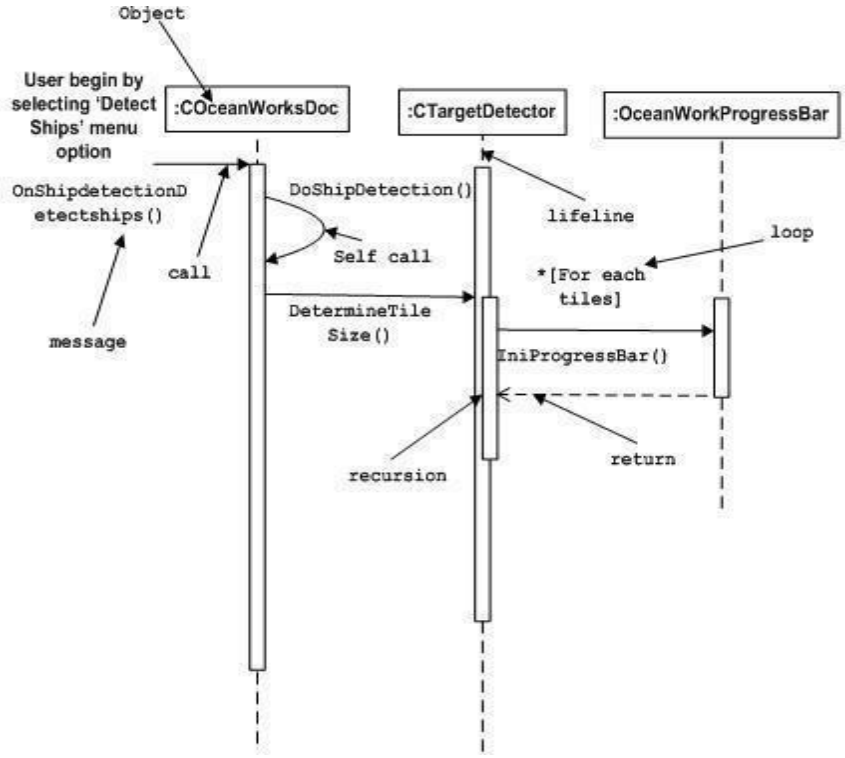
Dynamic parts are one of the most important elements in UML. UML has a set of powerful features to represent the dynamic part of software and non software systems. These features include interactions and state machines.

Interactions can be of two types:

- Sequential (Represented by sequence diagram)
- Collaborative (Represented by collaboration diagram)

Interaction Notation:

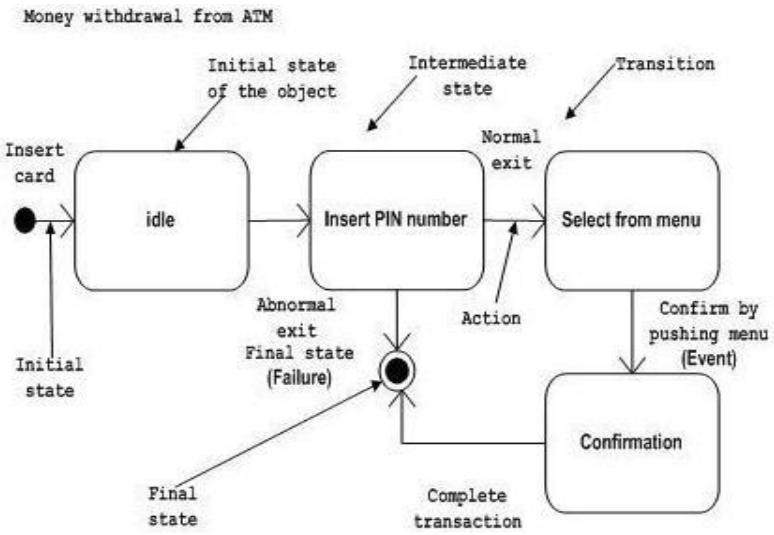
Interaction is basically message exchange between two UML components. The following diagram represents different notations used in an interaction.



Interaction is used to represent communication among the components of a system.

State machine Notation:

State machine describes the different states of a component in its life cycle. The notations are described in the following diagram.

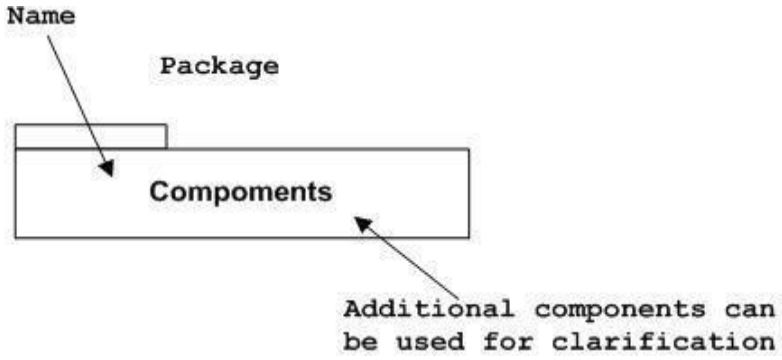


State machine is used to describe different states of a system component. The state can be active, idle or any other depending upon the situation.

Grouping Things:

Organizing the UML models are one of the most important aspects of the design. In UML there is only one element available for grouping and that is package.

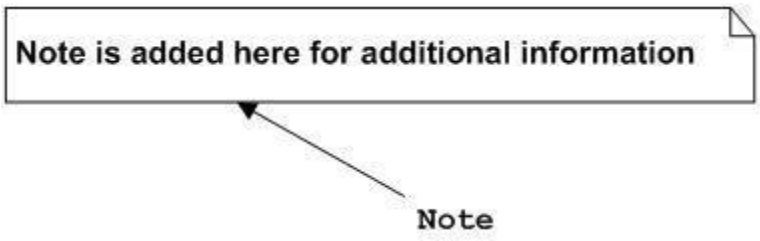
Package Notation:
Package notation is shown below and this is used to wrap the components of a system.



Annotational Things:

In any diagram explanation of different elements and their functionalities are very important. So UML has notes notation to support this requirement.

Note Notation:
This notation is shown below and they are used to provide necessary information of a system.

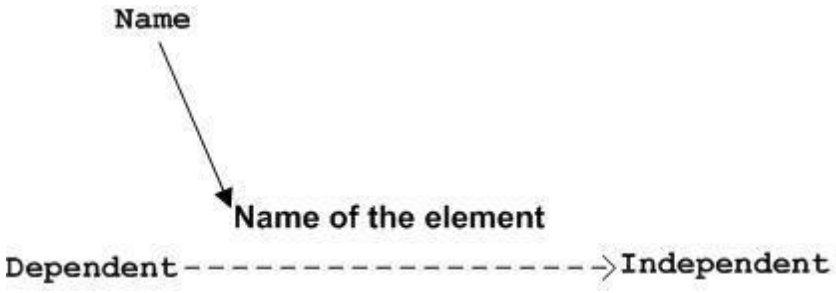


Relationships

A model is not complete unless the relationships between elements are described properly. The Relationship gives a proper meaning to an UML model. Following are the different types of relationships available in UML.

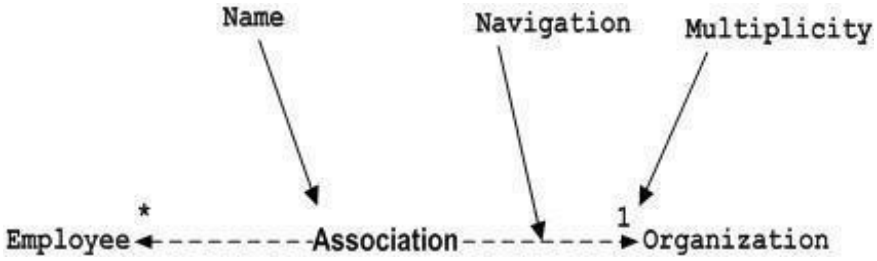
- Dependency
- Association
- Generalization
- Extensibility

Dependency Notation:
Dependency is an important aspect in UML elements. It describes the dependent elements and the direction of dependency. Dependency is represented by a dotted arrow as shown below. The arrow head represents the independent element and the other end the dependent element.



Dependency is used to represent dependency between two elements of a system.

Association Notation:
Association describes how the elements in an UML diagram are associated. In simple word it describes how many elements are taking part in an interaction.
Association is represented by a dotted line with (without) arrows on both sides. The two ends represent two associated elements as shown below. The multiplicity is also mentioned at the ends(1, * etc) to show how many objects are associated.



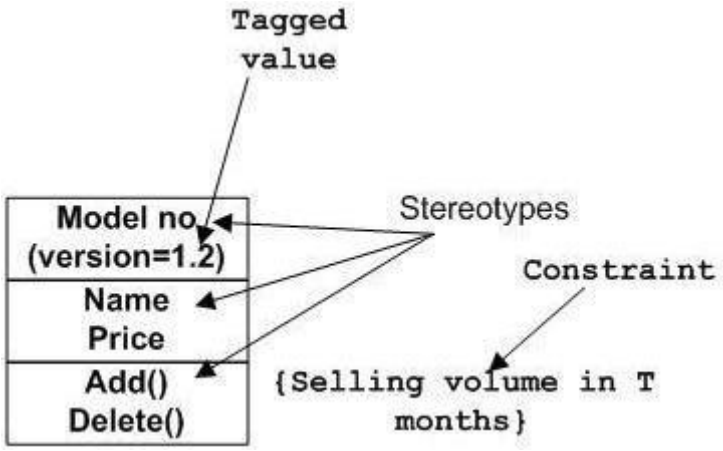
Association is used to represent the relationship between two elements of a system.

Generalization Notation:
Generalization describes the inheritance relationship of the object oriented world. It is parent and child relationship.
Generalization is represented by an arrow with hollow arrow head as shown below. One endrepresents the parent element and the other end child element.

Generalization is used to describe parent-child relationship of two elements of a system.

Extensibility Notation:
All the languages (programming or modelling) have some mechanism to extend its capabilities like syntax, semantics etc. UML is also having the following mechanisms to provide extensibility features.

- Stereotypes (Represents new elements)
- Tagged values (Represents new attributes)
- Constraints (Represents the boundaries)



Extensibility notations are used to enhance the power of the language. It is basically additional elements used to represent some extra behaviour of the system. These extra behaviours are not covered by the standard available notations.

UML Class Diagram

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

Purpose:

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

How to draw Class Diagram?

Class diagrams are the most popular UML diagrams used for construction of software applications. So it is very important to learn the drawing procedure of class diagram.

Class diagrams have lot of properties to consider while drawing but here the diagram will be considered from a top-level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system.

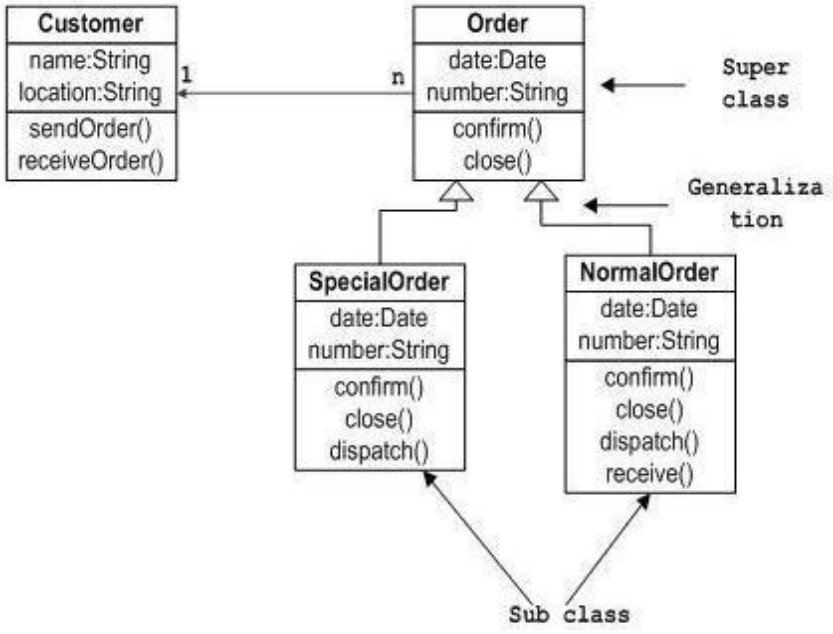
The following points should be remembered while drawing a class diagram:

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified.
- For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. Because at the endof the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

Now the following diagram is an example of an Order System of an application. So it describes a particular aspect of the entire application.

- First of all, Order and Customer are identified as the two elements of the system and they have a one to many relationships because a customer can have multiple orders.
- We would keep Order class is an abstract class and it has two concrete classes (inheritance relationship) Special-order and Normal Order.
- The two inherited classes have all the properties as the Order class. In addition they have additional functions like dispatch () and receive ().

So the following class diagram has been drawn considering all the points mentioned above:



UML Object Diagram

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

Purpose:

The purpose of a diagram should be understood clearly to implement it practically. The purposes of object diagrams are similar to class diagrams.

The difference is that a class diagram represents an abstract model consists of classes and their relationships. But an object diagram represents an instance at a particular moment which is concrete in nature.

It means the object diagram is more close to the actual system behaviour. The purpose is to capture the static view of a system at a particular moment.
So the purpose of the object diagram can be summarized as:

- Forward and reverse engineering.
- Object relationships of a system .
- Static view of an interaction.
- Understand object behavior and their relationship from practical perspective.

How to draw Object Diagram?

We have already discussed that an object diagram is an instance of a class diagram. It implies that an object diagram consists of instances of things used in a class diagram.

So both diagrams are made of same basic elements but in different form. In class diagram elements are in abstract form to represent the blue print and in object diagram the elements are in concrete form to represent the real world object.

To capture a particular system, numbers of class diagrams are limited. But if we consider object diagrams then we can have unlimited number of instances which are unique in nature. So only those instances are considered which are having impact on the system.

From the above discussion it is clear that a single object diagram cannot capture all the necessary instances or rather cannot specify all objects of a system. So the solution is:

- First, analyze the system and decide which instances are having important data and association.
- Second, consider only those instances which will cover the functionality.
- Third, make some optimization as the numbers of instances are unlimited.

Before drawing an object diagrams the following things should be remembered and understood clearly:

- Object diagrams are consist of objects.
- The link in object diagram is used to connect objects.
- Objects and links are the two elements used to construct an object diagram.

Now after this the following things are to be decided before starting the construction of the diagram:

- The object diagram should have a meaningful name to indicate its purpose.
- The most important elements are to be identified.
- The association among objects should be clarified.
- 0Values of different elements need to be captured to include in the object diagram.
- Add proper notes at points where more clarity is required.

The following diagram is an example of an object diagram. It represents the Order management system which we have discussed in Class Diagram. The following diagram is an instance of the system at a particular time of purchase. It has the following objects

- Customer
- Order
- SpecialOrder
- NormalOrder

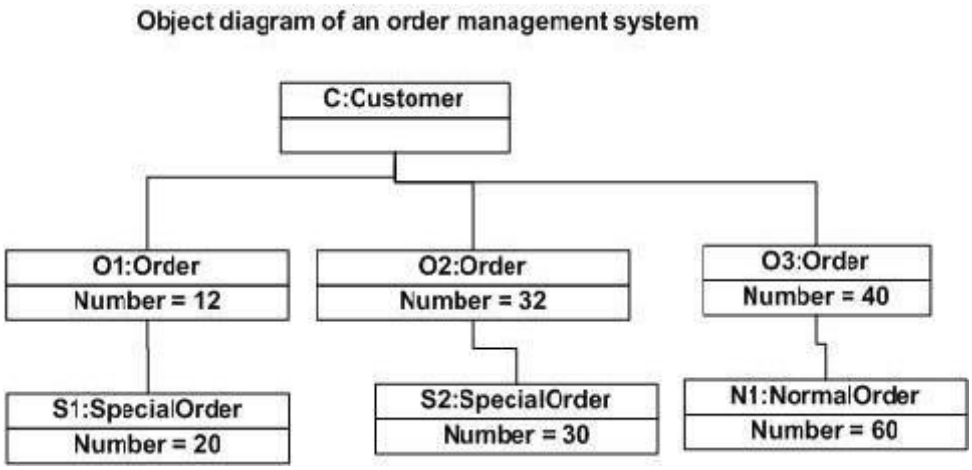
Now the customer object (C) is associated with three order objects (O1, O2 and O3). These order objects are associated with special order and normal order objects (S1, S2 and N1). The

customer is having the following three orders with different numbers (12, 32 and 40) for the particular time considered.

