SRI VASAVI ENGINEERING COLLEGE

(Autonomous)

Department Of Computer Science and Engineering, Pedatadepalli, Tadepalligudem.

Certificate

This is to certify that this is a bonafide record of Practical Work done in **Unified Modeling Language Lab (V20CSL13)** by Mr./Miss______________________________bearing Roll No.________________of CSE Branch of VI Semester during the academic year 2023 to 2024.

Faculty Incharge **Head of the Department**

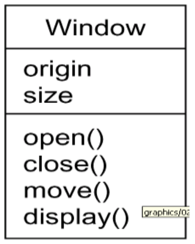
External Examiner

Index

S. No.	Name Of the Experiment	Date Performed	Date Submitted	Signature
1.	Usecase Diagram			
2.	Class Diagram			
3.	Object Diagram			
4.	Sequence Diagram & Communication Diagram			
5.	Statechart Diagram			
6.	Activity Diagram			
7.	Component Diagram			
8.	Deployment Diagram			
9.	CaseStudy-1**(**Library Management System**)**			
10.	CaseStudy-2**(**Hospital Management System**)**			
11.	CaseStudy-3**(**Railway Reservation System**)**			

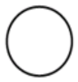
The UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.

Class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations.



Interface

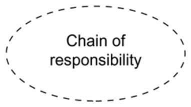
Interface is a collection of operations that specify a service of a class or component.
An interface therefore describes the externally visible behavior of that element.
An interface might represent the complete behavior of a class or component or only a part of that behavior.
An interface is rendered as a circle together with its name. An interface rarely stands alone. Rather, it is typically attached to the class or component that realizes the interface



I Spelling

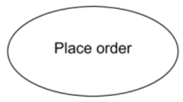
Collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Therefore, collaborations have structural, as well as behavioral, dimensions. A given class might participate in several collaborations.

Graphically, a collaboration is rendered as an ellipse with dashed lines, usually including only its name

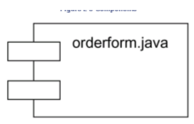


Usecase

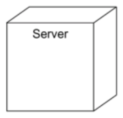
- Use case is a description of set of sequence of actions that a system performs that yields an observable result of value to a particular actor
- Use case is used to structure the behavioral things in a model.
- A use case is realized by a collaboration. Graphically, a use case is rendered as an ellipse with solid lines, usually including only its name



Component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Graphically, a component is rendered as a rectangle with tabs

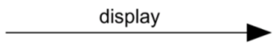


Node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability. Graphically, a node is rendered as a cube, usually including only its name



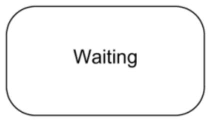
Interaction

Interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose
An interaction involves a number of other elements, including messages, action sequences and links
Graphically a message is rendered as a directed line, almost always including the name of its operation



State Machine

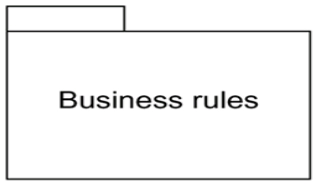
State machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events
State machine involves a number of other elements, including states, transitions, events and activities
Graphically, a state is rendered as a rounded rectangle, usually including its name and its substate



Package:-

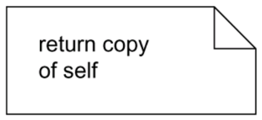
- A package is a general-purpose mechanism for organizing elements into groups. Structural things, behavioral things, and even other grouping things may be placed in a package

- Graphically, a package is rendered as a tabbed folder, usually including only its name and, sometimes, its contents



A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.

Graphically, a note is rendered as a rectangle with a dog-eared corner, together with a textual or graphical comment



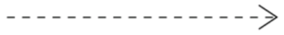
Relationships in the UML: There are four kinds of relationships in the UML:

1. Dependency
2. Association
3. Generalization
4. Realization

Dependency:-

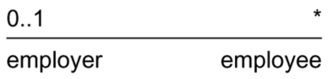
Dependency is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing

Graphically a dependency is rendered as a dashed line, possibly directed, and occasionally including a label

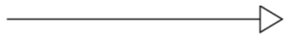


Association is a structural relationship that describes a set of links, a link being a connection among objects.

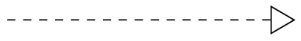
Graphically an association is rendered as a solid line, possibly directed, occasionally including a label, and often containing other adornments, such as multiplicity and role names



Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent



Realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out. Graphically a realization relationship is rendered as a cross between a generalization and a dependency relationship



Diagrams in the UML

- **Diagram** is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).
- In theory, a diagram may contain any combination of things and relationships.
- For this reason, the UML includes nine such diagrams:
    - Class diagram
    - Object diagram
    - Use case diagram
    - Sequence diagram
    - Collaboration diagram
    - Statechart diagram
    - Activity diagram
    - Component diagram
    - Deployment diagram

Class diagram

A class diagram shows a set of classes, interfaces, and collaborations and their relationships.
Class diagrams that include active classes address the static process view of a system.

Object diagram

- Object diagrams represent static snapshots of instances of the things found in class diagrams
- These diagrams address the static design view or static process view of a system
- An object diagram shows a set of objects and their relationship.

Use case diagram

- A use case diagram shows a set of use cases and actors and their relationships
- Use case diagrams address the static use case view of a system.
- These diagrams are especially important in organizing and modeling the behaviors of a system.

Interaction Diagrams

Both sequence diagrams and collaboration diagrams are kinds of interaction diagrams
Interaction diagrams address the dynamic view of a system
A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages
A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages

Sequence diagrams and collaboration diagrams are isomorphic, meaning that you can take one and transform it into the other

Statechart diagram

- A statechart diagram shows a state machine, consisting of states, transitions, events, and activities
- Statechart diagrams address the dynamic view of a system
- They are especially important in modeling the behavior of an interface, class, or collaboration and emphasize the event-ordered behavior of an object

Activity diagram

An activity diagram is a special kind of a statechart diagram that shows the flow from activity to activity within a system
Activity diagrams address the dynamic view of a system
They are especially important in modeling the function of a system and emphasize the flow of control among objects.

Component diagram

- A component diagram shows the organizations and dependencies among a set of components.
- Component diagrams address the static implementation view of a system
- They are related to class diagrams in that a component typically maps to one or more classes, interfaces, or collaborations.

Deployment diagram

- A deployment diagram shows the configuration of run-time processing nodes and the components that live on them
- Deployment diagrams address the static deployment view of an architecture

Why we need UML Lab?

The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.The overall goal of UML diagrams is to allow teams to visualize how a project is or will be working, and they can be used in any field, not just software engineering.

Task-1: Draw basic Usecase diagrams for capturing and representing requirements of the system.

Usecase diagram:

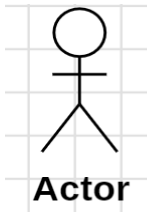
- A usecase diagram shows a set of use cases and actors and their relationships.it address the static usecase view of a system.
- These diagrams are especially important in organizing and modeling the behaviours of a system.

Use Cases:

- A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

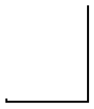
Actors:

- An actor is a person, organization, or external system that plays a role in one or more interactions with the system. Four relationships among use cases are used often in practice.

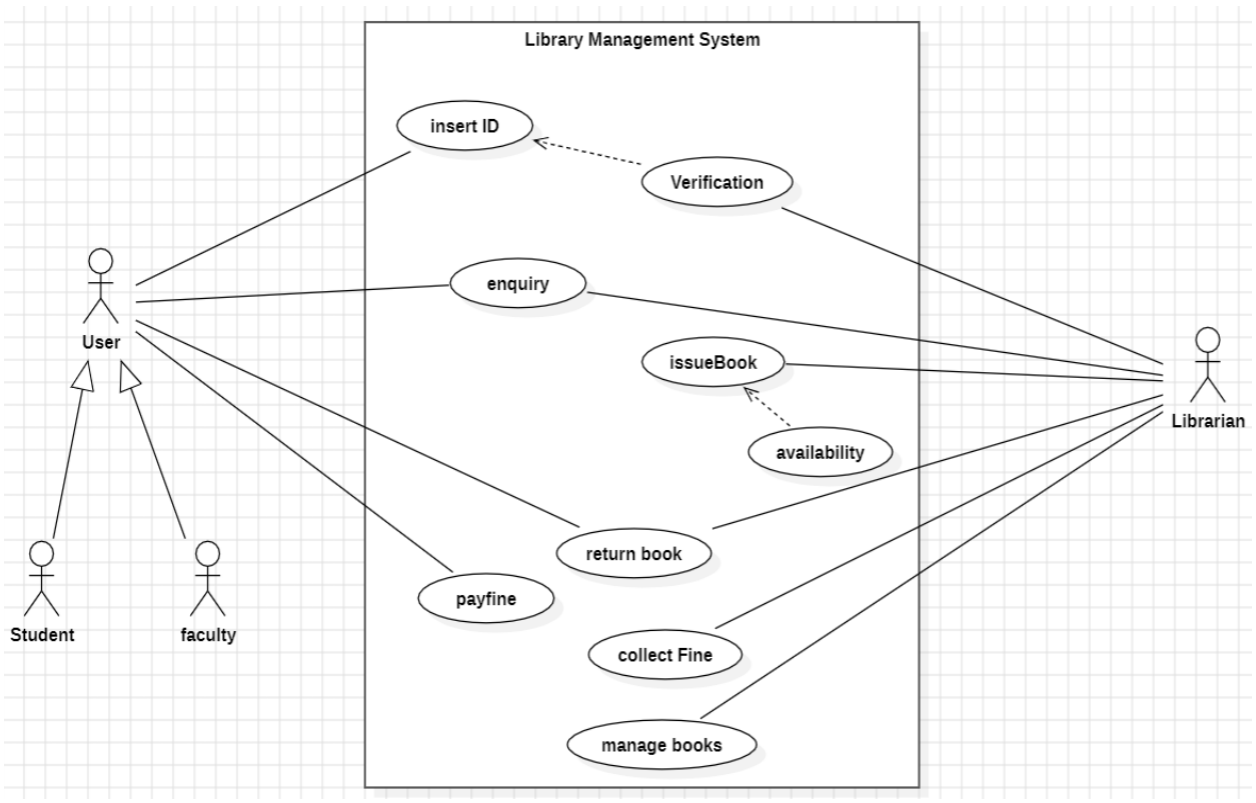


Associations:

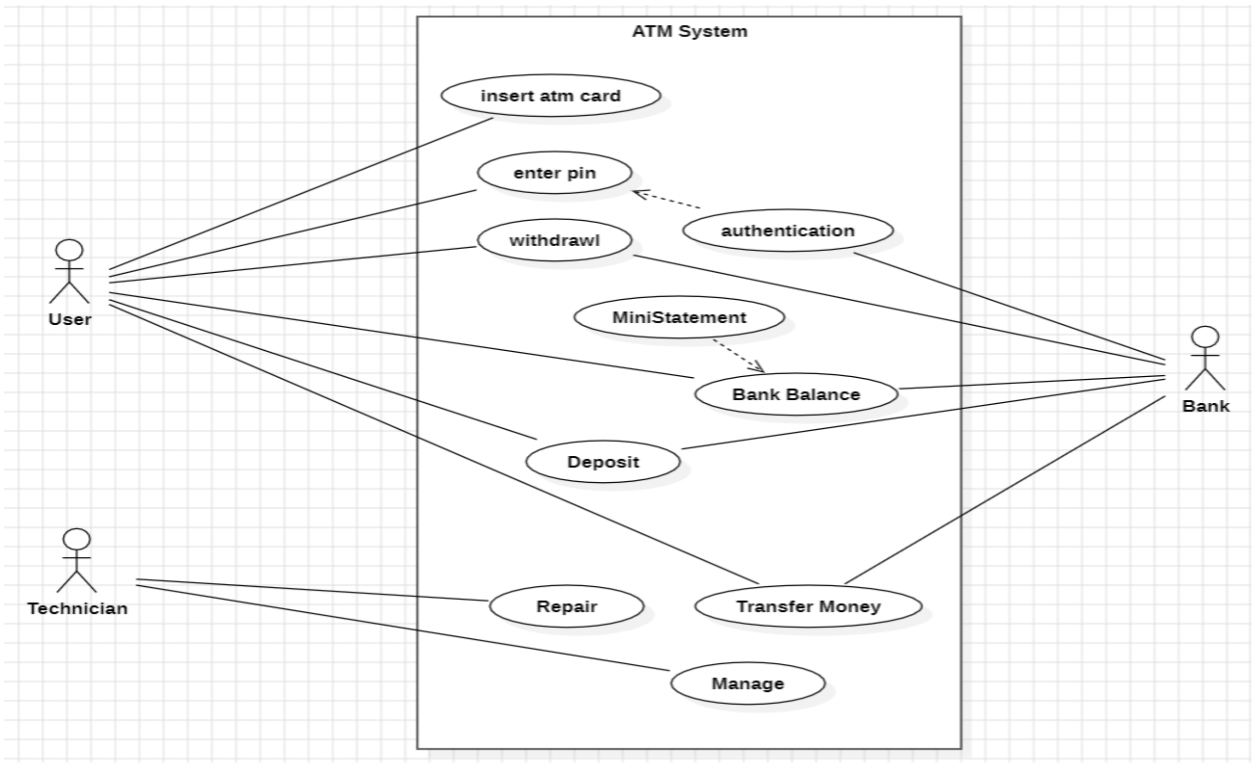
- Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line.



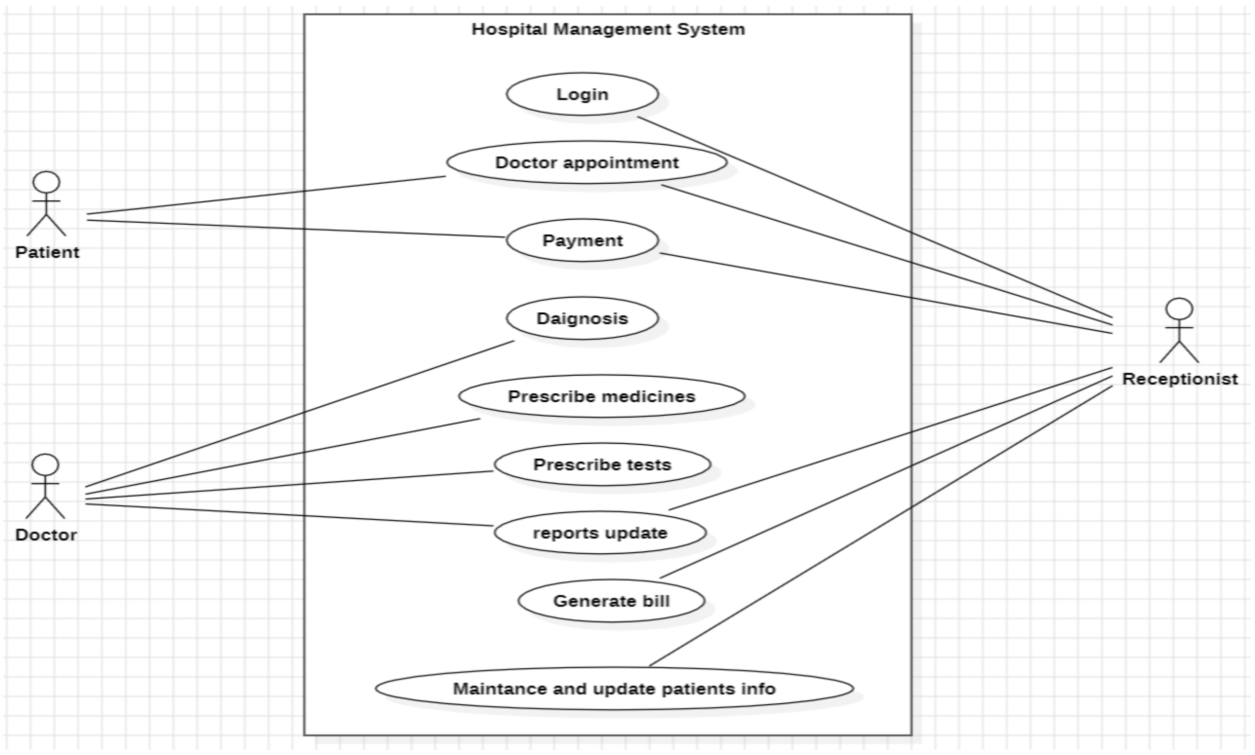
1)Library Management System



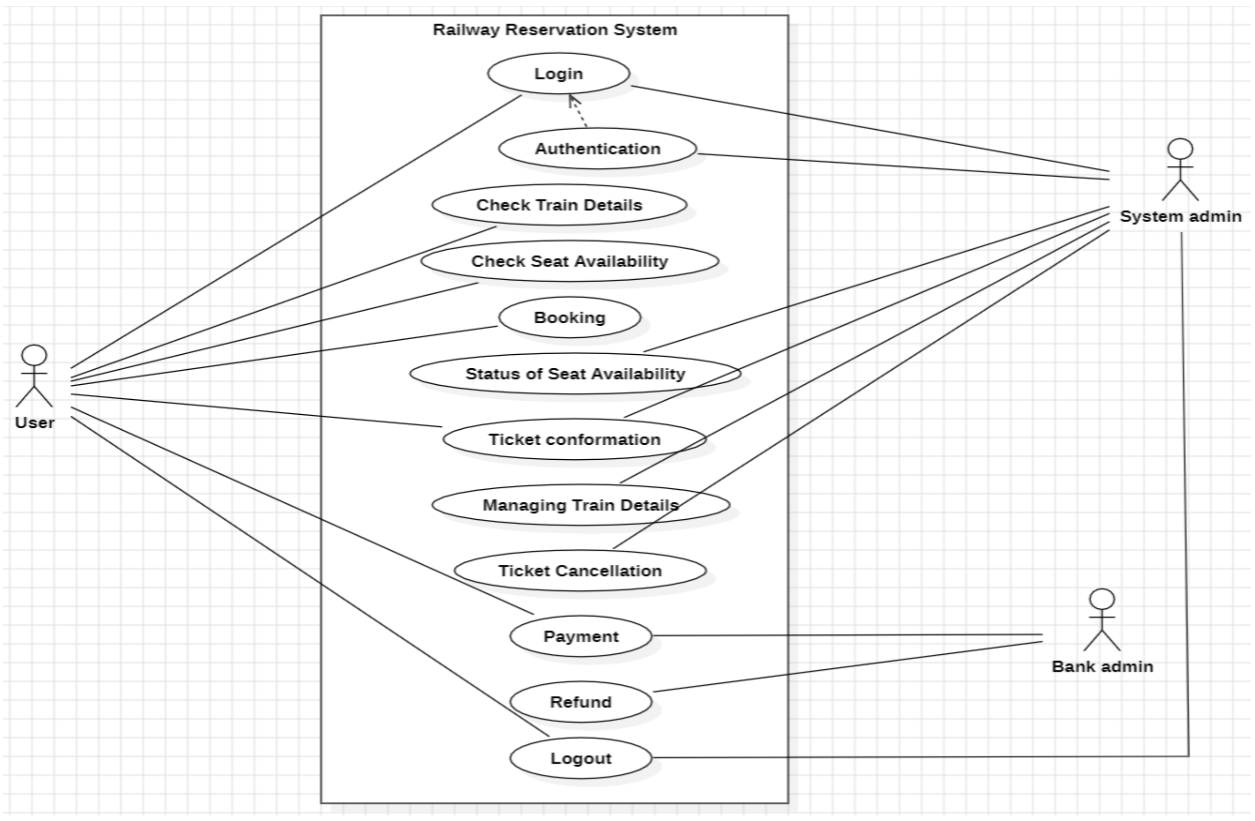
2)ATM Management System



3)Hospital Management System



4)Railway Reservation System



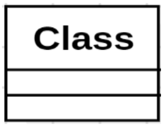
Task-2: Draw basic Class Diagram to identify and describe key concepts like classes, and their relationships.