Now the customer can increase number of orders in future and in that scenario the object diagram will reflect that. If order, special order and normal order objects are observed then we you will find that they are having some values.

For orders the values are 12, 32, and 40 which implies that the objects are having these values for the particular moment (here the particular time when the purchase is made is considered as the moment) when the instance is captured.

The same is for special order and normal order objects which are having number of orders as 20, 30 and 60. If a different time of purchase is considered then these values will change accordingly. So the following object diagram has been drawn considering all the points mentioned above:



UML Component Diagram

Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model physical aspects of a system.

Now the question is what are these physical aspects? Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node.

So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose:

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

So the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

How to draw Component Diagram?

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries etc.

So the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details.

Initially the system is designed using different UML diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.

This diagram is very important because without it the application cannot be implemented efficiently. A well prepared component diagram is also important for other aspects like application performance, maintenance etc.

So before drawing a component diagram the following artifacts are to be identified clearly:

- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.

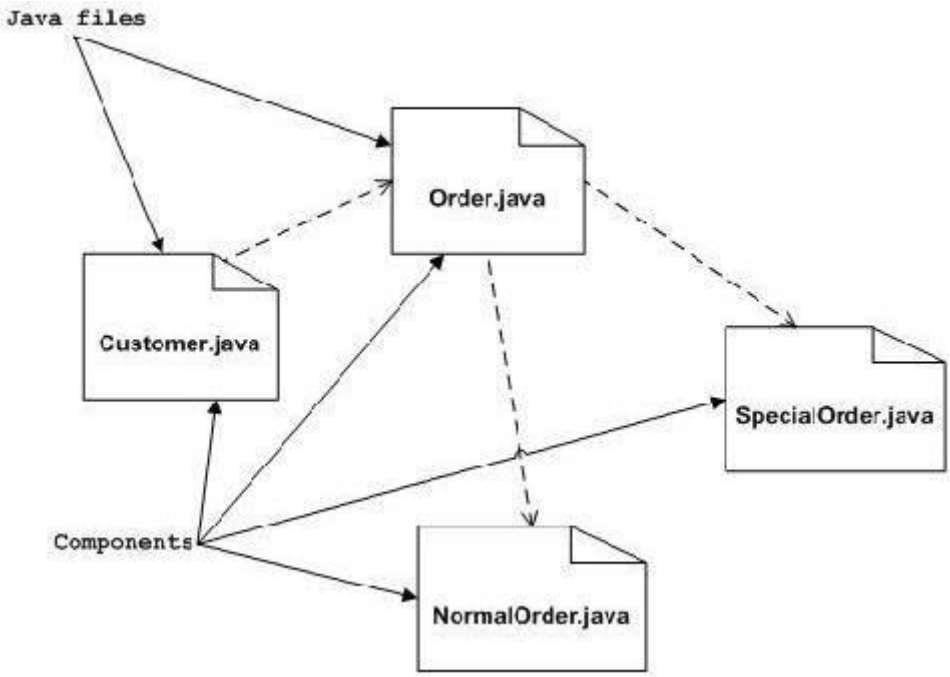
Now after identifying the artifacts the following points needs to be followed:

- Use a meaningful name to identify the component for which the diagram is to be drawn.
- Prepare a mental layout before producing using tools.
- Use notes for clarifying important points.

The following is a component diagram for order management system. Here the artifacts are files. So the diagram shows the files in the application and their relationships. In actual the component diagram also contains dlls, libraries, folders etc.

In the following diagram four files are identified and their relationships are produced. Component diagram cannot be matched directly with other UML diagrams discussed so far. Because it is drawn for completely different purpose.

So the following component diagram has been drawn considering all the points mentioned above:



UML Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Purpose:

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

How to draw Deployment Diagram?

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardwares used to deploy the application.

Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important because it controls the following parameters

- Performance
- Scalability
- Maintainability
- Portability

So before drawing a deployment diagram the following artifacts should be identified:

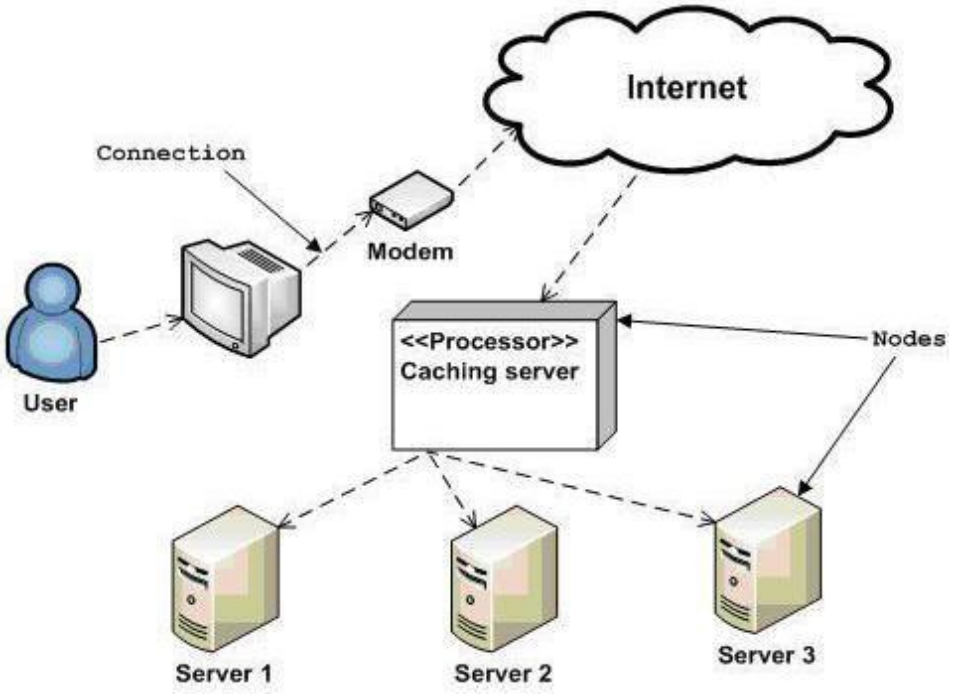
- Nodes
- Relationships among nodes

The following deployment diagram is a sample to give an idea of the deployment view of order management system. Here we have shown nodes as:

- Monitor
- Modem
- Caching server
- Server

The application is assumed to be a web based application which is deployed in a clustered environment using server 1, server 2 and server 3. The user is connecting to the application using internet. The control is flowing from the caching server to the clustered environment.

So the following deployment diagram has been drawn considering all the points mentioned above:



UML Use Case Diagram

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running /operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

So to model the entire system numbers of use case diagrams are used.

Purpose:

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view. So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

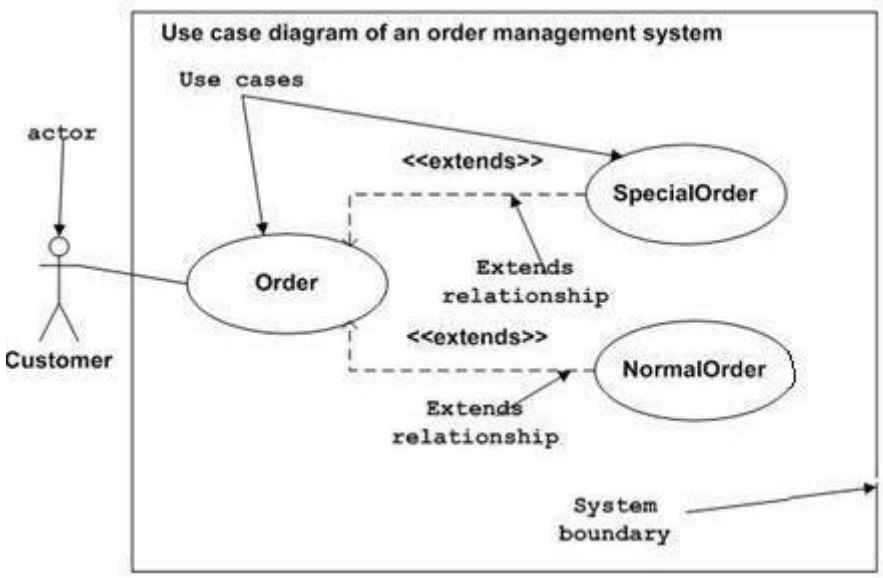
- Functionalities to be represented as an use case
- Actors
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note when ever required to clarify some important points.

The following is a sample use case diagram representing the order management system. Soif we look into the diagram then we will find three use cases (Order, SpecialOrder and NormalOrder) and one actor which is customer.

The SpecialOrder and NormalOrder use cases are extended from Order use case. So they have extends relationship. Another important point is to identify the system boundary which is shown in the picture. The actor Customer lies outside the system as it is an external user of the system.



UML Interaction Diagram

From the name Interaction it is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction is a part of dynamic behaviour of the system.

This interactive behaviour is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purposes of both the diagrams are similar.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

Purpose:

The purposes of interaction diagrams are to visualize the interactive behaviour of the system. Now visualizing interaction is a difficult task. So the solution is to use different types of models to capture the different aspects of the interaction.

That is why sequence and collaboration diagrams are used to capture dynamic nature but from a different angle.

So the purposes of interaction diagram can be describes as:

- To capture dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe structural organization of the objects.
- To describe interaction among objects.

How to draw Interaction Diagram?

As we have already discussed that the purpose of interaction diagrams are to capture the dynamic aspect of a system. So to capture the dynamic aspect we need to understand what a dynamic aspect is and how it is visualized. Dynamic aspect can be defined as the snap shot of the running system at a particular moment.

We have two types of interaction diagrams in UML. One is sequence diagram and the other is a collaboration diagram. The sequence diagram captures the time sequence of message flow from one object to another and the collaboration diagram describes the organization of objects in a system taking part in the message flow.

So the following things are to identified clearly before drawing the interaction diagram:

- Objects taking part in the interaction.
- Message flows among the objects.
- The sequence in which the messages are flowing.
- Object organization.

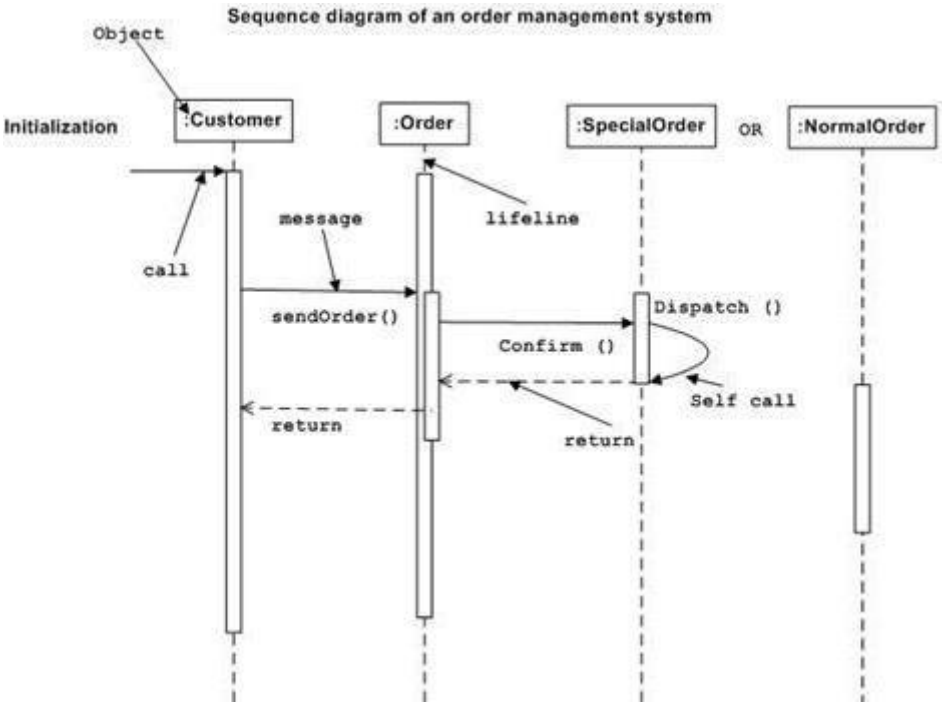
Following are two interaction diagrams modelling order management system. The first diagram is a sequence diagram and the second is a collaboration diagram.

The Sequence Diagram:

The sequence diagram is having four objects (Customer, Order, SpecialOrder and NormalOrder).

The following diagram has shown the message sequence for SpecialOrder object and the same can be used in case of NormalOrder object. Now it is important to understand the time sequence of message flows. The message flow is nothing but a method call of an object.

The first call is sendOrder () which is a method of Order object. The next call is confirm () which is a method of SpecialOrder object and the last call is Dispatch () which is a method of SpecialOrder object. So here the diagram is mainly describing the method calls from one object to another and this is also the actual scenario when the system is running.

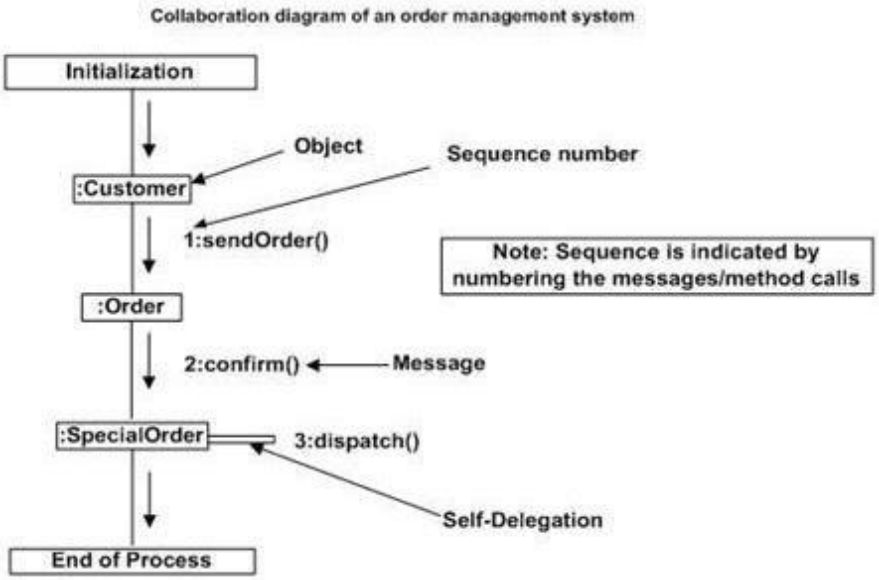


The Collaboration Diagram:

The second interaction diagram is collaboration diagram. It shows the object organization as shown below. Here in collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.

Now to choose between these two diagrams the main emphasis is given on the type of requirement. If the time sequence is important then sequence diagram is used and if organization is required then collaboration diagram is used.



UML Statechart Diagram

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A Statechart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in next chapter, is a special kind of a Statechart diagram. As Statechart diagram defines states it is used to model lifetime of an object.

Purpose:

Statechart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of Statechart diagram is to model life time of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using Statechart diagrams:

- To model dynamic aspect of a system.
- To model life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model states of an object.

How to draw Statechart Diagram?