Class diagram:

- ➢ Class diagram is a static structure diagram in software engineering,detailing a systems classes, their attributes, methods , and relationships, specifying how information is organized.

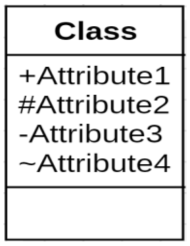
Classes:

- ➢ It is a rectangle divided into three sections: the top holds the class name, the middle lists attributes ,and the bottom details the class methods .



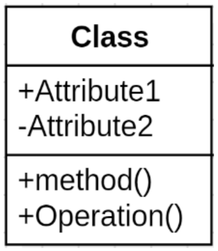
Attributes:

- ➢ Attributes in a class represent its properties or data, displayed as name: data type, e.g., "name: String" signifies an attribute named "name" with a data type of "String."



Methods:

- ➢ Methods in a class signify its operations, shown at the class's bottom with a name, parameters, and return type.

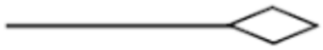


Associations:

- ➢ Associations between classes are shown as lines connecting the classes. The lines typically have arrowheads indicating the direction of the association.

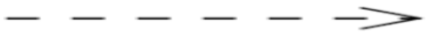
Aggregation and Composition:

- Aggregation (hollow diamond) and Composition (filled diamond) indicate part-whole relationships between classes in diagrams.



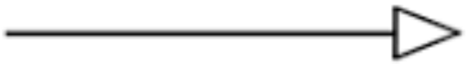
Dependencies:

- dependencies are indicated by a dashed line with an arrow, showing one class relying on another.

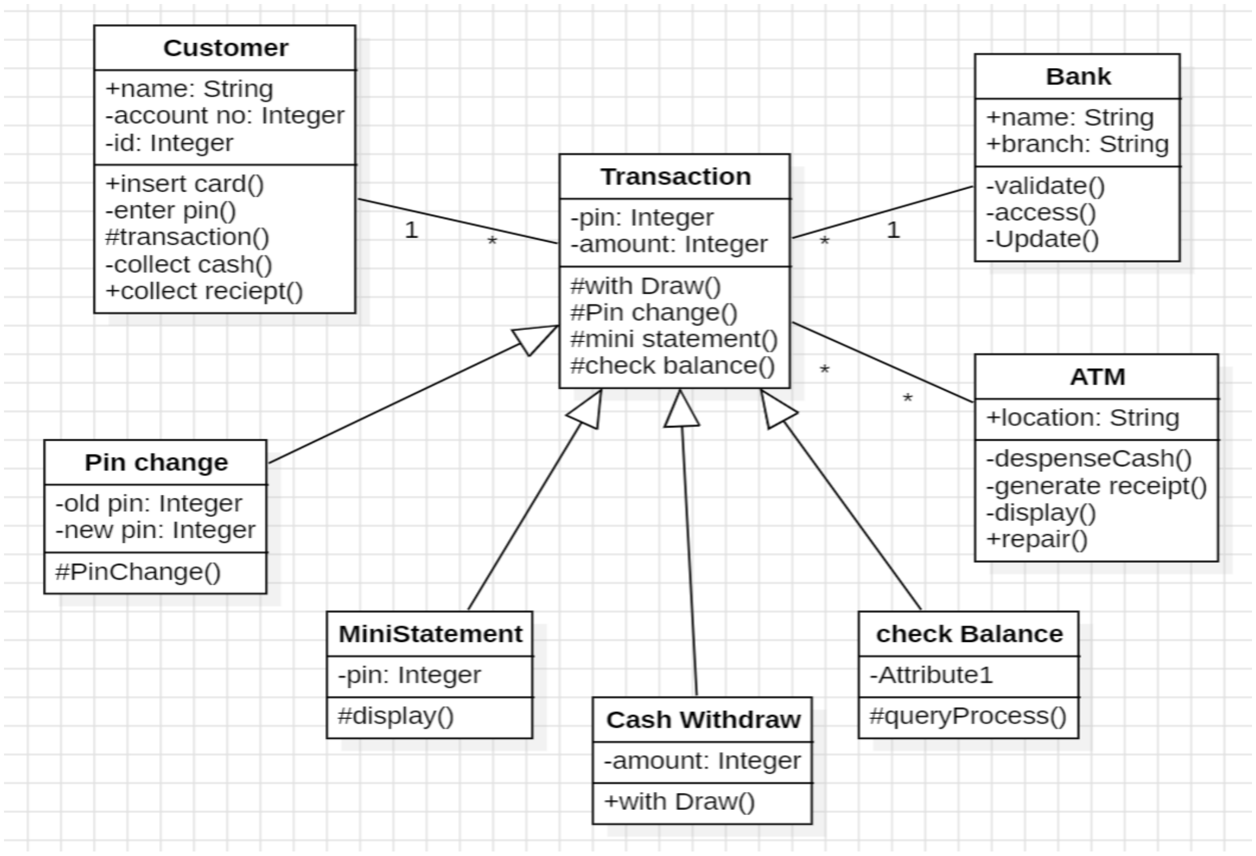


Inheritance:

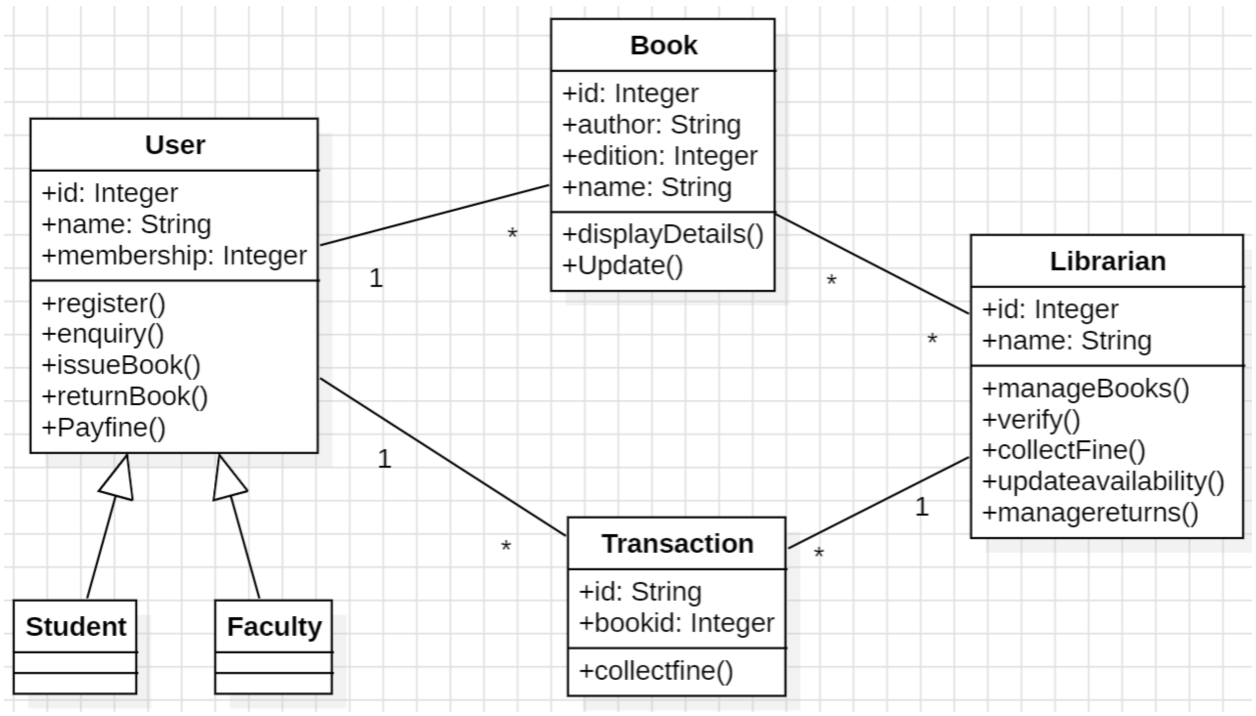
- Inheritance relationships are represented with a solid line with an open arrowhead from the subclass to the superclass.