Statechart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

Statechart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a Statechart diagram we must have clarified the following points:

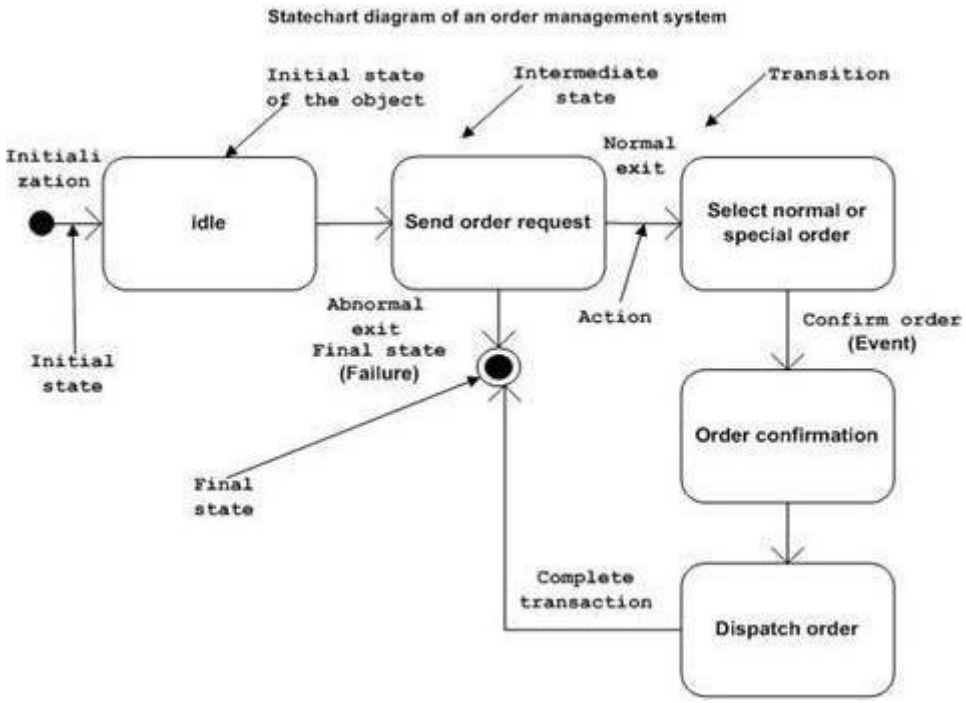
- Identify important objects to be analyzed.
- Identify the states.
- Identify the events.

The following is an example of a Statechart diagram where the state of Order object is analyzed.

The first state is an idle state from where the process starts. The next states are arrived for events like send request, confirm request, and dispatch order. These events are responsible for state changes of order object.

During the life cycle of an object (here order object) it goes through the following states and there may be some abnormal exists also. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete it is considered as the complete transaction as mentioned below.

The initial and final state of an object is also shown below:



UML Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system.

Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

Purpose:

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

So the purposes can be described as:

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

How to draw Activity Diagram?

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram are not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlane etc.

Before drawing an activity diagram we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities we need to understand how they are associated with constraints and conditions.

So before drawing an activity diagram we should identify the following elements:

- Activities
- Association
- Conditions
- Constraints

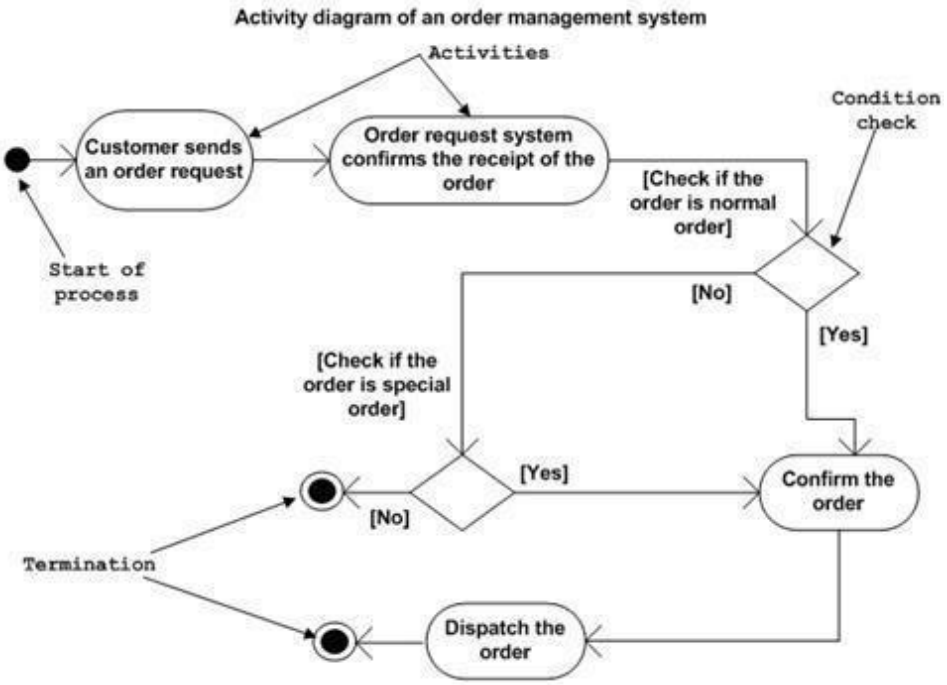
Once the above mentioned parameters are identified we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram.

The following is an example of an activity diagram for order management system. In the diagram four activities are identified which are associated with conditions. One important point should be clearly understood that an activity diagram cannot be exactly matched with the code. The activity diagram is made to understand the flow of activities and mainly used by the business users.

The following diagram is drawn with the four main activities:

- Send order by the customer
- Receipt of the order
- Confirm order
- Dispatch order

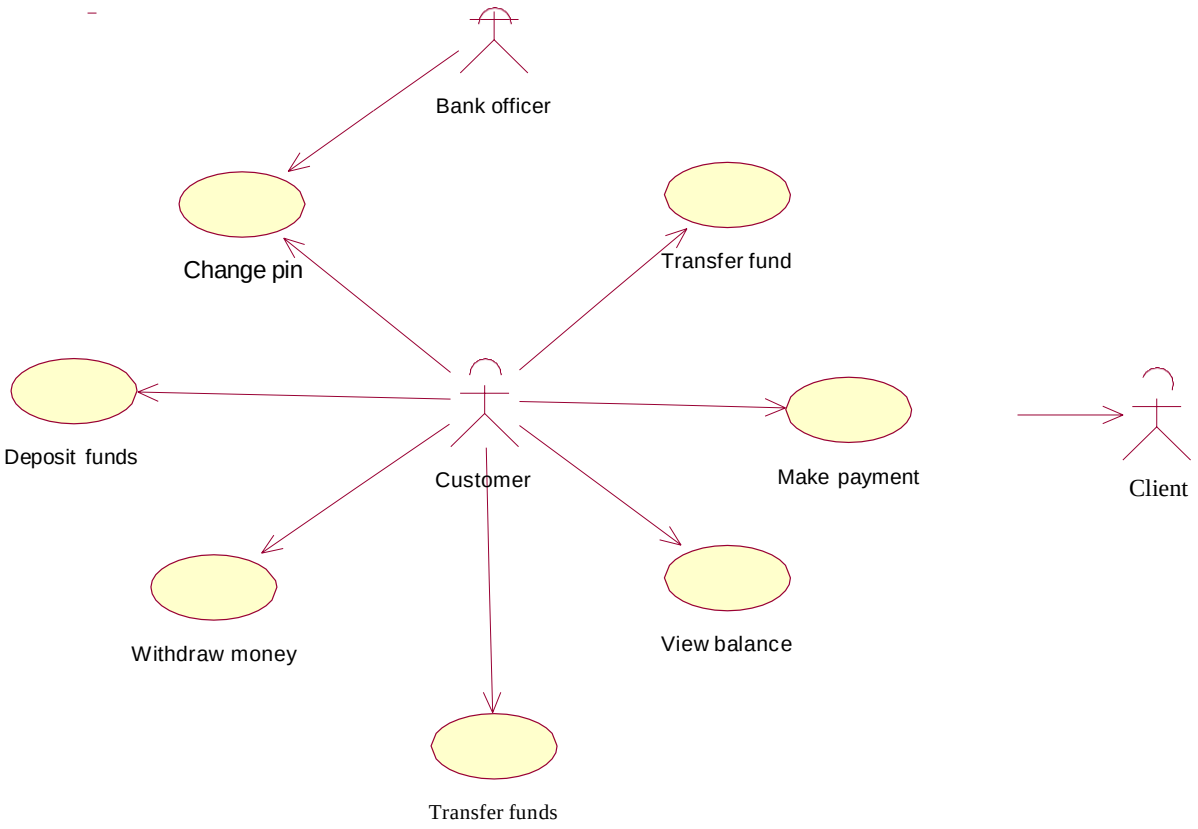
After receiving the order request condition checks are performed to check if it is normal or special order. After the type of order is identified dispatch activity is performed and that is marked as the termination of the process.


Activity diagram of an order management system
Activities
Condition check
Customer sends an order request
Order request system confirms the receipt of the order
[Check if the order is normal order]
Start of process
[No]
[Yes]
[Check if the order is special order]
[Yes]
[No]
Confirm the order
Termination
Dispatch the order

Experiments

1.Imagine you are tasked with developing a comprehensive UML diagram for an Automated Teller Machine (ATM) application. The ATM system should support basic banking transactions such as cash withdrawals, balance inquiries, and fund transfers

ATM Scenario Use Case Diagram:



Use Case Diagram for ATM System:

Actors:

- **Customer:** Interacts with the ATM to perform transactions.
- **Bank:** Manages the ATM system.

Use Cases:

1. **Withdraw Cash:**
   - **Actor:** Customer
   - **Description:** The customer inserts their bank card, enters the PIN, selects the withdrawal option, specifies the amount, and receives the cash.

2. **Deposit Cash:**
   - **Actor:** Customer
   - **Description:** The customer inserts their bank card, enters the PIN, selects the deposit option, inserts cash or checks, and confirms the transaction.

3. **Check Balance:**

- **Actor:** Customer
- **Description:** The customer inserts their bank card, enters the PIN, selects the balance inquiry option, and receives information about their account balance.
-

4. **Transfer Money:**

- **Actor:** Customer
- **Description:** The customer inserts their bank card, enters the PIN, selects the transfer option, specifies the recipient and amount, and confirms the transfer.

5. **Change PIN:**

- **Actor:** Customer
- **Description:** The customer inserts their bank card, enters the current PIN, selects the change PIN option, enters a new PIN, and confirms the change.

6. **ATM Maintenance:**

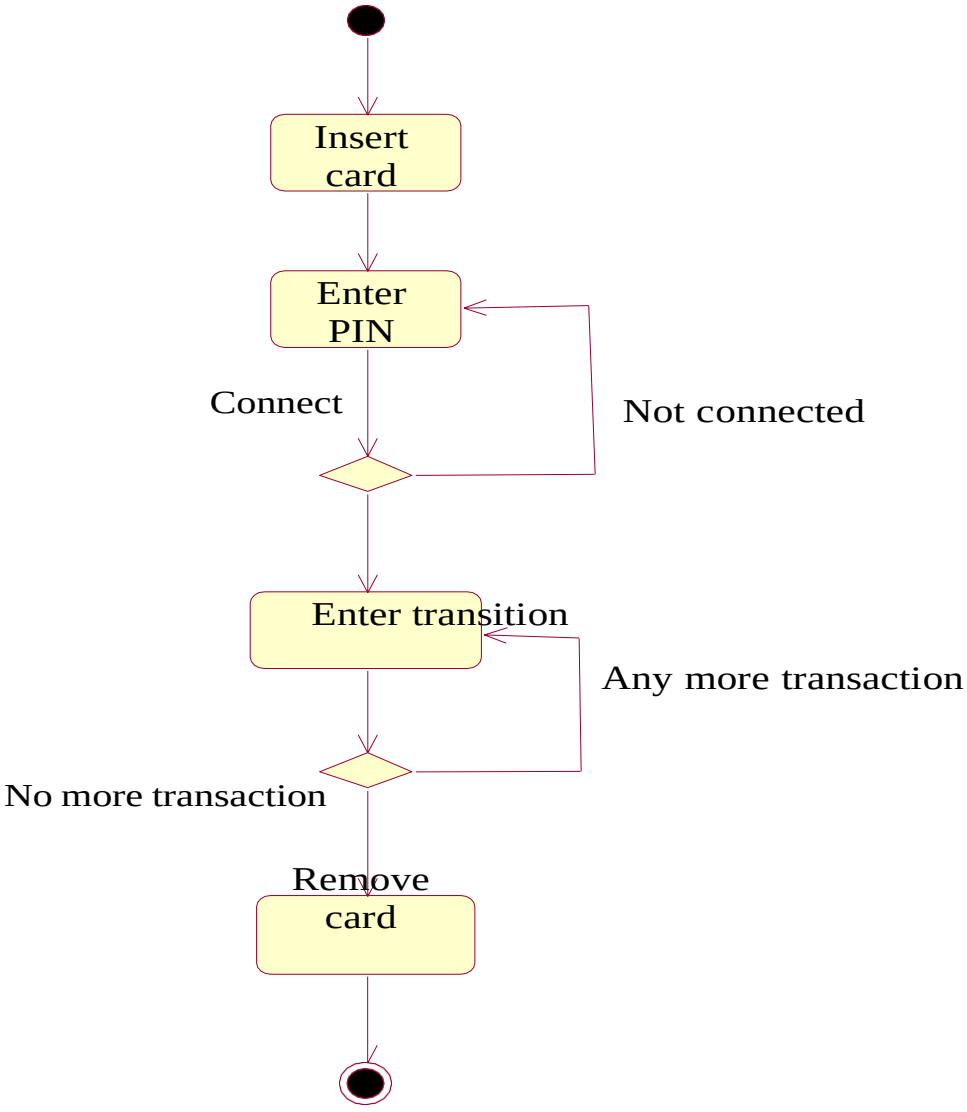
- **Actor:** Bank
- **Description:** The bank initiates maintenance activities on the ATM, such as refilling cash, updating software, or fixing hardware issues.

7. **Relationships:**

- **Includes Relationship:**
  - Withdraw Cash includes Check Balance (customer needs to check balance before withdrawing).
  - Transfer Money includes Check Balance (customer needs to check balance before transferring).
- **Extends Relationship:**
  - Deposit Cash extends Withdraw Cash (depositing is an extension of the withdrawal process).

This use case diagram provides a high-level overview of the interactions between actors and the ATM system. Note that this is a simplified representation, and additional details, such as preconditions, postconditions, and exceptions, could be added for a more comprehensive understanding.

ATM Scenario Activity Diagram:



Activity Diagram for ATM System:

An activity diagram models the workflow or activities involved in a particular process. Below is a simplified activity diagram for an ATM system:

Activities:

1. Start: Initial point of the activity diagram.

2. Customer Inserts Card: The process begins when the customer inserts their bank card into the ATM.

3. System Verifies Card: The ATM system verifies the inserted card, checking its validity.

4. Customer Enters PIN: The customer enters their Personal Identification Number (PIN).

5. System Verifies PIN: The ATM system verifies the entered PIN with the one stored in the system.

6. Menu Display: Once the card and PIN are verified, the ATM displays a menu with options such as Withdraw Cash, Deposit, Check Balance, Transfer Money, Change PIN, etc.

7. Customer Selects Transaction: The customer selects a transaction from the displayed menu.

8. Perform Transaction: Depending on the selected transaction, the system performs the necessary actions (e.g., dispensing cash, processing a deposit, checking balance, etc.).

9. Additional Transaction?

- **(Decision):** After completing the selected transaction, the system checks if the customer wants to perform another transaction.
- **If Yes:** The process loops back to the "Menu Display" step.
- **If No:** The process proceeds to the "End" activity.

10. End: Final point of the activity diagram.

Swim lanes:

- **Customer:** Activities performed by the customer, such as inserting the card, entering the PIN, and selecting a transaction.
- **ATM System:** Activities performed by the ATM system, including card verification, PIN verification, transaction processing, and menu display.
- **Arrows and Control Flow:** Arrows indicate the flow of control between activities.

Decision points are represented by diamond shapes, with outgoing arrows labelled with the conditions (e.g., Yes or No).

Notes:

- The activity diagram illustrates the sequence of actions in an ATM transaction from the customer's perspective.
- It assumes a successful verification process; error handling and exceptions can be added for a more detailed diagram.

This activity diagram provides a visual representation of the steps involved in a typical ATM transaction, making it easier to understand the flow of activities in the system.

2. Describe the UML representation of interactions in a Library Management System's book borrowing process, emphasizing actor roles, event flow, decision points, and ensuring scalability for future system enhancements.