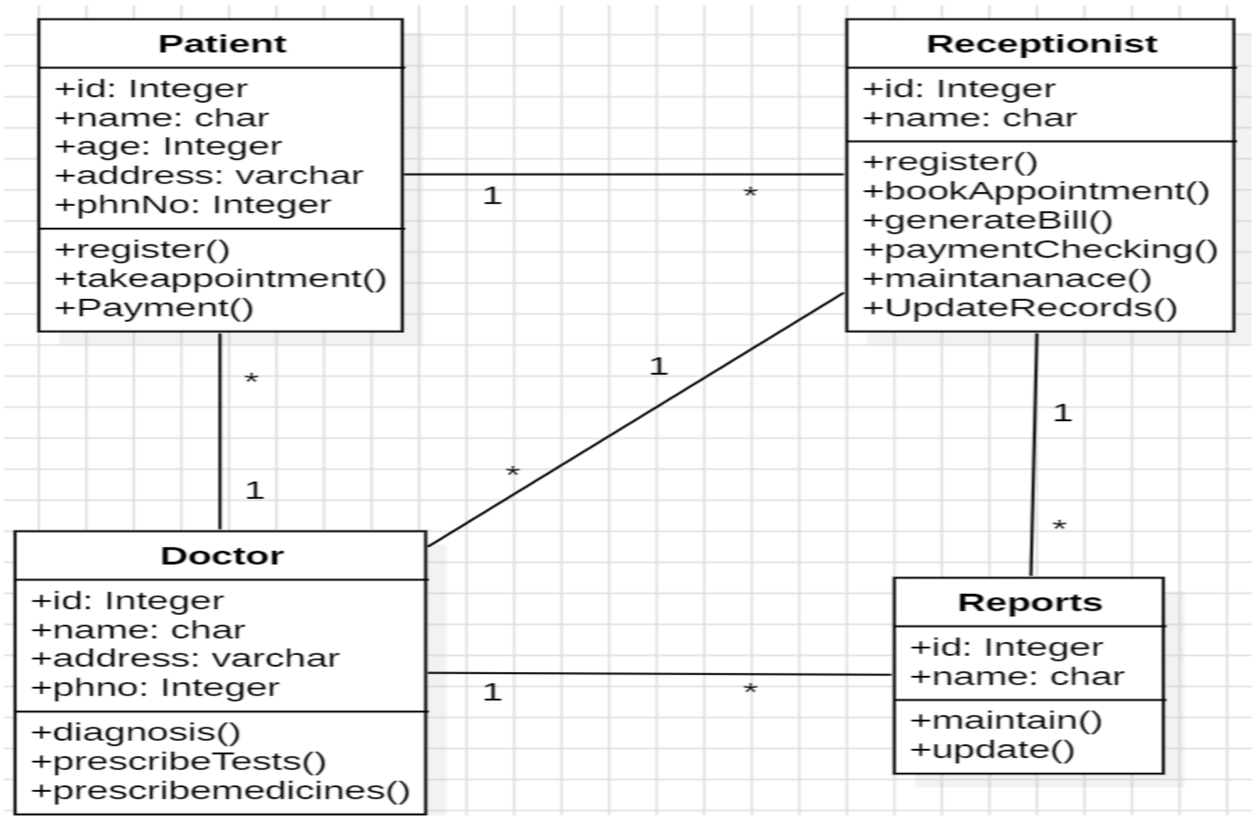
1)Library Management System



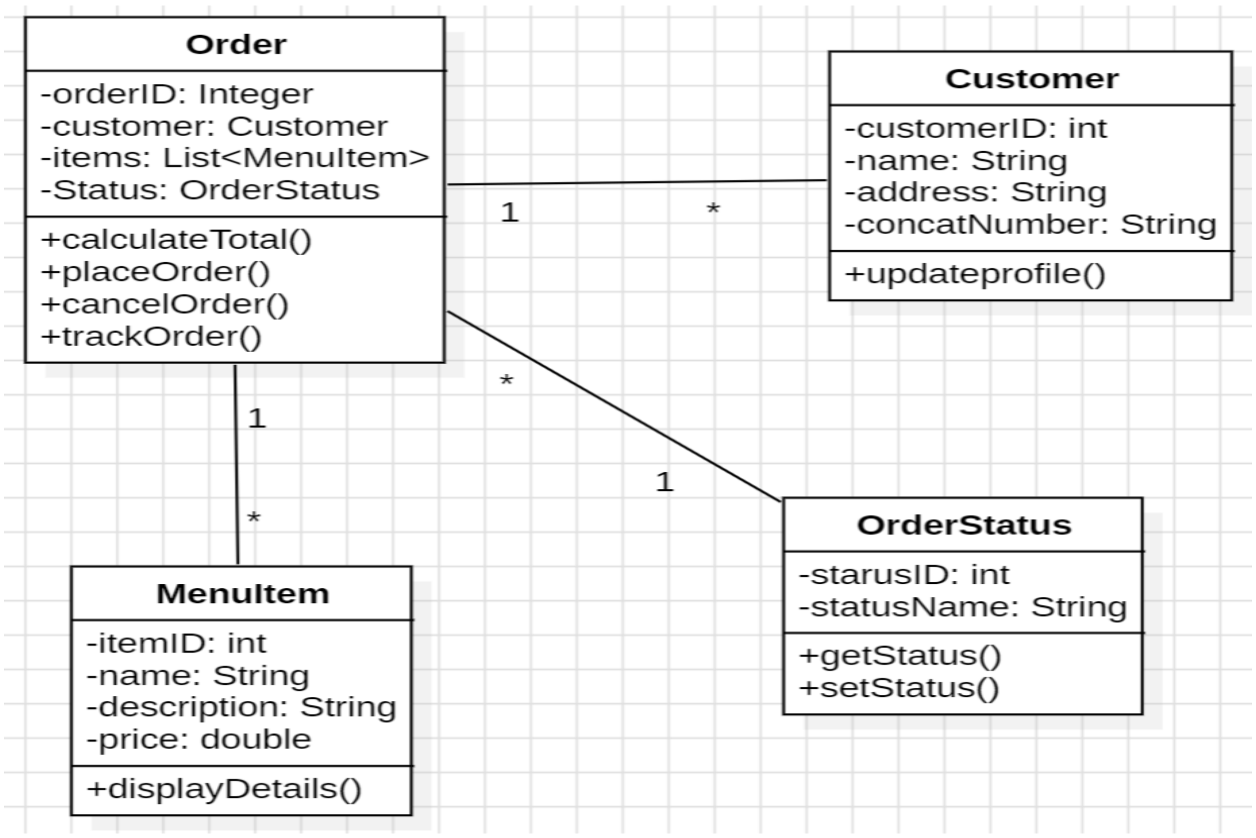
2)ATM Management System



3)Hospital Management System



4)Online Order Management System



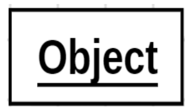
Task-3: Develop Object diagrams.

Object diagram:

- Object diagrams represent static snapshots of instances of the things found in class diagrams These diagrams address the static design view or static process view of a system
- An object diagram shows a set of objects and their relationships. they are often used to illustrate concrete examples of the system's structure and demonstrate how objects are connected.

Objects:

- Objects represent instances of classes in the system. They are depicted as rectangles with the object's name inside. each object in the diagram corresponds to an instance of a particular class.

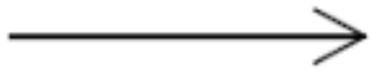


Links/Associations:

- Links or associations represent relationships between objects. These connections may indicate associations, aggregations, or compositions between the objects.

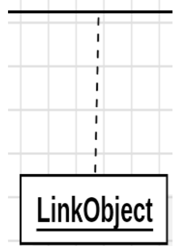
Directed Link:

- A Uml directed link indicates the association between two objects with arrow heads on the line, specifying the direction from the source object to the target object.



Link Object:

- A link object is an instance of an association between two cases. It's an intermediary object that captures information specific to the association.in uml, links objects can be used to represent additional details about the relationship.



Roles of object diagram:

- The role of an object in a particular relationship is often denoted near the link connecting the objects.
- Object diagrams may include attributes and values to provide additional information about the state of the objects at that specific point in time.

Task-4: Draw sequence diagrams and communication diagrams with advanced notation for system to show objects and their message exchanges.

Interaction Diagrams:

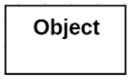
- Both sequence diagrams and collaboration diagrams are kinds of interaction diagrams Interaction diagrams address the dynamic view of a system.

sequence diagram:

- A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.
- Sequence diagrams are isomorphic, meaning that you can take one and transform it into the other.
- Sequence diagrams capture the flow of messages and interactions between objects or actors within a system, providing a visual representation of the dynamic behavior of the system.

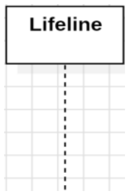
Objects:

- Represented by rectangles or named boxes at the top of the diagram, depicting entities involved in the interaction. Objects are labeled with their class or role names.



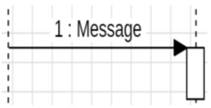
Lifelines:

- Vertical dashed lines extending from objects or actors, representing the timeline or lifespan of the object during the interaction.



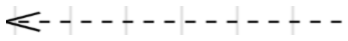
Message:

- Represents communication between objects, depicted by arrows indicating the flow of information.



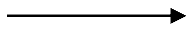
Reply Message:

- Indicates the response or return communication from a receiver to the sender.These are usually depicted with dashed arrows or lines.



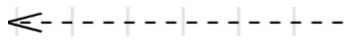
Synchronous Message:

- Represents a communication where the sender waits for a response before continuing. It is denoted by solid arrow.



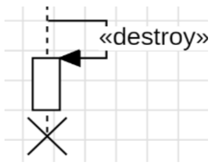
Asynchronous Message:

- Represents a communication where the sender continues without waiting for a response. It is denoted by dashed arrow.



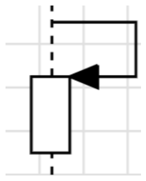
Destroy/Delete Message:

Indicates the termination or removal of an object from the interaction.



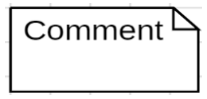
Reflexive Message:

- Represents a message sent from an object to itself. Represents a message sent from an object to itself.



Comment Line/Notes:

- Provides additional explanations or details within the diagram. Usually depicted as text boxes attached to specific elements.

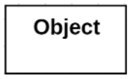


collaboration diagram:

- A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages.
- collaboration diagrams are isomorphic, meaning that you can take one and transform it into the other.
- Collaboration diagrams focus on the interactions between objects or roles to achieve a specific behavior or task, providing a visual representation of the communication flow within the system.

Objects:

- Represented by rectangles or named boxes at the top of the diagram, depicting entities involved in the interaction. Objects are labeled with their class or role names.



Links:

- Represented by arrows or lines between objects,indicating the communication or interaction between them.

Message:

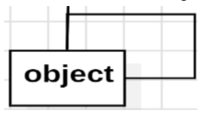
- Represented by arrows or lines between objects in a collaboration diagram.

Connector:

- Symbolizes a link between objects in a collaboration diagram.

Self Connector:

- Indicates a connection from an object to itself in a collaboration diagram.

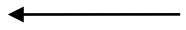


Forward Message:

- Represents a message sent from one object to another in a collaboration diagram.

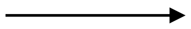
Reverse Message:

- Represents a message sent back from the receiver to the sender in a collaboration diagram.



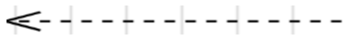
Synchronous Message:

- Indicates a message that requires a response before the sender can proceed.



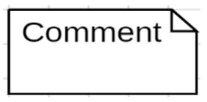
Asynchronous Message:

- Indicates a message that does not require an immediate response.



Comment Line or Notes:

- Provides additional explanations or details within the diagram.



Multiplicity:

- Indicates the number of instances of one object that can be associated with an instance of another object.

Task-5 : Develop State chart diagrams.