A use case diagram for a Library Management System (LMS) in UML (Unified Modelling Language) helps to depict the various interactions between users and the system. Use case diagrams illustrate the functionalities of a system and the relationships between different actors and use cases. In the context of an LMS, common actors might include Librarian, Member, and Guest. Here's a brief description of the key elements in a use case diagram for a Library Management System:

Use Case Diagram:

1.Actors:

- **Librarian:** The Librarian is a primary actor who manages the overall functioning of the library system. They have access to administrative functionalities such as adding or removing books, managing memberships, and handling fines.
- **Member:** The Member is a registered user of the library who can borrow and return books. They have access to functionalities like searching for books, checking their account status, and placing holds.
- **Guest:** A Guest is an unregistered user who can browse the available books but has limited functionalities compared to a Member.

2.Use Cases:

- **Search for Books:** Users (Members and Guests) can search for books based on various criteria such as title, author, or genre.
- **Borrow Book:** Members can borrow books from the library by selecting available copies of a book.
- **Return Book:** Members can return borrowed books to the library.
- **Membership:** Librarians can manage user memberships, including registering new Manage members, updating member information, and deactivating memberships.
- **Add/Remove Books:** Librarians can add new books to the library inventory or remove books that are no longer available.
- **Check Account Status:** Members can check their account status, including borrowed books, due dates, and any fines incurred.
- **Place Hold:** Members can place a hold on a book that is currently checked out by another member.

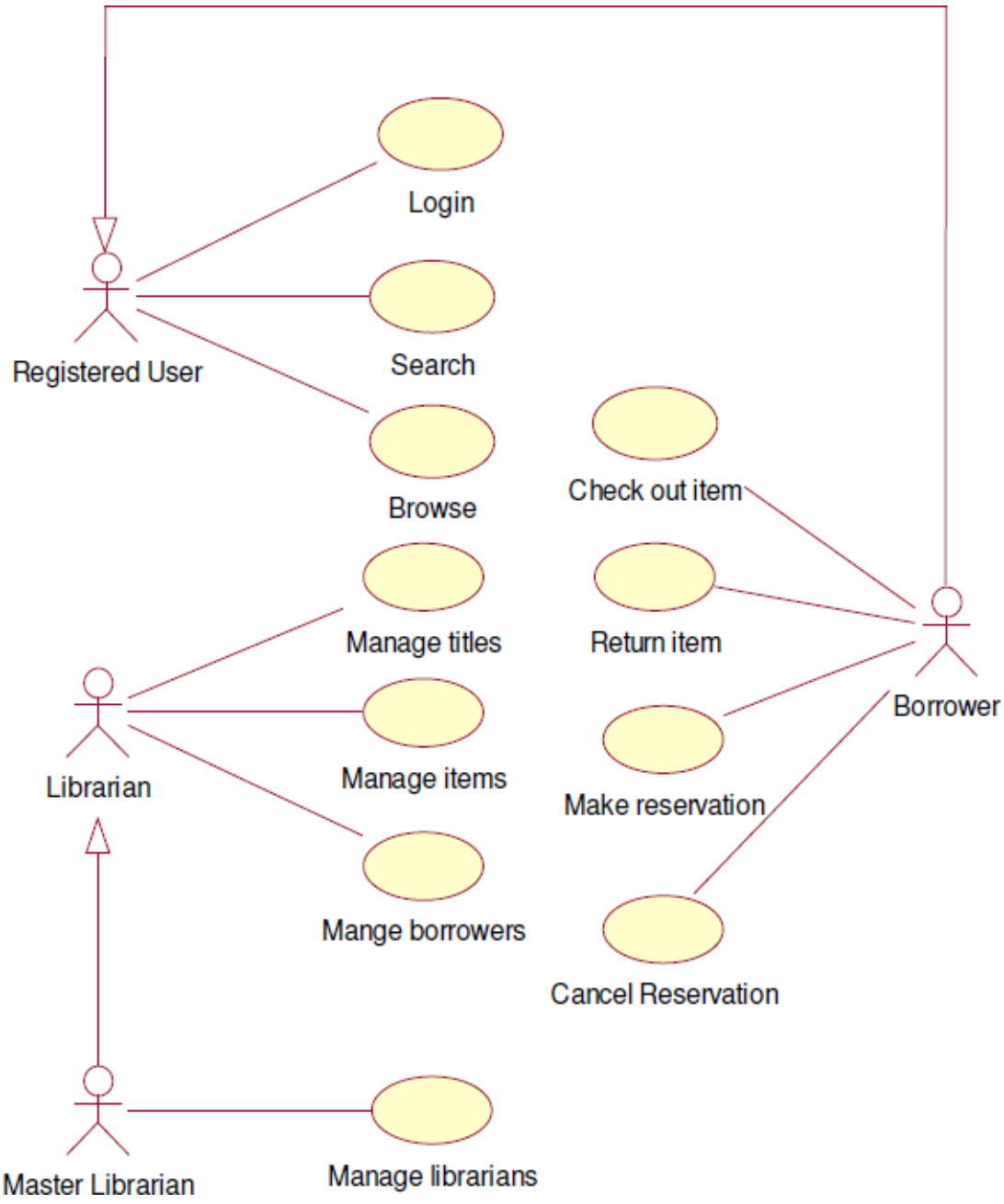
3.Associations:

- **Association between Librarian and Manage Membership:** Indicates that the Librarian is associated with the "Manage Membership" use case.
- **Association between Member and Borrow Book/Return Book:** Indicates that Members can interact with the system to borrow and return books.
- **Association between Guest and Search for Books:** Indicates that Guests can search for books in the library.

4.**Include and Extend Relationships:**

- **Include relationship between Borrow Book and Check Account Status:** Indicates that when a member borrows a book, their account status is automatically checked to ensure eligibility.
- **Extend relationship between Place Hold and Borrow Book**: Indicates that placing a hold is an optional extension of the borrowing process.

This use case diagram provides a high-level overview of the interactions within the Library Management System and helps in understanding the system's functionality from the perspective of different actors.



Activity Diagram:

An activity diagram for a Library Management System (LMS) in UML (Unified Modelling Language) helps to depict the flow of activities within the system as users interact with it. Activity diagrams are especially useful for modelling the workflow and sequential steps involved in various processes. Below is a description of an activity diagram for a Library Management System:

1. **Activity Diagram for Library Management System:**

• The diagram begins with a start node, representing the initiation of the Library Management System.

2. **User Login:**

The first activity is the "User Login" process. This involves users (Librarian, Member, or Guest) providing their credentials to access the system.

3. **Branch Based on User Type:**

After successful login, the system determines the user type (Librarian, Member, or Guest).

4. **Librarian Activities:**

If the user is identified as a Librarian, the following activities occur:

- **Manage Membership:** The Librarian can manage user memberships, including adding new members, updating member information, and deactivating memberships.
- **Add/Remove Books:** The Librarian can add new books to the library inventory or remove books that are no longer available.

5. **Member Activities:**

If the user is identified as a Member, the following activities occur:

- **Search for Books:** Members can search for books based on criteria such as title, author, or genre.
- **Borrow Book:** Members can select available copies of a book and borrow them.
- **Return Book:** Members can return borrowed books to the library.
- **Check Account Status:** Members can check their account status, including borrowed books, due dates, and any fines incurred.

- **Place Hold:** Members can place a hold on a book that is currently checked out by another member.

6. **Guest Activities:**

If the user is identified as a Guest, the following activities occur:

- **Search for Books:** Guests can browse the available books in the library but have limited functionalities compared to Members.

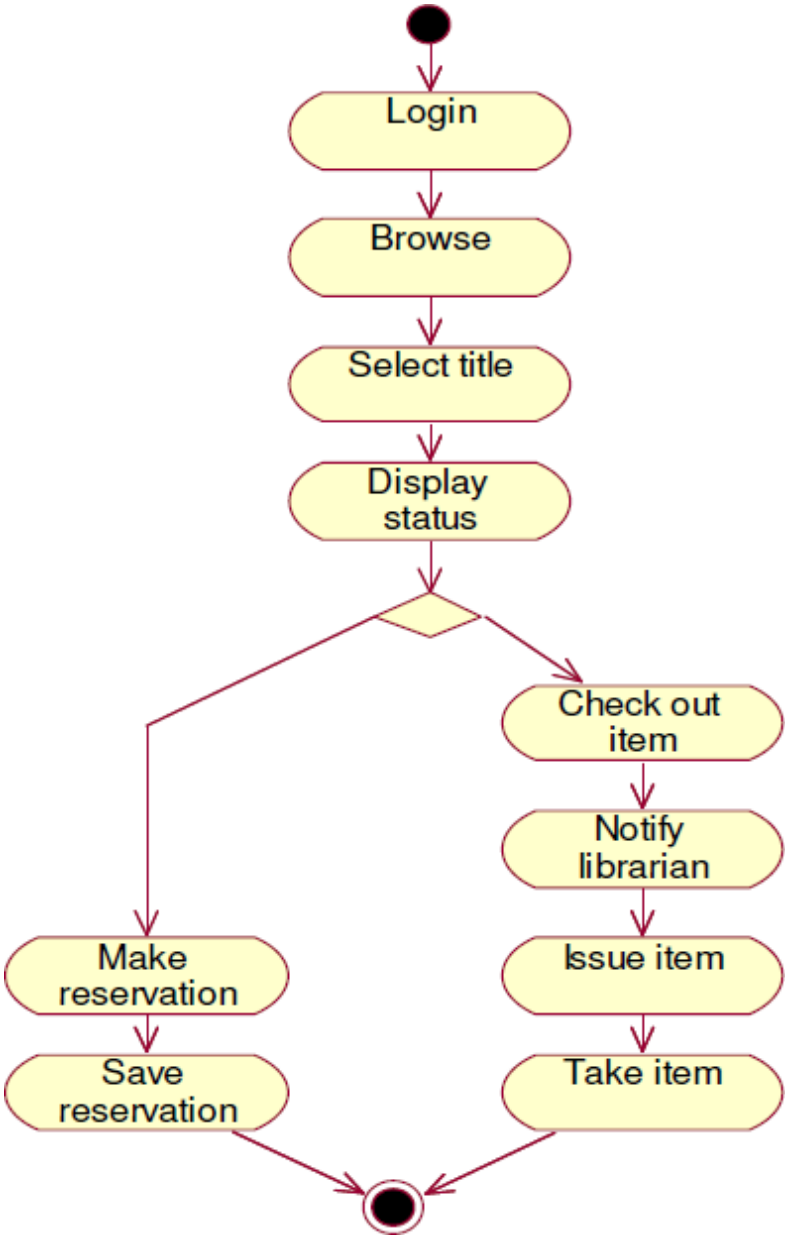
7. **End:**

The diagram concludes with an end node, indicating the completion of the activities.

Additional Considerations:

- **Decision Points:** Decision points may be included to represent conditions, such as checking whether a user is a Librarian, Member, or Guest.
- **Loops:** Loops can be added to represent repetitive activities, such as searching for multiple books or borrowing multiple books.
- **Parallel Activities:** Parallel activities may be depicted to show simultaneous processes, such as a Librarian managing memberships while a Member searches for books.

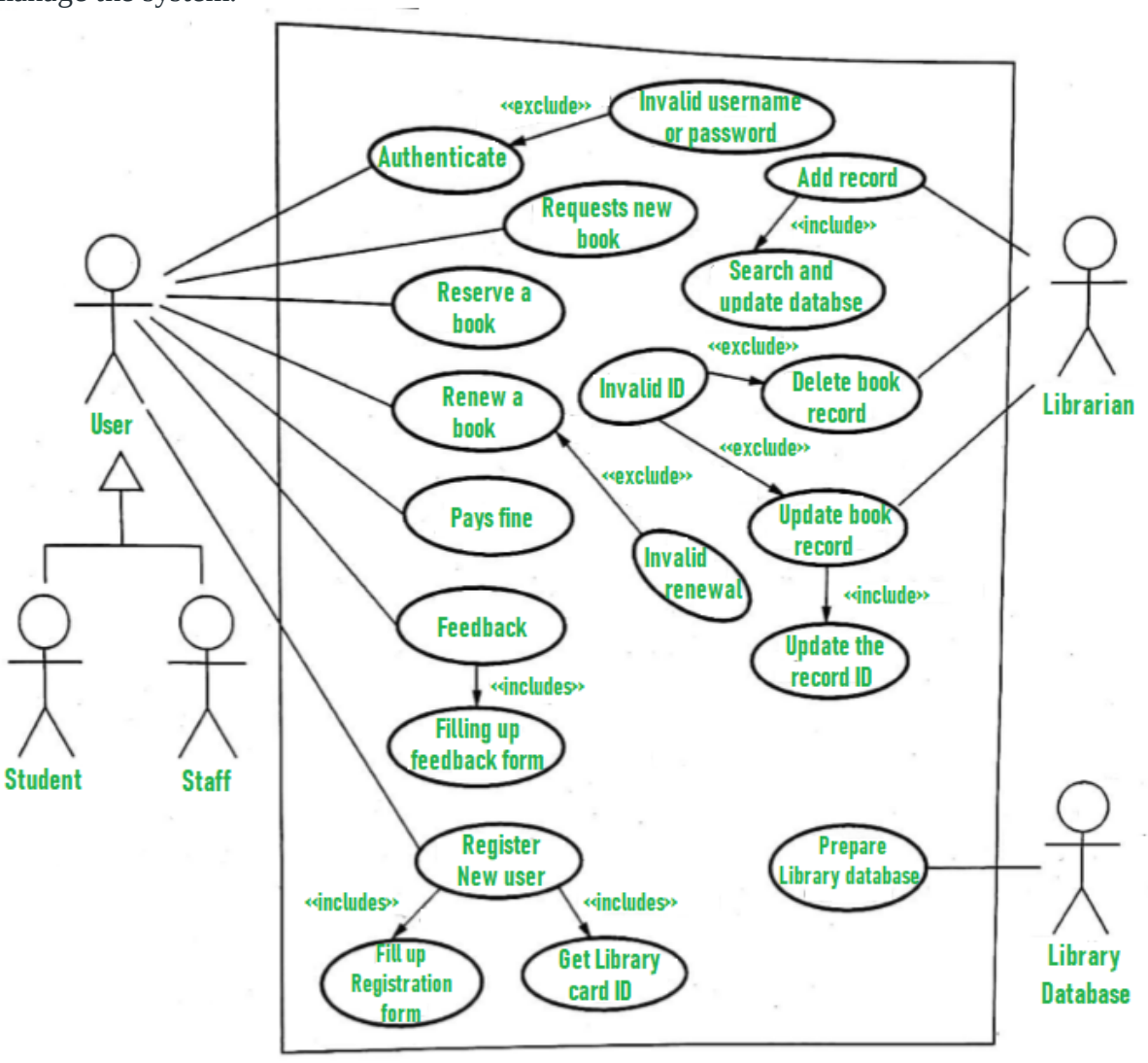
This activity diagram provides a visual representation of the sequential activities within the Library Management System, highlighting the interactions between users and the system.


Login
Browse
Select title
Display status
Check out item
Notify librarian
Issue item
Take item
Make reservation
Save reservation

3. Design an UML representation of interactions in an Online Book Shop, emphasizing system components, user roles, transaction processes, and scalability features for future enhancements.

Use Case Diagram:

Use case diagrams referred as a behavior model or diagram. It simply describes and displays the relation or interaction between the users or customers and providers of application service or the system. It describes different actions that a system performs in collaboration to achieve something with one or more users of the system. Use case diagram is used a lot nowadays to manage the system.



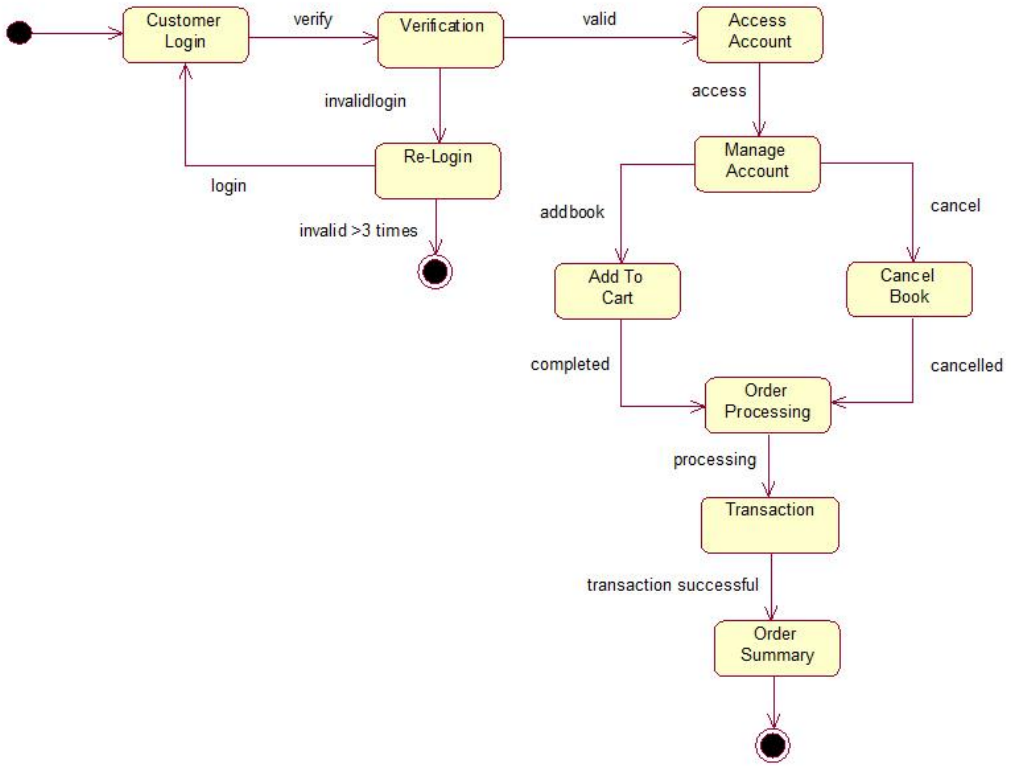
Here, we will understand the designing use case diagram for the library management system. Some scenarios of the system are as follows:

1. User who registers himself as a new user initially is regarded as staff or student for the library system.
    - For the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user.

- After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.

2. After getting the library card, a new book is requested by the user as per there requirement.
3. After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.
4. Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.
5. If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.
6. User can fill the feedback form available if they want to.
7. Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying fine.
8. Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.
9. Updating database is the important role of Librarian.

Activity Diagram for Online Book Shop



An activity diagram is a type of UML (Unified Modeling Language) diagram that visually represents the flow of actions within a system or a process. In the context of an online bookshop, an activity diagram can illustrate the different activities and interactions involved in the process of purchasing a book online. Here's a description of the key elements you might

include in an activity diagram for an online bookshop:

- **Start/End Point:** The diagram typically starts with a circle to represent the beginning and end of the process.
- **Customer Activities:** The customer initiates the process by browsing the online bookshop.
- Activities include searching for books, viewing book details, and adding books to the shopping cart.
- **Account Management:** If the customer is a registered user, there might be activities related to logging in or creating an account.
- **Shopping Cart**: Once a customer selects a book, it goes into the shopping cart.
- The customer can add or remove items from the cart.
- **Checkout Process**: Activities related to initiating the checkout process.
- This may involve providing shipping information, selecting a payment method, and confirming the order.
- **Payment Processing:** Interaction with payment gateways to handle the financial transaction securely.
- Activities such as entering credit card information or using other payment methods.
- **Order Confirmation**: After successful payment, the customer receives an order confirmation.

  This might involve sending an email or displaying a confirmation message on the website.
- **Inventory Management**: Behind the scenes, there are activities related to managing the book inventory.

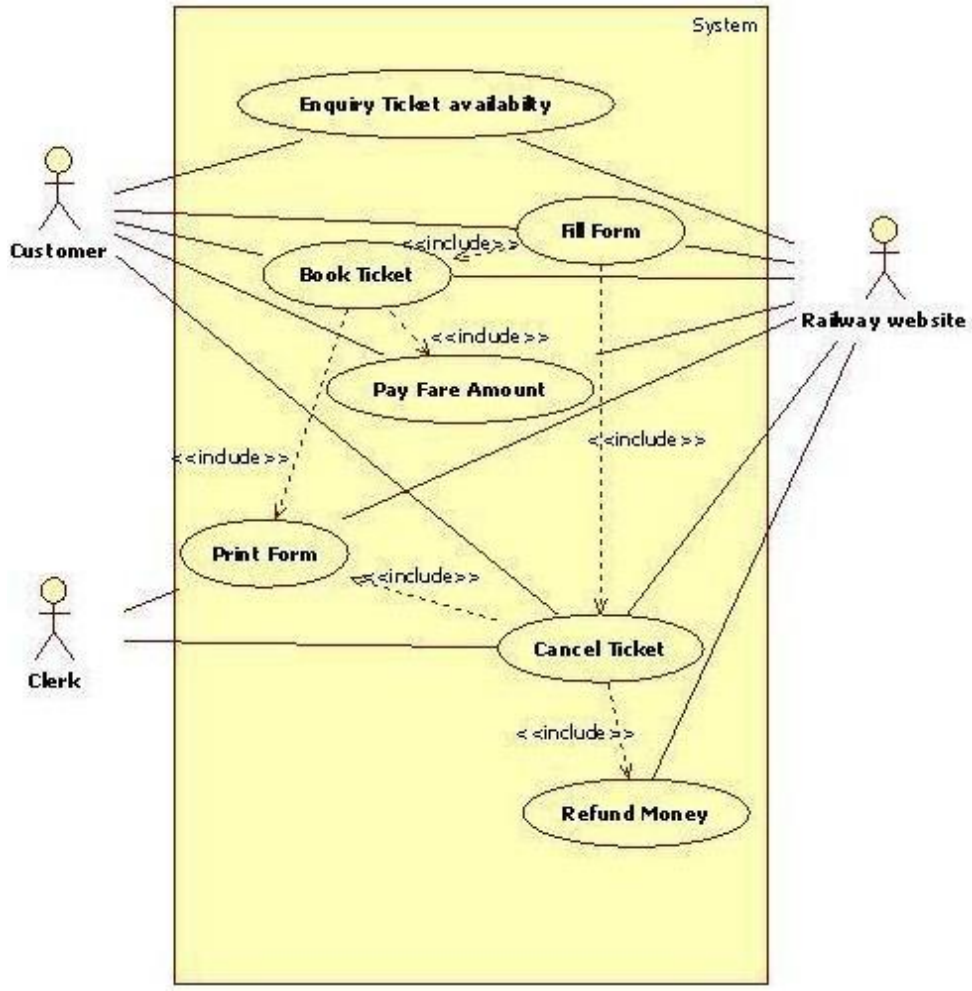
  Updating stock levels after a successful purchase.
- **Shipping Process:** If physical books are being shipped, activities related to packaging and dispatching the order.

  Generating shipping labels and tracking information.
- **Customer Support:** Activities related to customer support, such as handling inquiries or addressing issues with orders.
- **Feedback and Reviews:** Optionally, activities related to collecting customer feedback and reviews.
- **End Point:** The process concludes with the delivery of the books and customer satisfaction.
- Activity diagrams provide a high-level overview of the flow of activities without

detailing the internal workings of each activity. They are useful for understanding the sequential order of actions and the interactions between different elements in a system.

4. Design the UML diagram for a Railway Reservation System, emphasizing interactions, user roles, booking processes, and scalability features tailored for accommodating future system enhancements.

Use Case Diagram:



A use case diagram for a railway reservation system provides a high-level view of the system's functionality from the perspective of different actors (users or external systems) and the interactions they have with the system. Here's a description of key elements in a use case diagram for a railway reservation system:

Actors:

Customer:
Represents individuals who use the system to make reservations, check train schedules, and manage their bookings.

Administrator:
Represents system administrators who have the authority to manage users, trains, and other system-related tasks.

Train System:
Represents external systems or services that might interact with the railway reservation system, such as a payment gateway.

Use Cases:

Register Account:
The Customer initiates this use case to create a new account in the system, providing necessary details.

Log In:
Both Customers and Administrators can log into the system using their credentials.
Search Trains: Customers can search for available trains based on criteria like source, destination, date, etc.

Book Ticket:
Customers can select a train, specify the number of tickets, and make a reservation.

Cancel Reservation:
Customers can cancel their existing reservations.

Manage Bookings:
Customers can view and modify their existing reservations, check PNR status, etc.

Manage Trains:
Administrators can add, edit, or delete train information, including schedules and seat availability.

Manage Users:
Administrators can add or remove users, reset passwords, etc.

Generate Reports:
Administrators can generate reports related to bookings, revenue, or other system metrics.

Process Payment:
The system interacts with external payment systems to process payments for reservations.

Associations:

- Associations between actors and use cases show which actors are involved in each use case. For example, the "Book Ticket" use case involves the "Customer" actor.
- Associations between use cases indicate dependencies or flow of actions. For instance, "Book Ticket" might be dependent on "Search Trains."

Include and Extend Relationships:

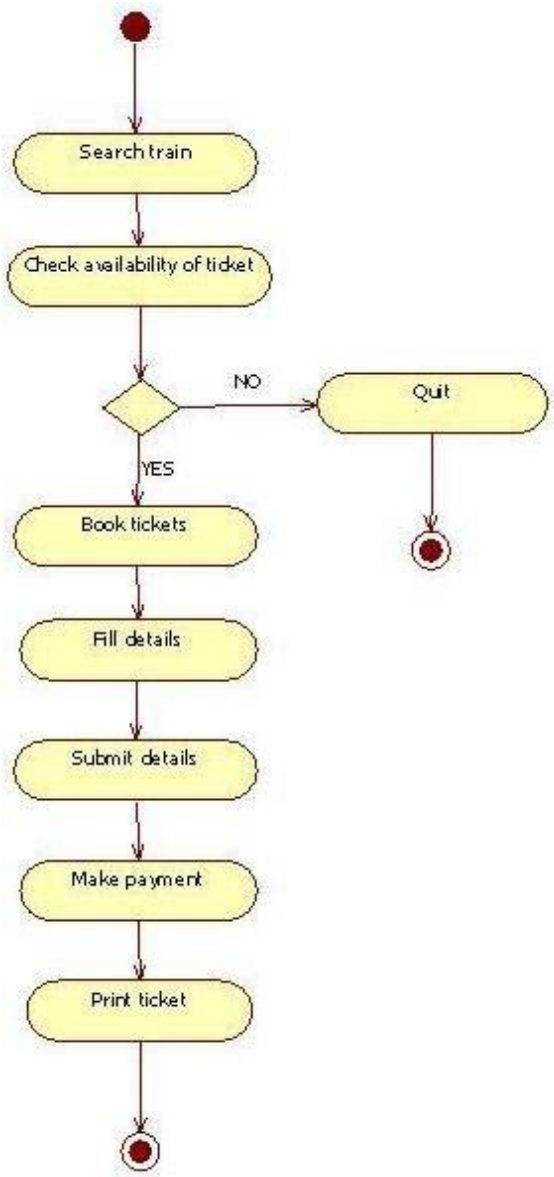
Include Relationship:
Represents a situation where one use case includes another. For instance, "Book Ticket" may include "Process Payment."
Extend Relationship:
Represents optional or conditional behavior that can extend the functionality of a base use case.

For example, "Cancel Reservation" might extend "Manage Bookings."

System Boundary:

- Represents the boundary of the railway reservation system, encapsulating all the use cases and actors involved.
- A use case diagram visually illustrates how various actors interact with the system and the different functionalities provided by the system. It serves as a valuable tool for communication and understanding the system's high-level behavior.

RRS Activity Diagram for Booking Ticket:



- **Enter Details:**

  The process starts with the user entering journey details, including the source station, destination station, travel date, and the number of passengers.

- **Search Available Trains:**

  The system performs a search for available trains based on the entered journey details. It retrieves a list of trains that match the criteria.

- **Select Train:**

  The user reviews the list of available trains and selects a preferred option. This involves considering factors such as departure time, duration, and class of service.

- **Enter Passenger Details:**

  After selecting a train, the user enters details for each passenger, including names, ages, and any other required information.

- **Choose Seat/Class:**

  The user chooses specific seats or selects a travel class based on preferences. This step may involve selecting seat types (e.g., window seat) or specifying class options (e.g., economy or first class).

- **Confirm Reservation:**

  The system validates the entered details, ensuring they meet the system's requirements. If everything is in order, the reservation is confirmed, and the system proceeds to the next step.

- **Process Payment:**

  The system interacts with an external payment gateway to process the payment for the reservation. This step involves secure payment processing, and the user may need to provide payment details.

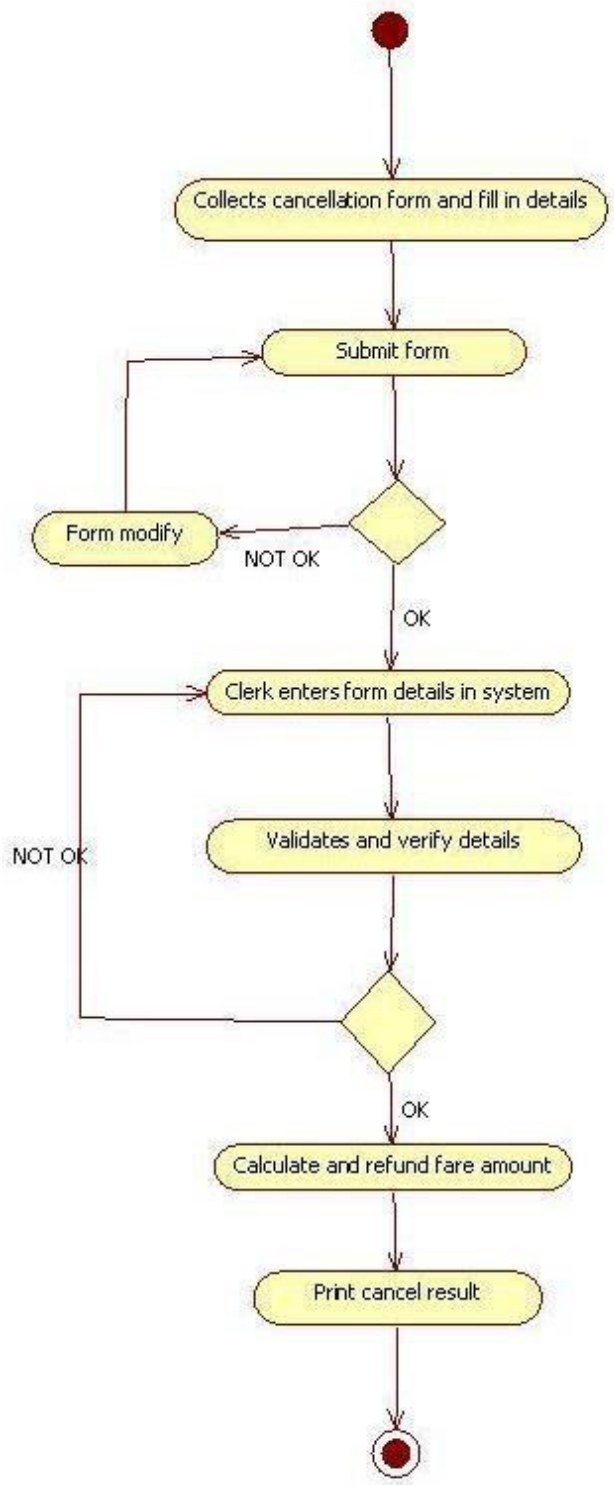
- **Generate Ticket:**

  After successful payment, the system generates a ticket with all the relevant details, including train information, passenger details, seat assignments, and a unique booking reference (PNR).