State chart diagram:

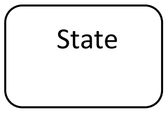
- A state chart diagram shows a state machine, consisting of states, transitions, events, and activities.
- State chart diagrams address the dynamic view of a system.
- They are especially important in modeling the behavior of an interface, class, or collaboration and emphasize the event-ordered behavior of an object.

State machine:

- State machine is behavior that specifies the sequences of states an object.
- State machine involves a number of other elements, including states, transitions, events and activities.
- Graphically, a state is rendered as a rounded rectangle, usually including its name and its substates.

States:

- Represented by rounded rectangles, depicting the various conditions or situations in which an object or system can exist.



Transitions:

- It defines the movement from one state to another and is triggered by events or conditions that occur while the system is in a particular state, represented by arrows between states in the diagram, with labels indicating the event or condition.

Events:

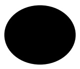
- Represented by triggers or conditions that cause a transition from one state to another, labeled on transition arrows.

Actions:

- Activities performed when entering, exiting, or within states, often associated with transitions.

Initial State:

- Denotes the starting point of the state machine, depicted by a filled circle.

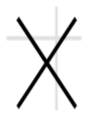


Initial Activity with States:

- The initial activity in a state chart diagram defines the actions when entering the initial state, depicted as a small black-filled circle attached to the initial state.

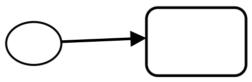
Exit or Termination:

- Exit or termination in a state chart diagram refers to the process of leaving a state and performing associated actions, depicted by a label on the transition arrow existing the state.



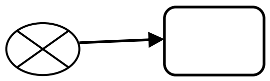
Activity Entry:

- Actions executed upon entering a state, depicted as a label on the transition arrow entering the state.



Activity Exit:

- Actions performed when leaving a state, depicted as a label on the transition arrow exiting the state.



Composition:

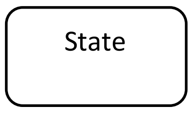
- Indicates a strong form of association where the composed object cannot exist without the composite object, depicted by a filled diamond.

Guard:

- Represents a condition that must be satisfied for a transition to occur, shown in square brackets next to the transition arrow.

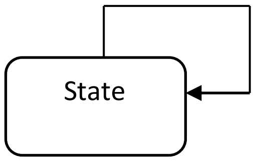
Simple State:

- Represents a basic state without nested states, depicted by a rounded rectangle.



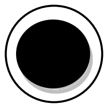
Self Transition:

- Represents a transition from a state back to the same state, depicted as an arrow looping back to the state.



Final State:

- Signifies the end of the state machine, shown as a circle with a dot inside.



Task- 6: Draw activity diagrams to display either business flows or like flow charts.

Activity diagram:

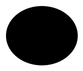
- An activity diagram is a special kind of a statechart diagram that shows the flow from activity to activity within a system
- Activity diagrams address the dynamic view of a system
- They are especially important in modeling the function of a system and emphasize the flow of control among objects.

Components:

- **Activity:** Represents a unit of work or action in a process.
- **Association:** Represents a relationship between elements, shown as a line connecting them.
- **Conflicts:** Indicate incompatible actions or states that cannot occur simultaneously.
- **Destination:** Represents the target or endpoint of a transition or flow in a diagram.

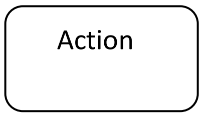
Initial Node:

- Represents the starting point of a process, often denoted by a filled circle, indicating where a process begins.



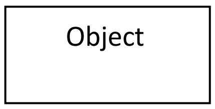
Activity or Action:

- Represents a specific task or behavior in a process, typically depicted as a rounded rectangle with the action's name inside.



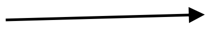
Object:

- Represents an instance of a class or entity within a system, shown as a rectangle with the object's name inside.



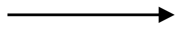
Edges or Control Flow:

- Arrows indicating the sequence of actions or transitions between nodes in a diagram, illustrating the flow of control.



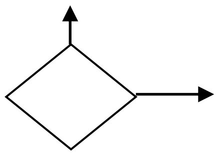
Object Control Flow:

- Arrows illustrating the flow of control between objects in a system, indicating interactions or messages sent between objects.



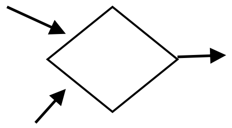
Decision Node:

- Represents a decision point in a process flow, where different paths or actions may be taken based on certain conditions, typically depicted as a diamond shape.



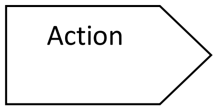
Merge/Join:

- Indicates a point in the process flow where multiple paths converge back into a single path, often shown as a bar merging multiple arrows.



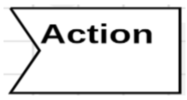
Signal Sending:

- Represents the act of sending a signal from one part of a system to another, shown as an arrow with a label indicating the signal.



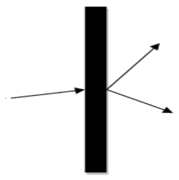
Signal Receiving:

- Represents the act of receiving a signal in a system, shown as an arrow with a label indicating the signal.



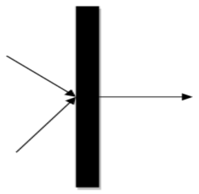
Fork:

- Represents a point in the process flow where a single path splits into multiple parallel paths, shown as a line splitting into multiple arrows.



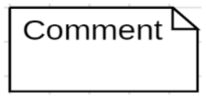
Join:

- Represents a point in the process flow where multiple parallel paths converge back into a single path, shown as multiple arrows merging into a single line.



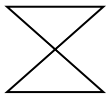
Comment:

- Provides additional information or explanations within a diagram, typically shown as a note attached to a specific element.



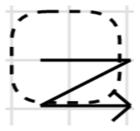
Time/Event:

- Represents a specific time or event trigger within a process flow, indicating when a particular action should occur.



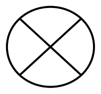
Interrupt:

- Represents an interruption or disruption in the normal flow of a process, often depicted as a lightning bolt.



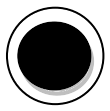
Control Flow End:

- Represents the end of a process flow or sequence of actions, typically shown as a circle.



Final Activity:

- Represents the final action or state in a process flow, indicating the completion or termination of the process.



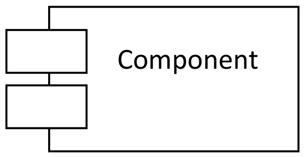
Task-7: Draw component diagrams assuming that build the system reusing existing components along with a few new ones.

Component diagram:

- A component diagram shows the organizations and dependencies among a set of components.
- Component diagrams address the static implementation view of a system.
- They are related to class diagrams in that a component typically maps to one or more classes, interfaces, or collaborations.

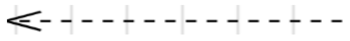
Component:

- Represents a modular, deployable, and replaceable part of a system, denoted by a rectangle with two smaller rectangles protruding from its top.



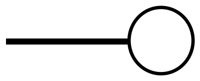
Dependency:

- Represents a relationship where one component depends on another, denoted by a dashed arrow from the dependent component to the independent component.



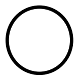
Interface Realization:

- Represents the implementation of an interface by a component, denoted by a solid line with a hollow triangle at the interface end and an arrow at the component end.



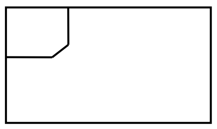
Interface:

- Represents a collection of operations that define a service or behavior, denoted by a circle labeled with the interface name.



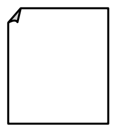
Frame:

- Represents a logical boundary within which a group of components can be placed, denoted by a rectangle with a dashed border.



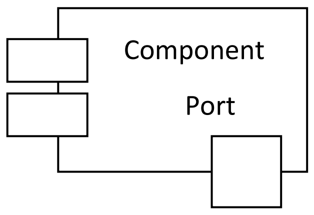
Artifact:

- Represents a physical piece of information that is used or produced by a software development process, denoted by a rectangle with a folded corner.



Port:

- Represents a point of interaction between a component and its environment, denoted by a small square on the edge of the component.



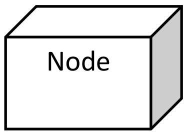
Task-8: Draw deployment diagrams to model the runtime architecture of system.

Deployment diagram:

- A deployment diagram shows the configuration of run-time processing nodes and the components that live on them.
- Deployment diagrams address the static deployment view of an architecture.

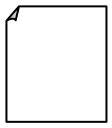
Node:

- Represents a physical entity within the system's environment, such as a server, PC, or mobile device, denoted by a box with the node's name inside.



Artifact:

- Represents a physical piece of information that is used or produced by a software development process, denoted by a rectangle with a folded corner.

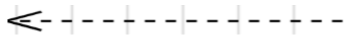


Association:

- Represents a relationship between a deployment artifact and a node, indicating that the artifact is deployed on that node, denoted by a solid line connecting the artifact to the node.

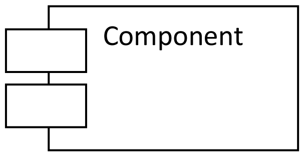
Dependency:

- Represents a relationship where one node depends on another, denoted by a dashed arrow from the dependent node to the independent node.



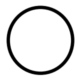
Component:

- Represents a modular, deployable, and replaceable part of a system, denoted by a rectangle with two smaller rectangles protruding from its top.



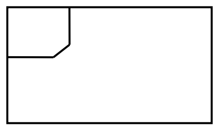
Interface:

- Represents a collection of operations that define a service or behavior, denoted by a circle labeled with the interface name.