- **Display Ticket:**
  - The generated ticket is displayed to the user on the system interface.
  - This allows the user to review the booking details, verify correctness, and confirm the successful completion of the reservation.
  - The activity diagram captures the sequential flow of actions involved in the user's process of booking a ticket in the Railway Reservation System. It provides a clear visual representation of the steps, decisions, and interactions between the user and

the system throughout the booking process. This diagram is a valuable tool for understanding the user's journey and can be used for system design and communication purposes**.**

RRS Activity Diagram for Cancelling Ticket:



- **Enter Booking Details:**
  The cancellation process begins with the user providing necessary details to identify the

booking that needs to be canceled. This typically includes entering the booking reference (PNR), ticket number, or other relevant identification information.

- **Retrieve Booking Info:**

- The system retrieves the booking information associated with the details provided by the user.
- This information includes details about the booked ticket, such as train details, passenger names, and other relevant data.

- **Confirm Cancellation:**

The user confirms the cancellation request. At this stage, the system may validate the request against certain conditions, such as checking if the cancellation is within the allowed time frame or ensuring that the ticket is eligible for cancellation based on the fare rules.

- **Process Refund:**

If the cancellation is eligible for a refund according to the system's rules and policies, the system initiates the refund process. This involves interacting with the payment gateway to refund the applicable amount to the user's account. The amount refunded may be subject to cancellation charges or other rules.

- **Update Database:**

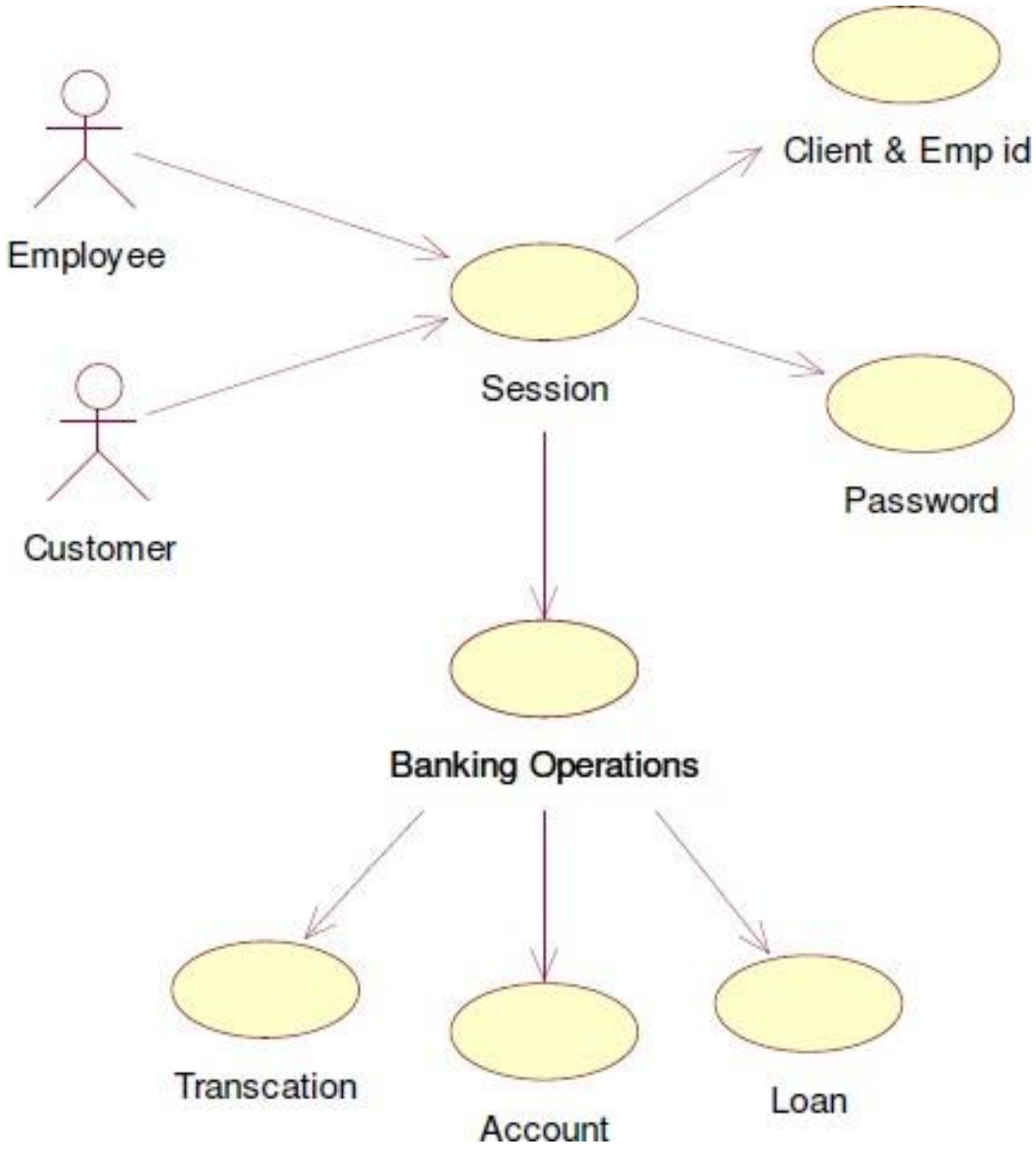
The system updates its database to reflect the canceled ticket. This involves adjusting seat availability, updating passenger records, and recording the cancellation details for reporting purposes.

- **Display Cancellation Confirmation:**

- The system provides a confirmation to the user that the ticket has been successfully canceled.
- This confirmation includes relevant details such as the refunded amount, if applicable. It serves to inform the user about the completion of the cancellation process.
- The activity diagram visually represents the sequential flow of actions involved in canceling a ticket in the Railway Reservation System. It helps stakeholders, including system designers and users, understand the steps and interactions involved in the cancellation process. Note that the actual implementation details may vary based on the specific business rules and requirements of the railway reservation system.

5. Demonstrate the UML representation of the Banking System's Account Transfer Process, highlighting interactions, user roles, transaction sequence, security measures, validation, exception handling, and scalability for future enhancements.

Banking System Use Case Diagram:



Actor: Customer

- **Open Account:**
  The customer can initiate the process of opening a new bank account. The system collects necessary information, verifies it, and generates an account.

- **Deposit Money:**
  Customers can deposit money into their accounts through various channels, such as over-the-counter transactions, ATMs, or mobile banking.

- **Withdraw Money:**
  Customers can initiate a withdrawal of funds from their accounts through ATMs, over-the-counter transactions, or online banking.

- **Transfer Funds:**
  Customers can transfer funds between their own accounts or to other accounts within the same bank or to external accounts.

- **Check Balance:**
  Customers can check the balance of their accounts through ATMs, online banking, or by visiting a bank branch.

Actor: Bank Teller

- **Process Deposit:**
  The bank teller can assist customers in processing deposits over the counter, verifying the transaction details and updating the account balance.

- **Process Withdrawal:**
  The bank teller can assist customers in processing withdrawals over the counter, verifying the transaction details and updating the account balance.

- **Open Account for Customer:**
  The bank teller can assist customers in opening new accounts by collecting necessary information and completing the account setup process.

Actor: System

- **Generate Monthly Statement:**
  The system generates monthly account statements for customers, providing a summary of transactions and the current balance.

- **Verify Customer Information:**
  The system verifies customer information during account opening, ensuring accuracy and compliance with regulatory requirements.

- **Process Fund Transfer:**
  The system processes fund transfers initiated by customers, updating the respective accounts and ensuring transaction security.

Actor: Manager

- **Monitor Transactions:**
  The manager can monitor and review transaction logs and reports to ensure the security and compliance of the banking system.

- **Approve Loan Application:**
  The manager has the authority to review and approve or reject loan applications submitted by customers.

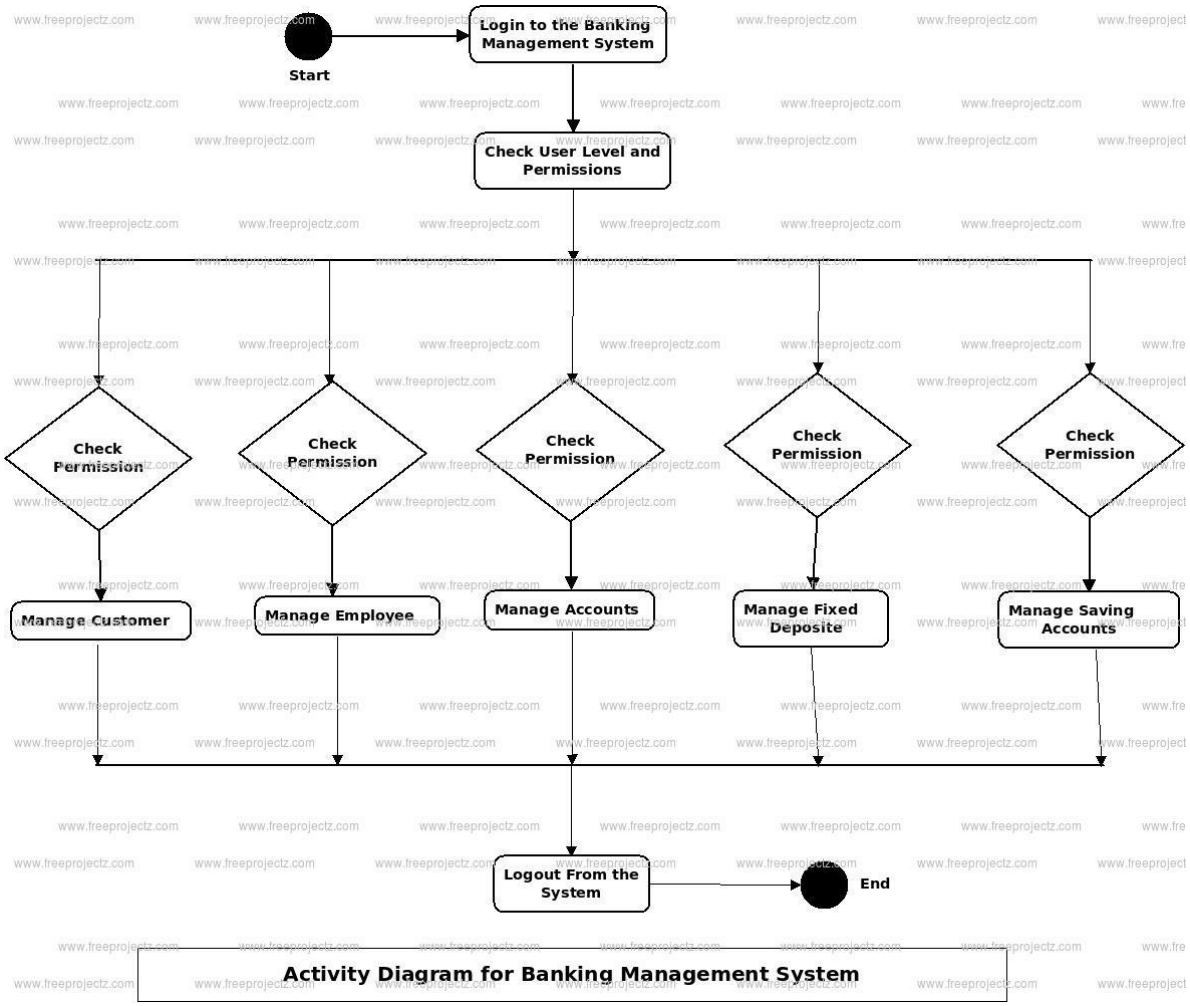
Actor: ATM

- **Dispense Cash:**
  The ATM dispenses cash to customers who request withdrawals, updating account balances accordingly.

- **Accept Deposits:**

- The ATM allows customers to deposit cash or checks, updating account balances accordingly.
- This use case diagram provides an overview of the interactions between different actors and the banking system, capturing the primary functionalities and transactions in a typical banking environment.

Activity diagram for a Banking System:



Activity: Customer Account Management

- **Open Account:**
  The customer initiates the account opening process by providing necessary information. The system verifies the information, generates an account, and sends a confirmation to the customer.

- **Deposit Money:**
  The customer chooses the deposit option and specifies the amount. The system updates the account balance accordingly and generates a deposit confirmation.

Activity: Fund Transfer

- **Transfer Funds:**
  The customer selects the fund transfer option, specifies source and destination accounts, and provides the transfer amount. The system verifies the accounts and updates the balances of both accounts.

Activity: Withdrawal

- **Withdraw Money**
  The customer requests a withdrawal, specifying the amount. The system checks the account balance, processes the withdrawal, and updates the account balance accordingly.

Activity: Banking Operations

- **Monthly Statement Generation**
  The system periodically generates monthly statements for each customer, summarizing transactions and providing the current balance.

- **Loan Application**
  The customer applies for a loan. The system reviews the application, checks eligibility, and either approves or rejects the loan. A notification is sent to the customer.

Activity: Bank Teller Operations

- **Over-the-Counter Transactions:**
  A bank teller assists customers in processing transactions such as deposits and withdrawals over the counter. The teller verifies details, updates the account balance, and issues relevant receipts.

- **New Account Setup:**
  The bank teller assists in the account opening process, collecting necessary information, verifying it, and completing the account setup.

Activity: ATM Transactions

- **ATM Withdrawal:**
  The customer uses an ATM to withdraw cash. The system verifies the account and dispenses cash while updating the account balance.

- **ATM Deposit:**
  The customer deposits money through an ATM. The system verifies the deposit and updates the account balance.

Activity: System Maintenance

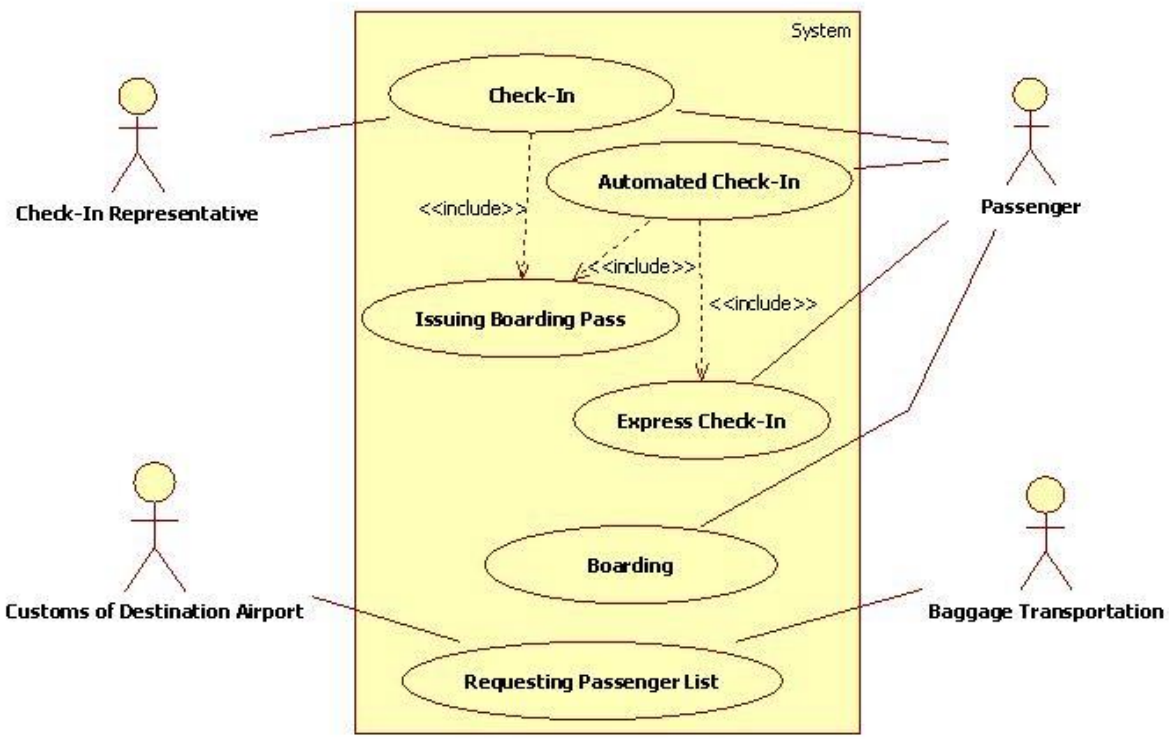
- **Security Checks:**
  The system regularly conducts security checks to ensure the integrity of customer data and protect against unauthorized access.