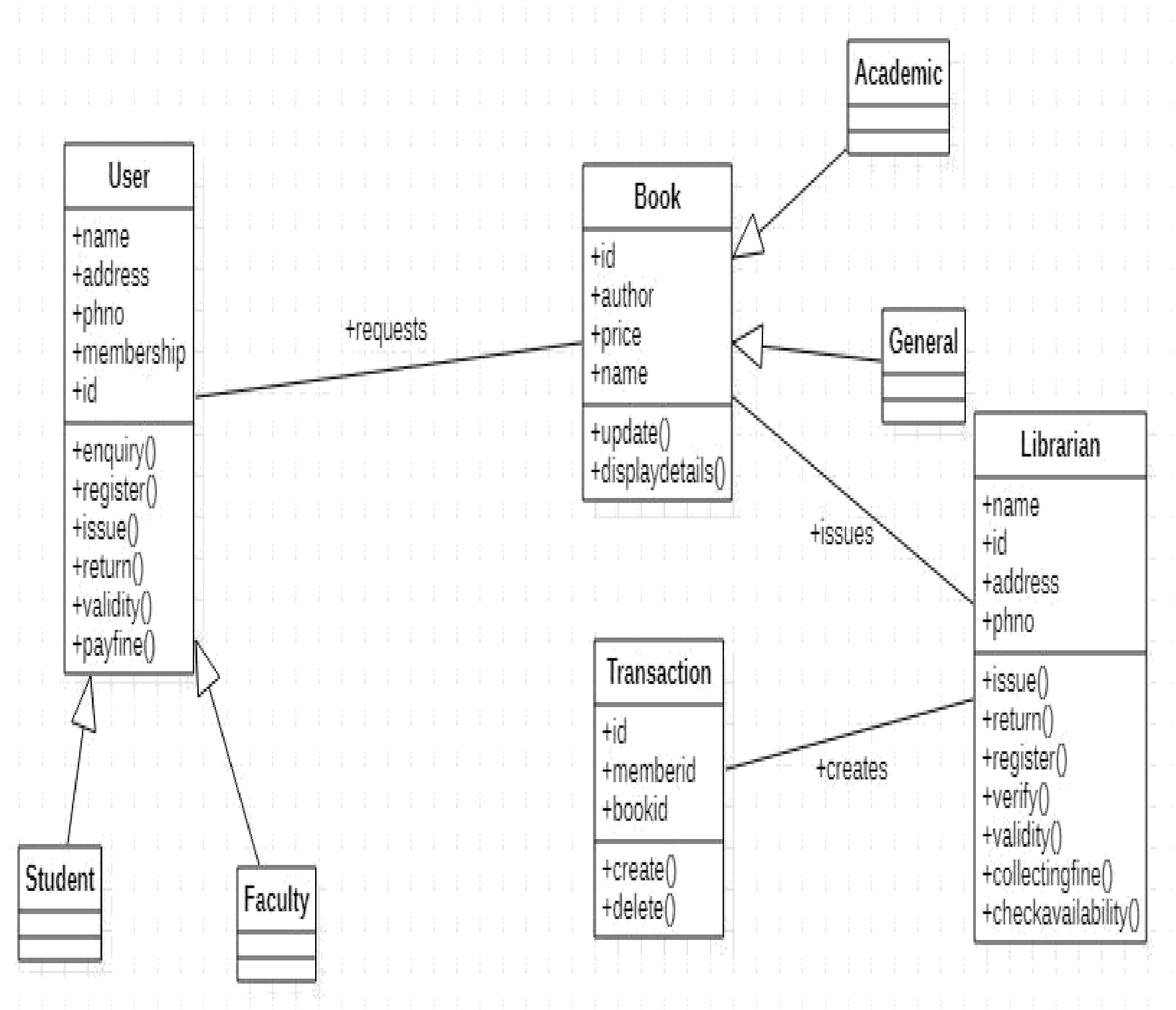
Frame:

- Represents a grouping of nodes or artifacts within a deployment diagram, denoted by a rectangle with a dashed border.

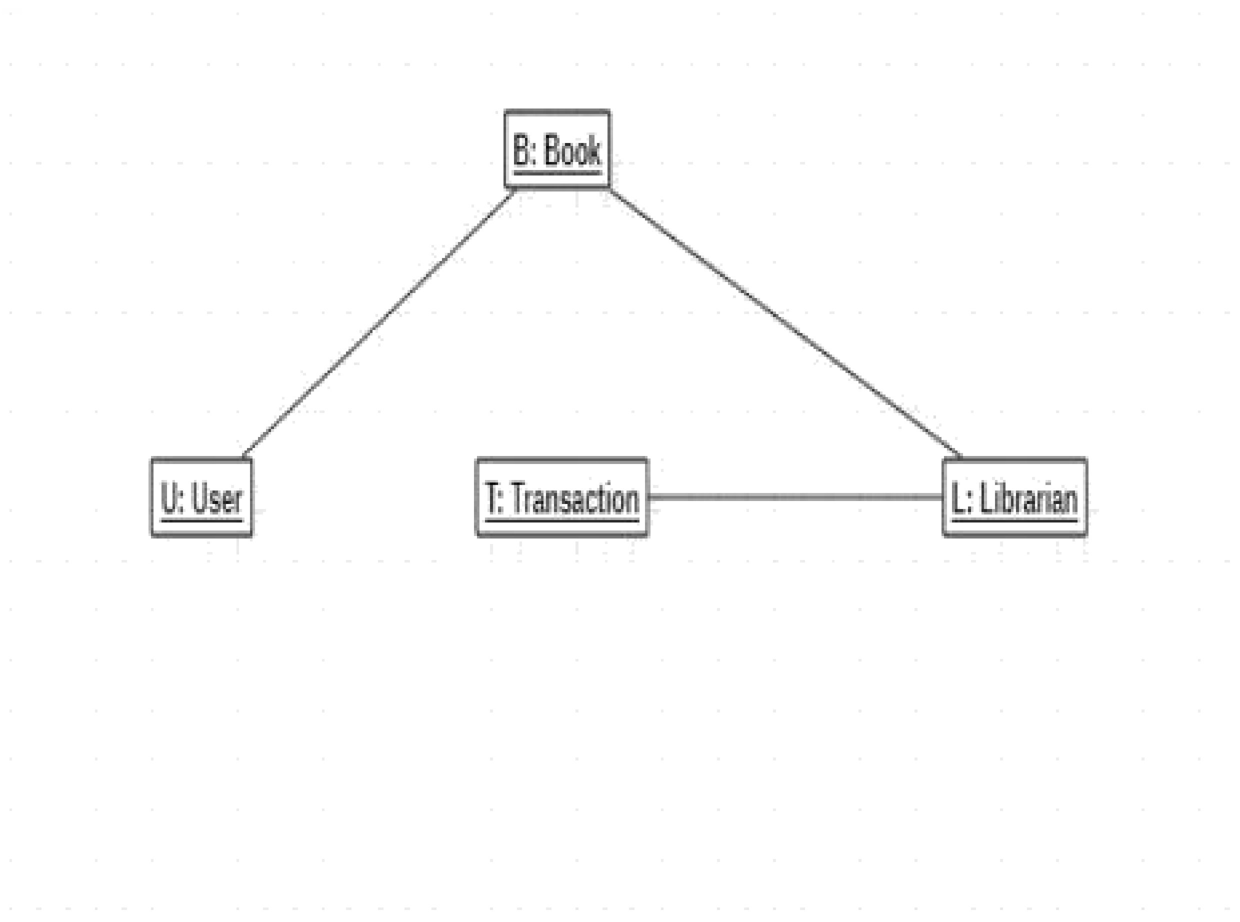


Task-9 Case study for Library Management System.

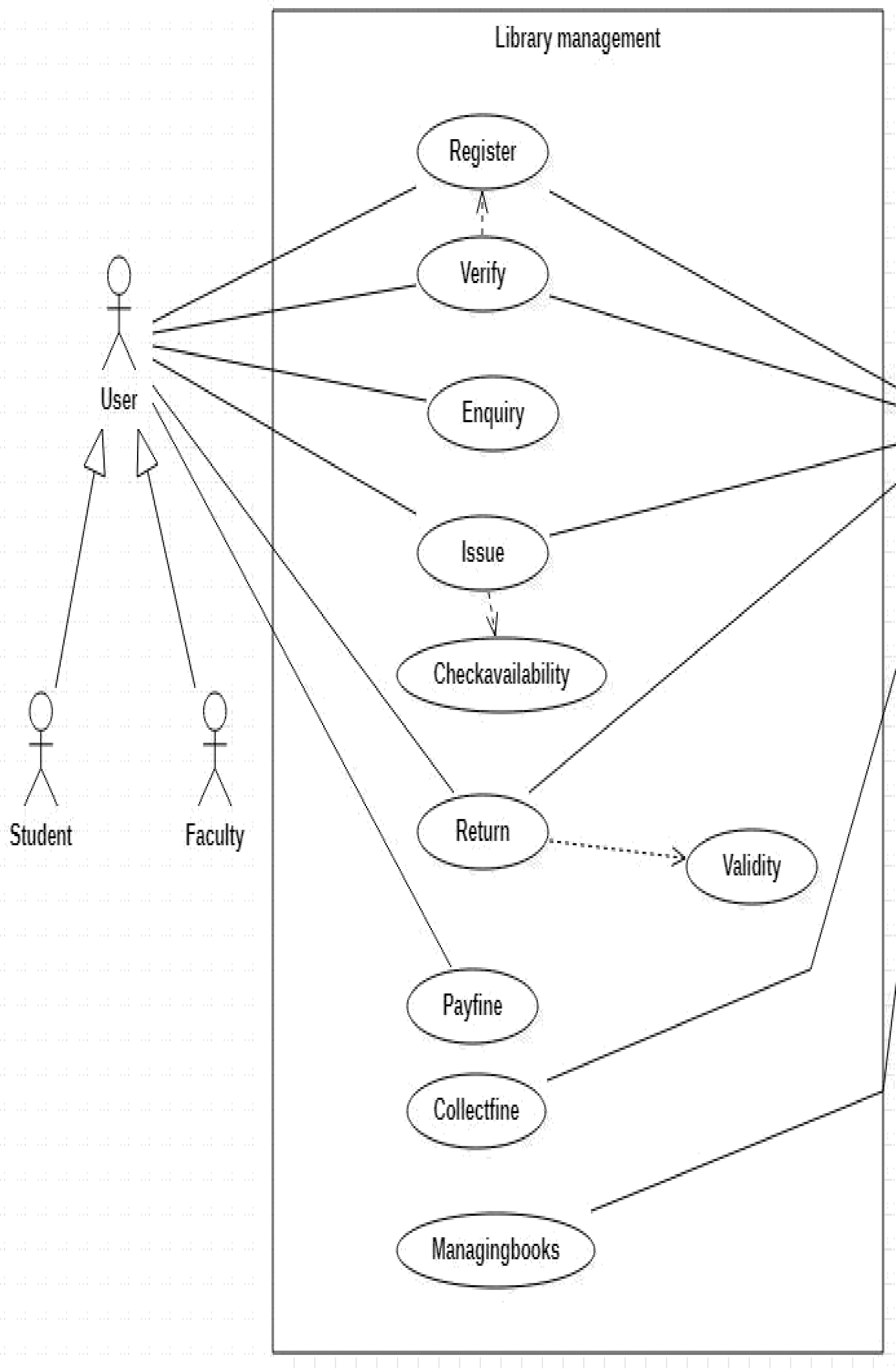
9.1Class Diagram Library Management

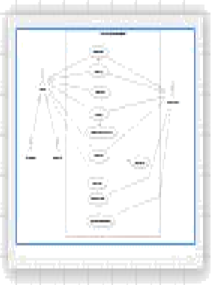


Object Diagram Library Management

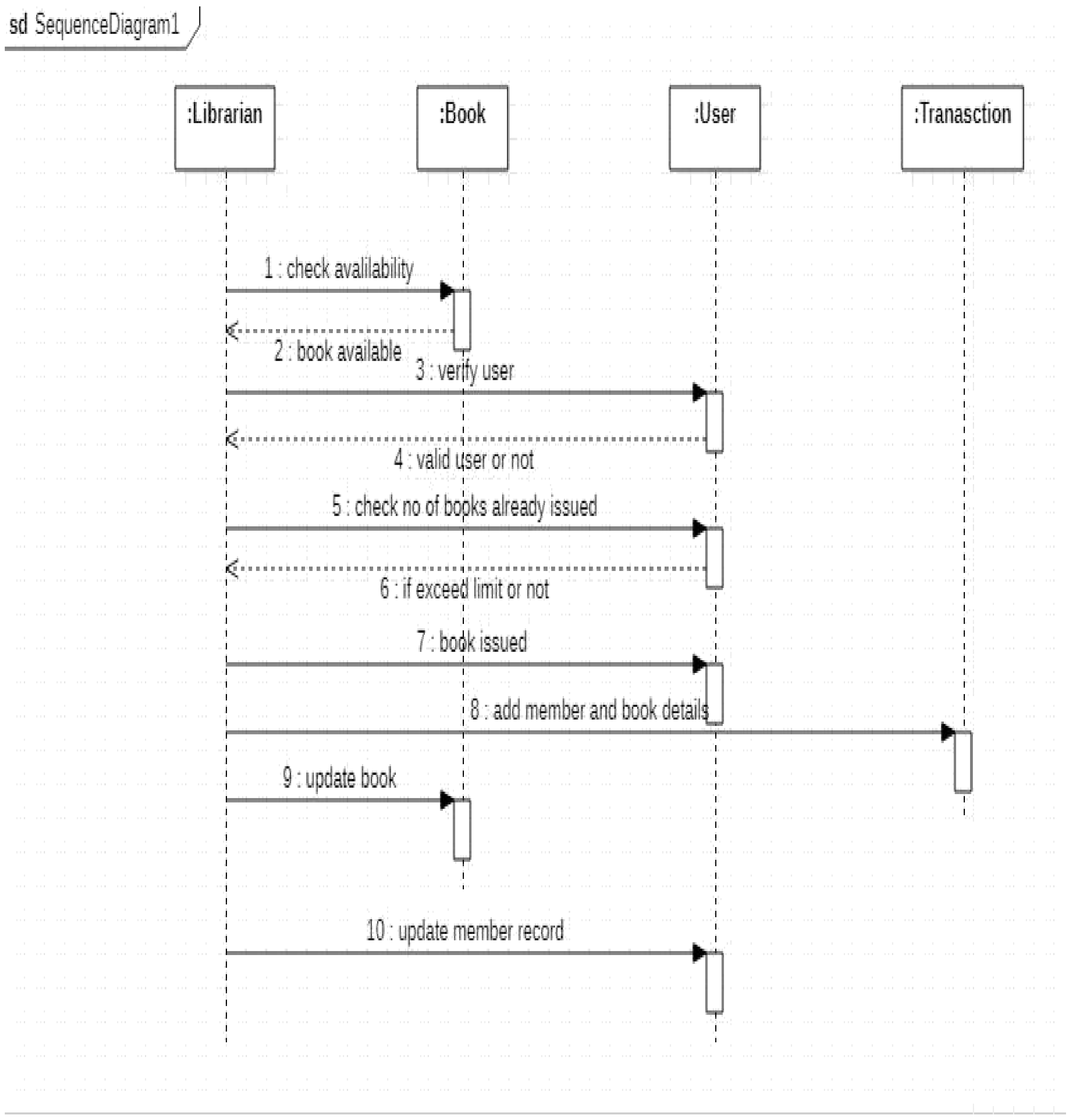


9.2 Use Case For Library Management

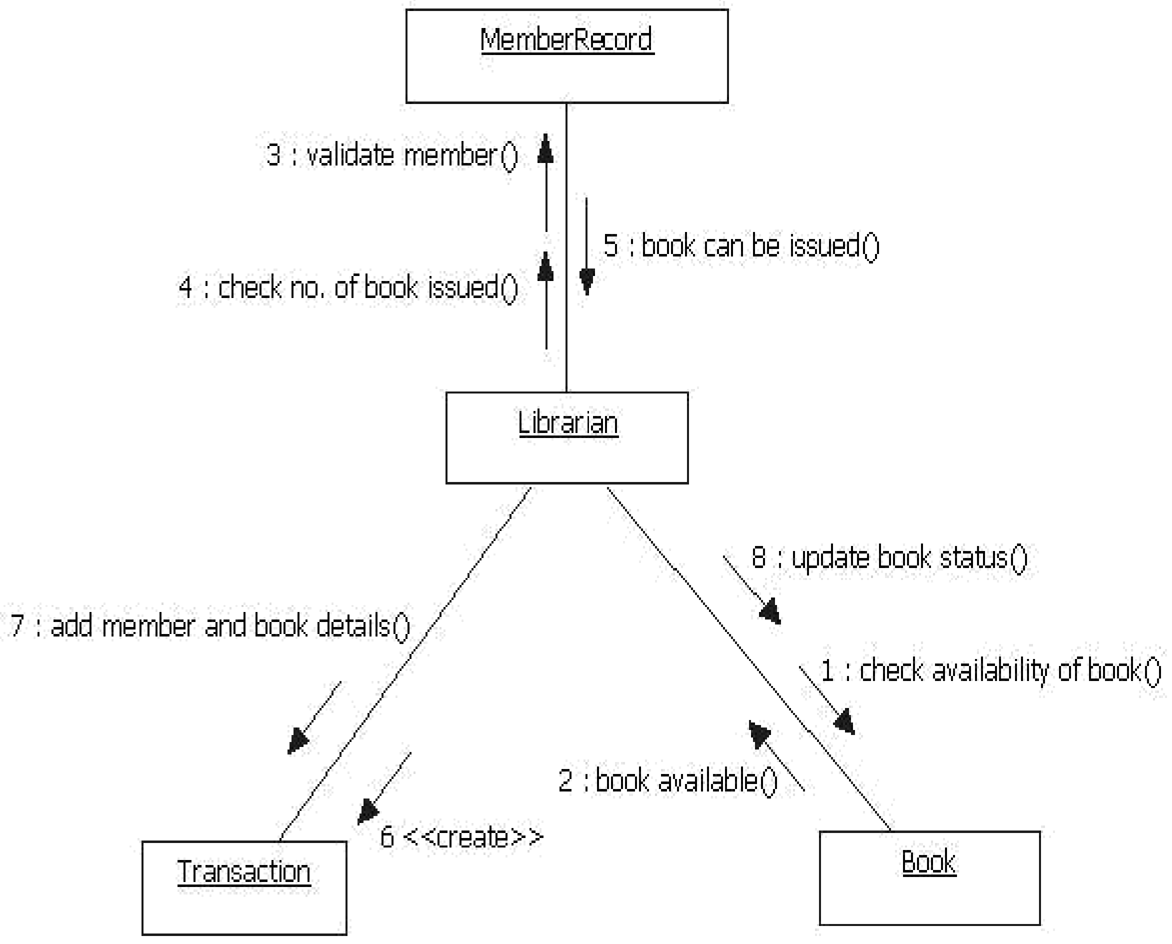