Database Maintenance:

- Routine maintenance tasks are performed on the database to optimize performance and ensure data integrity.
- This activity diagram illustrates the flow of activities within a banking system, highlighting interactions between customers, the system, bank tellers, and ATMs. It provides a comprehensive view of the key processes and operations involved in managing customer accounts and conducting banking transactions.

6. Draw a model for Airport management system in different views i.e. Use case view, logical view, component view, Deployment view, Database design, forward and Reverse Engineering, and Generation of documentation of the project.

Use Case Diagram for Airport Management System:



Actor: Passenger

- **Search for Flights**
  The passenger can search for available flights by providing criteria such as departure and destination airports, travel dates, and preferred class.

- **Book Flight:**
  The passenger can book a flight by selecting a specific flight, providing passenger details, and making a payment.

- **Cancel Reservation:**
  The passenger can cancel a booked flight reservation, subject to the airline's cancellation policy.

- **Check-in Online:**
  The passenger can perform online check-in, select seats, and receive a digital boarding pass.

Actor: Airport Staff

- **Check-in Counter Operations:**
  Airport staff can assist passengers with check-in at the airport counter, issue boarding passes, and handle special requests.

- **Baggage Handling:**
  Airport staff can manage the process of handling checked baggage, including tagging and loading onto the aircraft.

Actor: Flight Crew

- **Conduct Flight:**
  The flight crew is responsible for conducting the flight, ensuring passenger safety, and adhering to aviation regulations.

Actor: System

- **Manage Flight Schedule:**
  The system allows administrators to manage and update the schedule of flights, including adding new flights and modifying existing ones.

- **Manage Passenger Information:**
  The system manages passenger information, including reservations, check-in details, and preferences.

- **Generate Reports:**
  The system generates various reports related to flight bookings, passenger statistics, and financial data.

Actor: Maintenance Crew

- **Aircraft Maintenance:**
  The maintenance crew can use the system to schedule and log maintenance activities for aircraft to ensure their airworthiness.

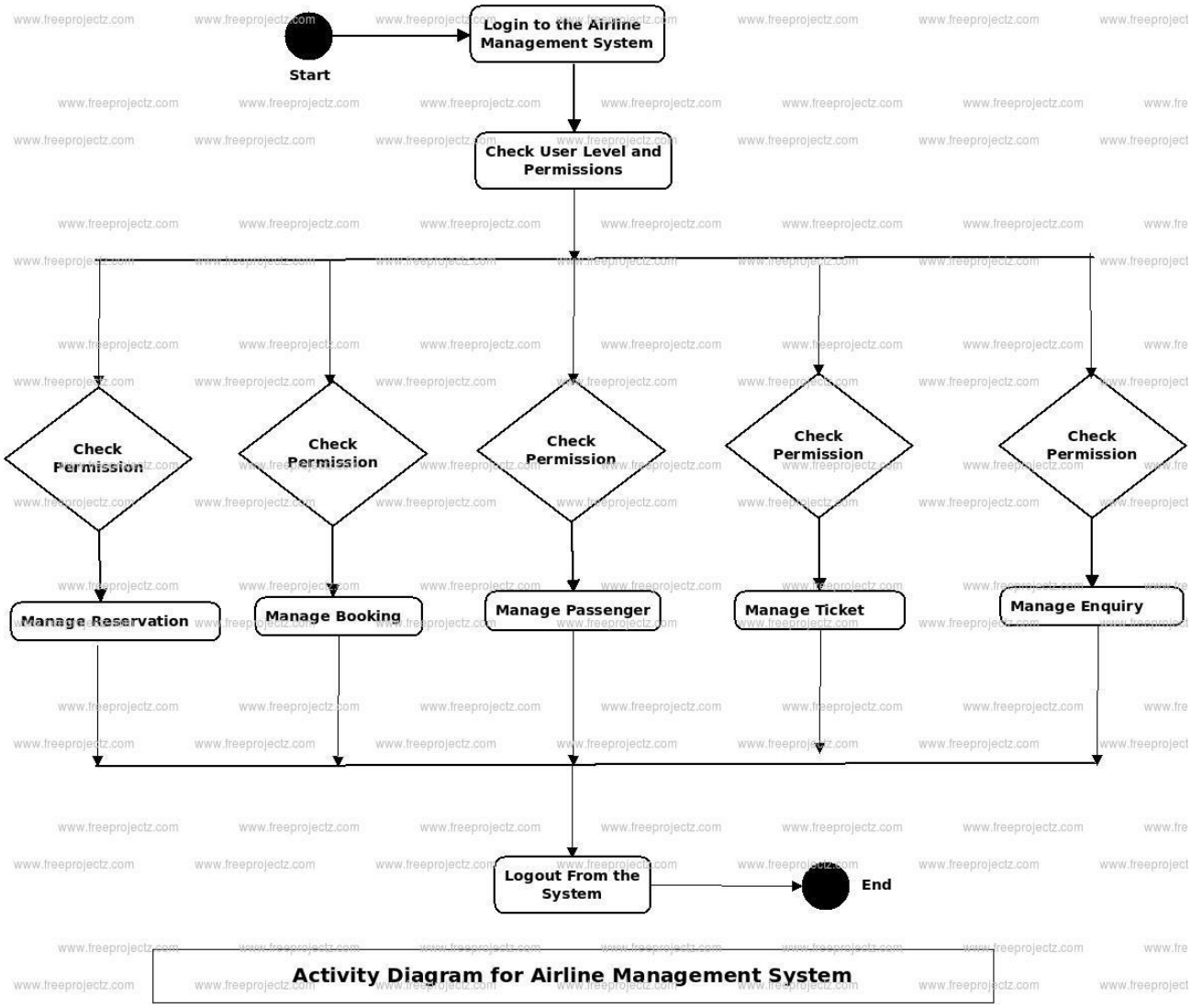
Actor: External System (e.g., Payment Gateway)

- **Process Payment:**

- The external payment gateway interacts with the system to process payments for flight bookings made by passengers.
- This use case diagram outlines the interactions between different actors and the Airline System. It provides a high-level view of the functionalities of the system, including

passenger-related actions, airport operations, flight management, and external interactions.

Activity Diagram for Airline Management System:



Activity: Booking a Flight

- **Search for Flights:**
  The passenger initiates a search for available flights by specifying the departure and destination airports, travel dates, and other preferences.

- **Select Flight:**
  The passenger chooses a specific flight from the search results. The system validates seat availability and provides the passenger with options.

- **Provide Passenger Information:**
  The passenger enters personal and travel information required for booking, including names, contact details, and any special requests.

- **Confirm Booking:**
  The passenger reviews the booking details, confirms the reservation, and makes the payment. The system updates the seat inventory and generates a booking confirmation.

Activity: Check-in Process:

- **Online Check-in:**
  The passenger initiates the online check-in process by providing booking details and passenger information.

- **Select Seats:**
  The passenger selects preferred seats during the online check-in process. The system validates seat availability and updates the seating arrangement.

- **Receive Boarding Pass:**
  Upon successful check-in, the system generates and provides the passenger with a digital boarding pass.

- **Baggage Drop:**
  The passenger drops off checked baggage at the airport. The system verifies baggage details and updates the handling process.

- **Security Check:**
  The passenger goes through the security check process. The system updates the passenger's status as cleared for boarding.

Boarding Process

- **Boarding The passenger boards the flight:**
  The system updates the boarding status and monitors passenger attendance.

- **Seat Verification:**
  The system verifies that each passenger is seated in their assigned seat, ensuring accurate passenger count and compliance with safety regulations.

In-flight Services:

- **Food and Beverage Service:**
  The flight crew provides food and beverage services to passengers. The system tracks inventory and updates service records.

- **Entertainment:**
  Passengers access in-flight entertainment services. The system monitors usage and updates entertainment logs.

Flight Completion:

- **Landing:**
  The flight lands at the destination airport. The system updates the flight status to "landed."

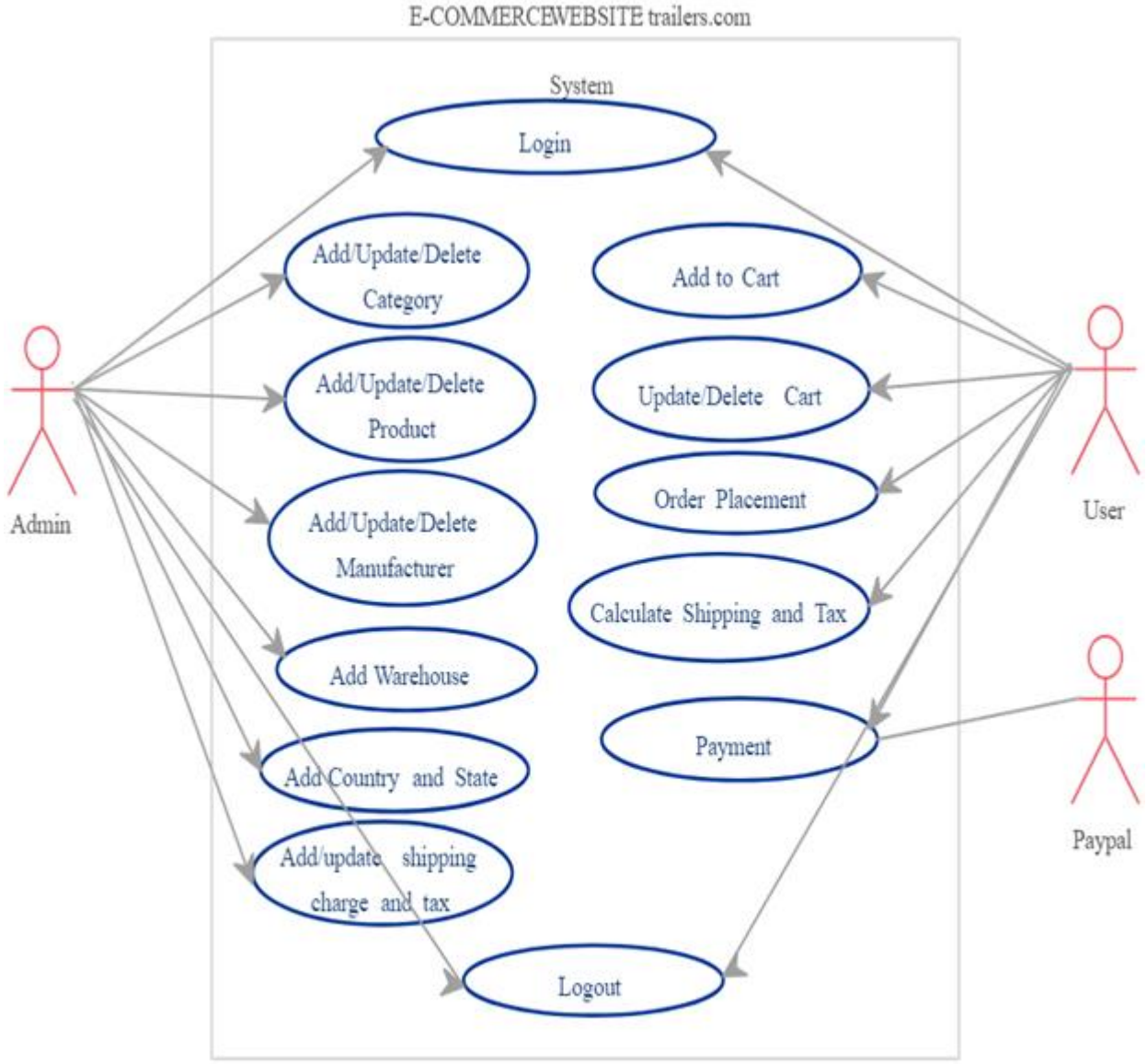
- **Baggage Retrieval:**
  Passengers retrieve their checked baggage. The system updates the baggage status and records successful retrieval.

System Maintenance:

- **Database Update:**
  - The system performs routine maintenance tasks, updating databases with flight schedules, passenger information, and other relevant data.
  - This activity diagram provides an overview of the processes involved in booking a flight, passenger check-in, airport operations, the boarding process, in-flight services, and flight completion within an airline system.

7. Draw a model for E-commerce sites in different views i.e Use case view, logical vie component view, Deployment view, Database design, forward and Reverse Engineering, and Generation of documentation of the project.

Use Case Diagram



A use case diagram for an e-commerce system illustrates the interactions and functionalities between different users (actors) and the system itself. Here's a description of the main components and interactions in a typical e-commerce use case diagram:

Actors:

- **Guest:**
- Represents a user who is not logged in but can browse products.

- **Registered User:**
- Represents a user who has created an account on the e-commerce platform.

- **Customer Support:**
- Represents customer support staff who can assist users with inquiries and issues.

- **Administrator:**
- Manages the overall functioning of the e-commerce system, including user accounts, product listings, and system configurations.

Use Cases:

Browse Products:
Actors - Guest, Registered User:
Users, whether guests or registered, can browse and search for products on the platform.

Register/Login:
Actors - Guest, Registered User:
Users can register for a new account or log in to an existing one to access personalized features, such as order history and preferences.

Add to Cart:
Actors - Registered User:
Registered users can add products to their shopping cart for future purchase.

Checkout:
Actors - Registered User:
Users can proceed to checkout, review their order, and provide shipping and payment information to complete the purchase.

View Order Status:
Actors: Registered User
Users can check the status of their orders, including processing, shipped, or delivered.

Manage Account:
Actors: Registered User
Users can update their account information, change passwords, or manage other account-related settings.

Provide Feedback/Rating:
Actors: Registered User
Users can leave feedback or ratings for products they have purchased.

Customer Support Interaction:
Actors: Registered User, Customer Support
Users can interact with customer support for inquiries, assistance, or issue resolution.

Administrator Functions:
Actors: Administrator
The administrator can manage user accounts, product listings, and system configurations.

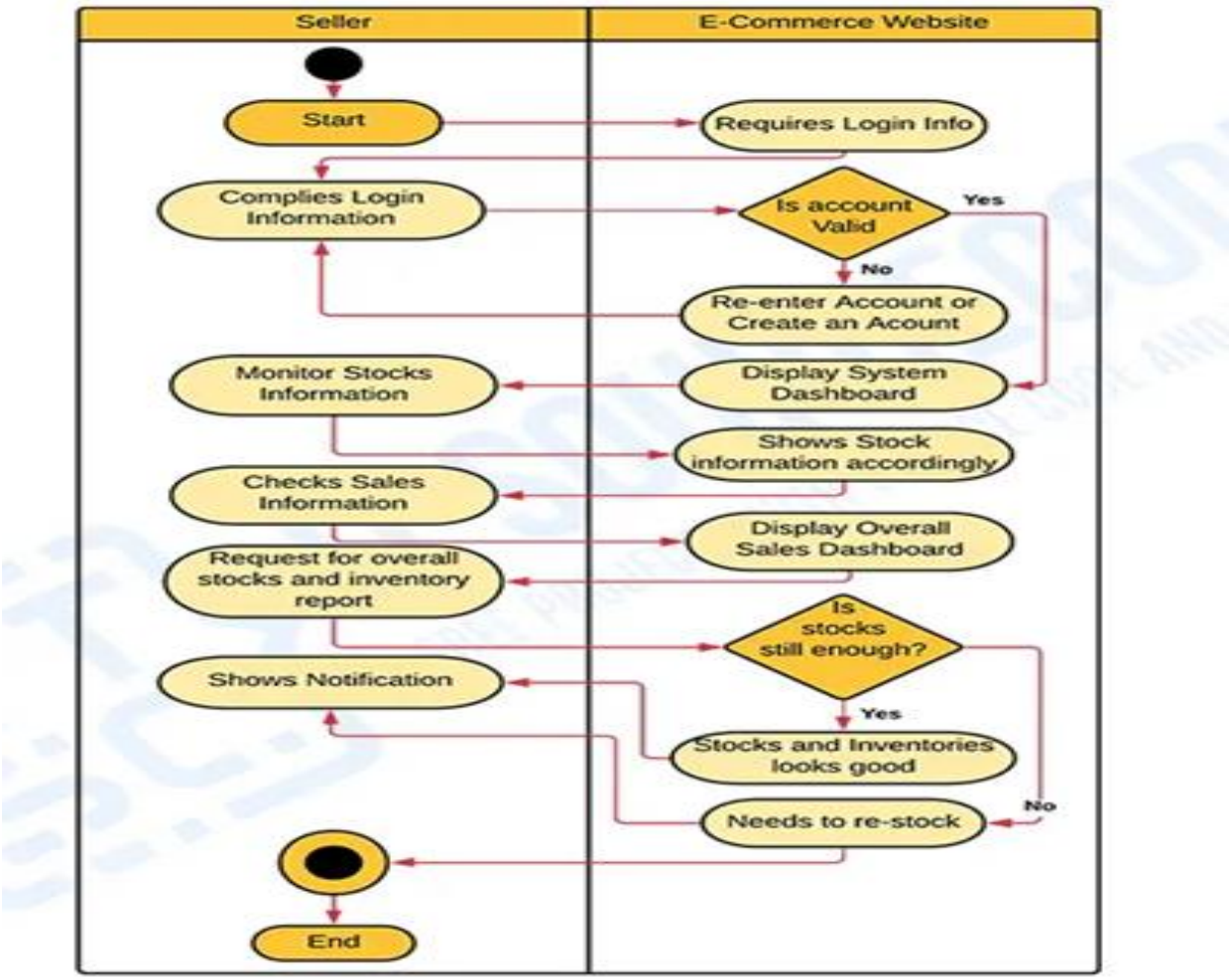
Associations:

- Relationships between actors and use cases are represented by lines connecting them.
- For example, there will be associations between the Guest actor and the Browse Products use case.

System Boundary:

- The boundary of the use case diagram represents the scope of the e-commerce system.
- This use case diagram provides a visual representation of the key functionalities and interactions within an e-commerce system, showing how different users interact with the system to perform specific tasks related to browsing, shopping, and managing their accounts.

Activity Diagram:



An activity diagram for an e-commerce system illustrates the workflow and activities involved in various processes, such as browsing products, placing orders, and managing accounts. Below is a description of the main components and activities in an activity diagram for an e-commerce system:

Browse and Search Products:

Activities:

- Start browsing
- Search for products
- Display product details

Flow:

- Initiated when a user starts browsing or searching for products.
- Involves searching for products and displaying their details.
- Ends when the user decides to view product details or continues browsing.

User Registration and Login:

Activities:

- Register for an account
- Log in
- Authenticate user

Flow:

- Initiated when a user decides to register or log in.
- Involves entering account details, authentication, and account creation or login.
- Ends when the user successfully registers or logs in.

Add to Cart:

Activities:

- Add product to the shopping cart
- Update shopping cart
- View shopping cart

Flow:

- Triggered when a user adds a product to the shopping cart.
- Involves updating the cart and optionally viewing the contents.
- Ends when the user decides to proceed to checkout or continue shopping.

Checkout Process:

Activities:

- Proceed to checkout
- Provide shipping details
- Provide payment details

Confirm order:

Flow:

- Initiated when the user decides to proceed to checkout.
- Involves entering shipping and payment details and confirming the order.
- Ends when the order is successfully placed or canceled.

Order Processing:

Activities:

- Process order
- Generate order confirmation
- Update inventory

Flow:

- Triggered when an order is confirmed.
- Involves processing the order, generating confirmation, and updating inventory.
- Ends when the order is processed.

Manage User Account:

Activities:

- View account details
- Update account information
- Change password

Flow:

- Initiated when a user decides to manage their account.
- Involves viewing and updating account information.
- Ends when the user completes account management tasks.

Customer Support Interaction:

Activities:

- Initiate support request
- Customer support responds
- Resolve issue

Flow:

- Represents the interaction between a user and customer support.
- Involves initiating a support request, customer support responding, and resolving the issue.
- Ends when the issue is resolved.

Administrator Functions:

Activities:

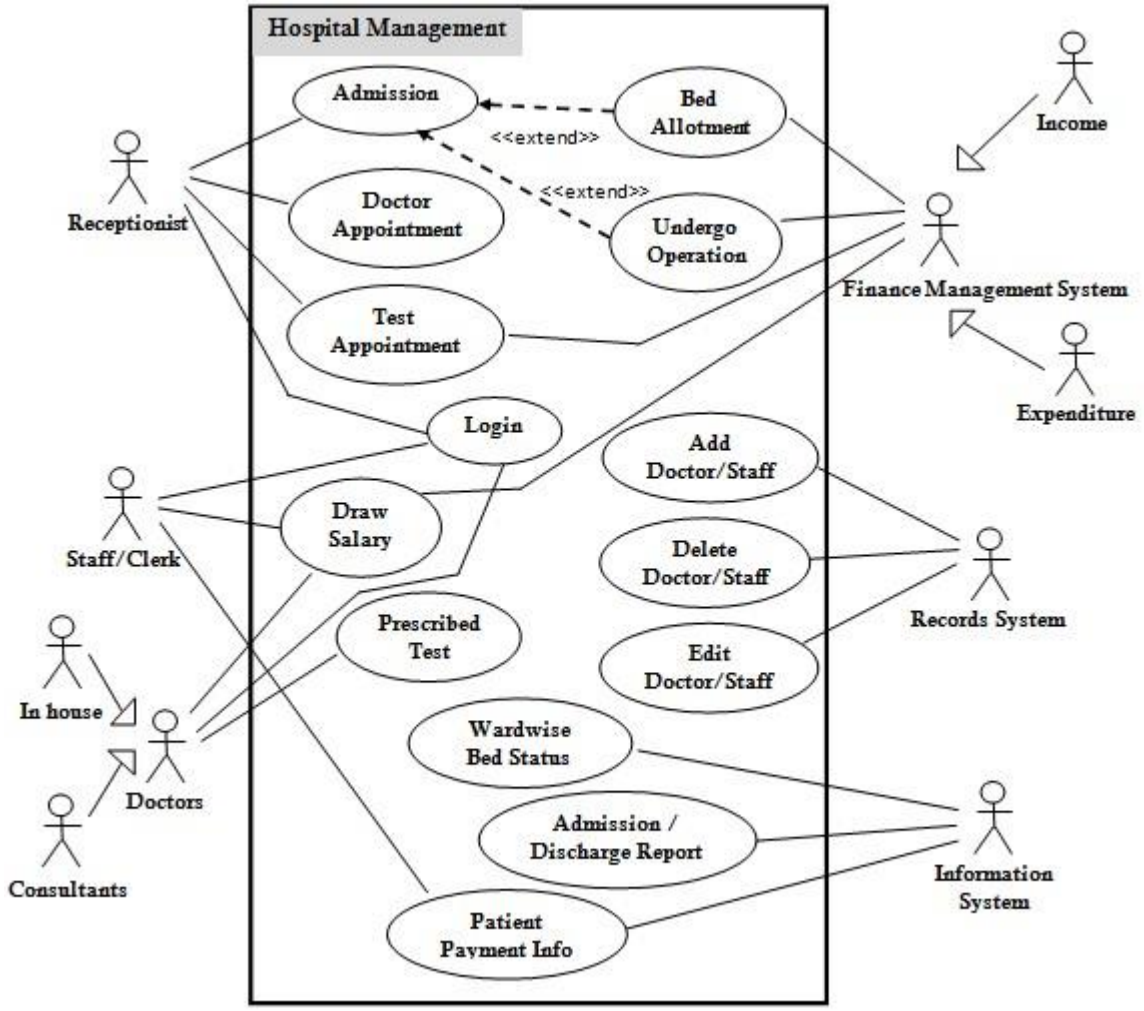
- Manage product listings
- Manage user accounts
- Monitor system performance

Flow:

- Represents activities performed by administrators.
- Involves managing product listings, user accounts, and monitoring system performance.
- These activity diagrams provide a visual representation of the sequential and parallel activities within an e-commerce system, illustrating how users interact with the system to perform various tasks related to shopping, account management, and support.

8. Design Activity and Class Diagram for Hospital management system to demonstrate the Activities which will be carried out in Hospital.

Use Case Diagram



A use case diagram provides a high-level overview of the functionalities and interactions of a system from the perspective of its users. In the context of a Hospital Management System, the use case diagram outlines the various actions or tasks that different users (actors) can perform within the system. Here's a description of the main components and interactions in a Hospital Management System use case diagram:

Actors:

- **Patient:** Represents individuals who seek medical assistance or treatment from the hospital.
- **Doctor:** Refers to the healthcare professionals responsible for diagnosing and treating patients.
- **Nurse:** Represents the nursing staff responsible for patient care and support.
- **Administrator:** Manages the overall functioning of the hospital system, including user

roles, permissions, and system configurations.

Use Cases:

Register/Login:
Actors: Patient, Doctor, Nurse, Administrator
Users should be able to register or log in to the system based on their roles to access personalized features.

Schedule Appointment:
Actors: Patient, Doctor
Patients can request appointments, and doctors can schedule or reschedule appointments.

Admit/Discharge Patient:
Actors: Administrator, Nurse
The administrator and nursing staff can admit or discharge patients from the hospital.

View Medical Records:
Actors: Patient, Doctor
Patients can view their medical records, while doctors can access and update patient records.

Prescribe Medication:
Actors: Doctor
Doctors can prescribe medications and treatments for patients.

Update Patient Status:
Actors: Nurse, Doctor
The nursing staff and doctors can update the status of patients based on their condition.

Generate Reports:
Actors: Doctor, Administrator
Doctors can generate medical reports for patients, and administrators can generate system-wide reports.

Manage Inventory:
Actors: Administrator
The administrator can manage the inventory of medical supplies and equipment.

Associations:

- Relationships between actors and use cases are represented by lines connecting them.
- For example, there will be associations between the Patient actor and use cases like Schedule Appointment, View Medical Records, and Register/Login.

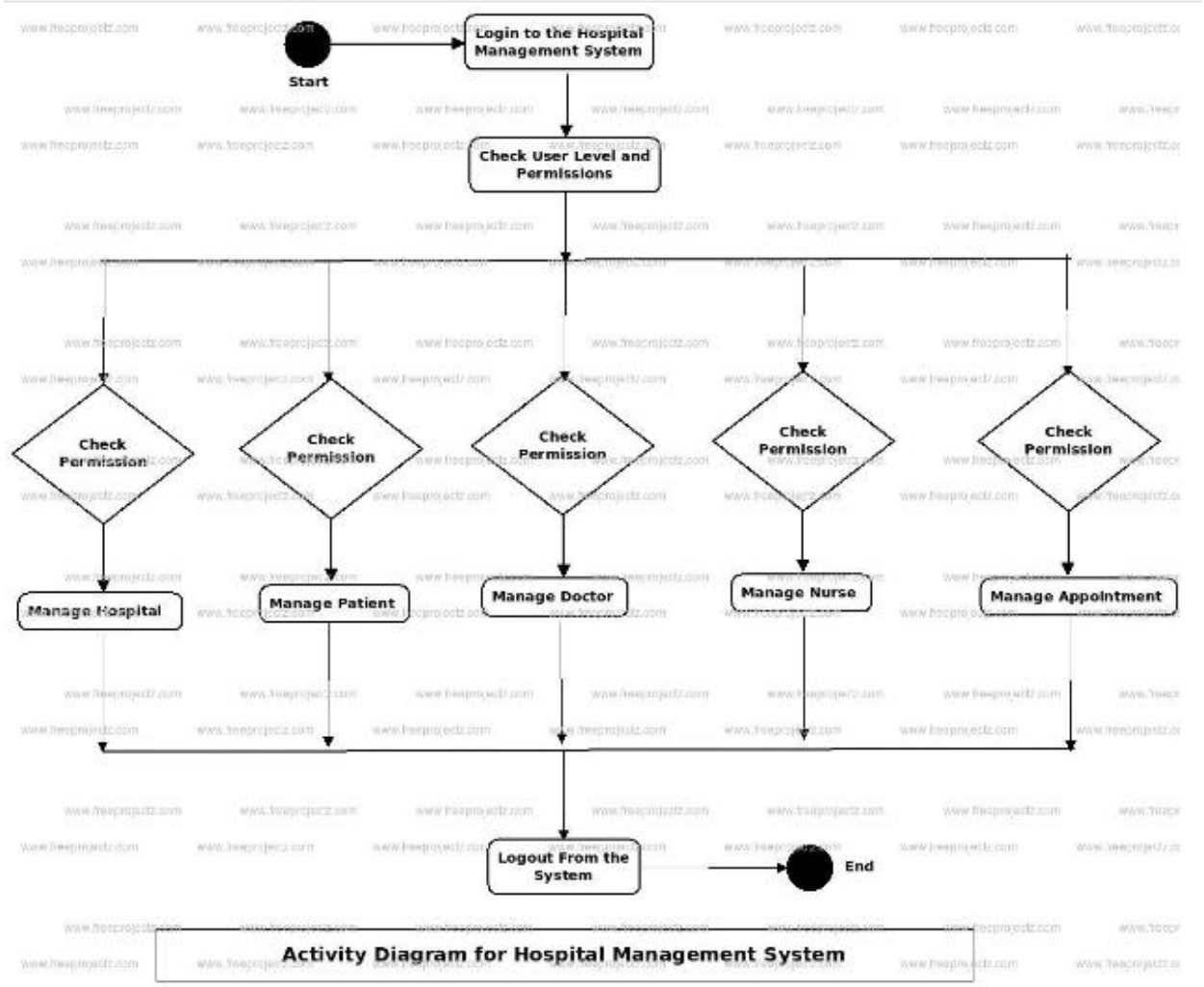
System Boundary:

- The boundary of the use case diagram represents the scope of the Hospital Management

System.

- This use case diagram provides a visual representation of the key functionalities and interactions within a Hospital Management System, offering a clear understanding of how different users interact with the system to perform specific tasks.

Activity Diagram:



An activity diagram for a Hospital Management System illustrates the flow of activities and actions within the system to achieve specific goals. Below is a description of the main components and activities in an activity diagram for a Hospital Management System:

Patient Registration Process:

Activities:

- Input patient information
- Verify patient details
- Create a patient record

Flow:

- Starts when a patient arrives at the hospital.
- Ends when the patient registration is complete, and a patient record is created.

Appointment Scheduling Process:

Activities:

- Request appointment
- Check doctor's availability
- Schedule appointment

Flow:

- by a patient requesting an appointment.
- Involves checking the availability of the requested doctor.
- Ends when the appointment is successfully scheduled.

Patient Admission Process:

Activities:

- Admit patient
- Assign room
- Record admission details

Flow:

- Triggered when a patient arrives for admission.
- Includes assigning a room and recording admission details.
- Ends when the patient is successfully admitted.

Medical Treatment Process:

Activities:

- Consultation
- Diagnosis
- Treatment

Flow:

- Started by a doctor conducting a consultation with a patient.
- Involves diagnosing the medical condition and prescribing treatment.
- Ends when the treatment is complete or ongoing.

Nursing Care Process:

Activities:

- Patient monitoring
- Medication administration
- Record patient status

Flow:

- Initiated by nursing staff responsible for patient care.
- Involves monitoring the patient, administering medications, and updating patient records.
- Ends when the nursing care tasks are complete.

Billing and Payment Process:

Activities:

- Generate bill
- Verify insurance details
- Process payment

Flow:

- Started when the patient's treatment is completed.
- Involves generating a bill, verifying insurance information, and processing the payment.
- Ends when the payment is successfully processed.

Inventory Management Process:

Activities:

- Monitor inventory levels
- Reorder supplies
- Update inventory records

Flow:

- Initiated by the administrator responsible for managing inventory.
- Involves monitoring levels, reordering supplies, and updating inventory records.
- Ends when the inventory management tasks are complete.

System Maintenance Process:

Activities:

- Regular updates
- Security checks
- Backup procedures

Flow:

- Represents routine maintenance tasks carried out by the system administrator.
- Involves activities such as software updates, security checks, and data backups.
- Ends when the system maintenance tasks are completed.

These activity diagrams provide a visual representation of the sequential and parallel activities within the Hospital Management System, showcasing the flow of actions from the initiation of various processes to their completion.