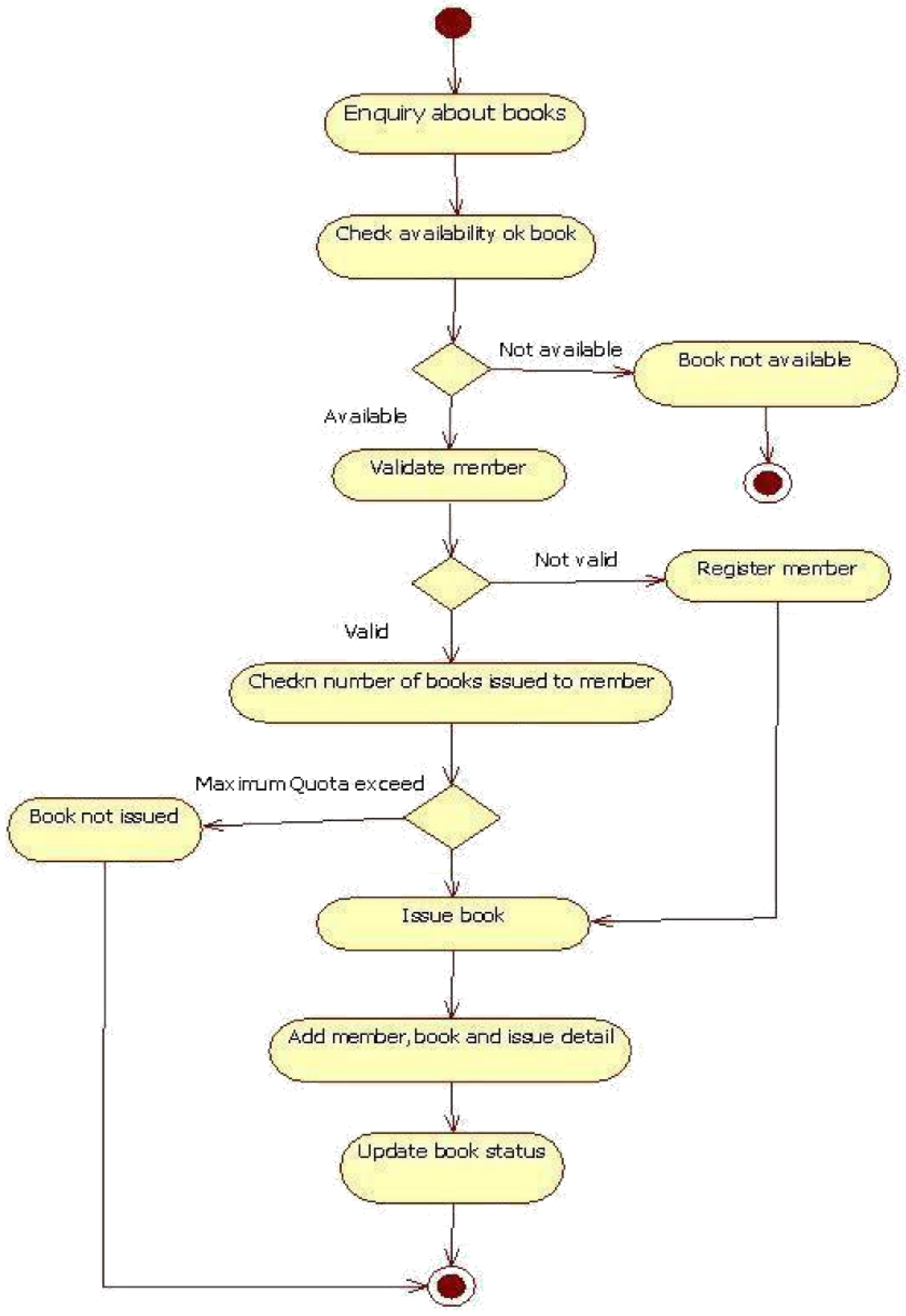
9.3 Sequence Diagram For Library Management



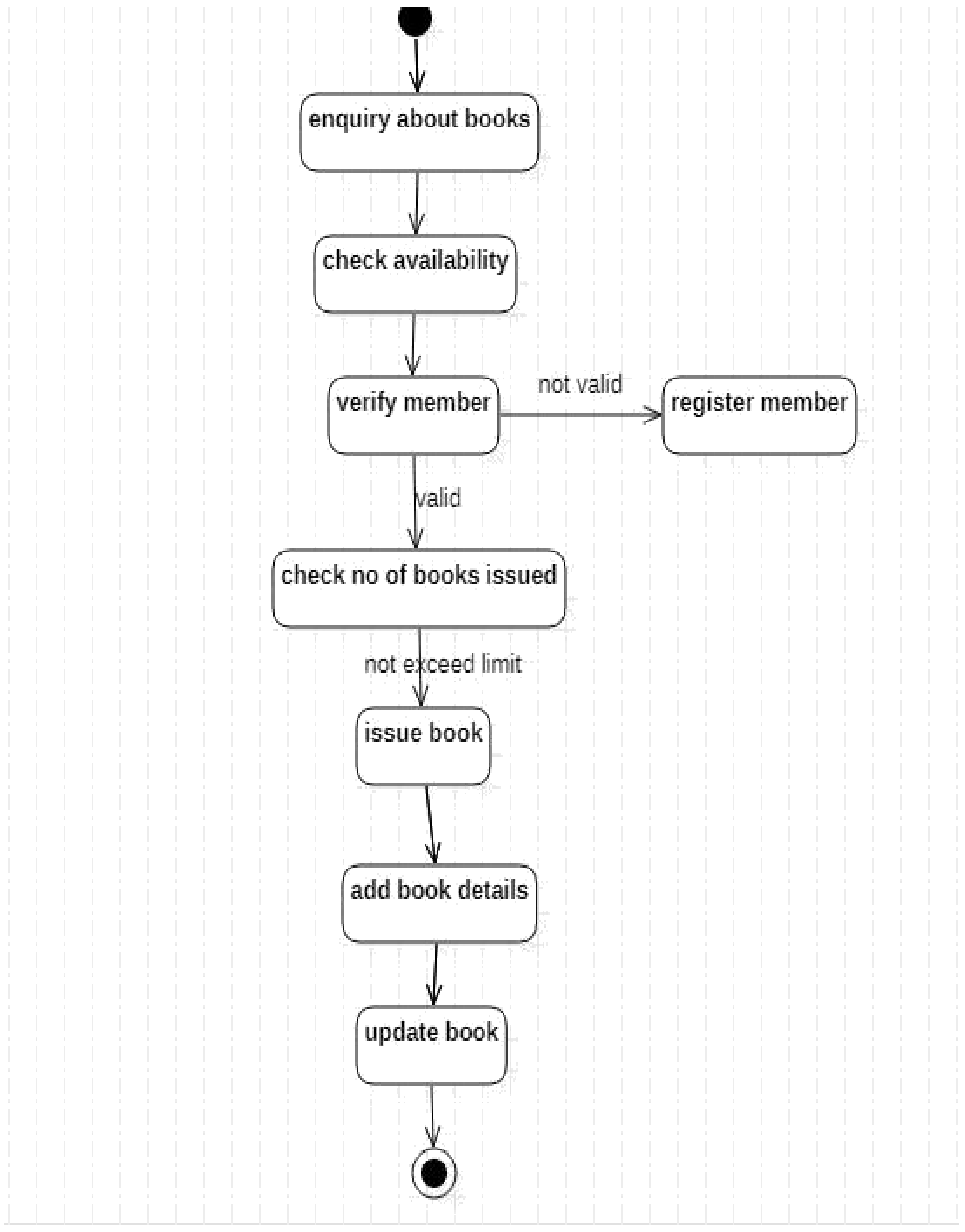
Collaboration Diagram For Library



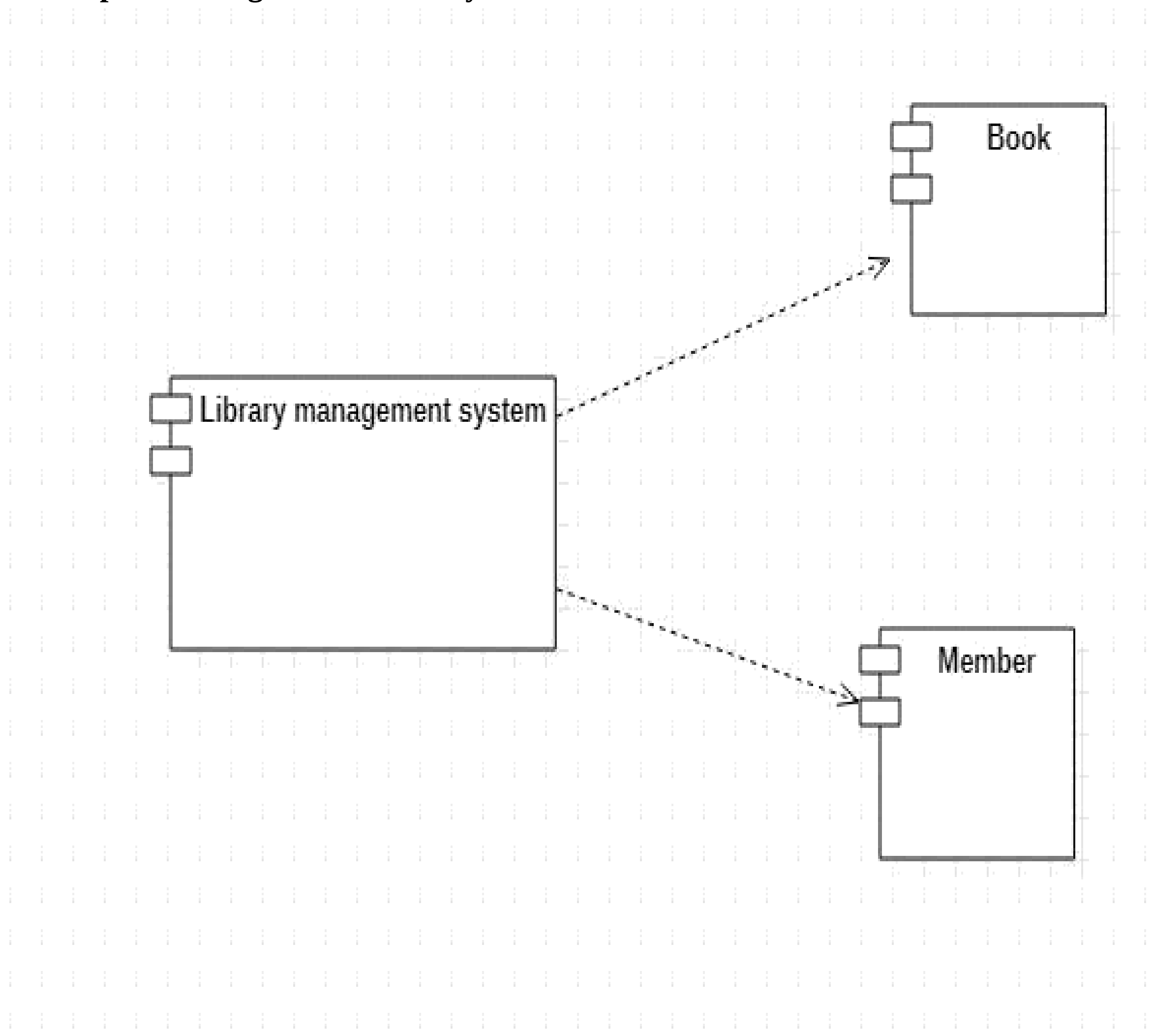
9.4Activity Diagram Library Management



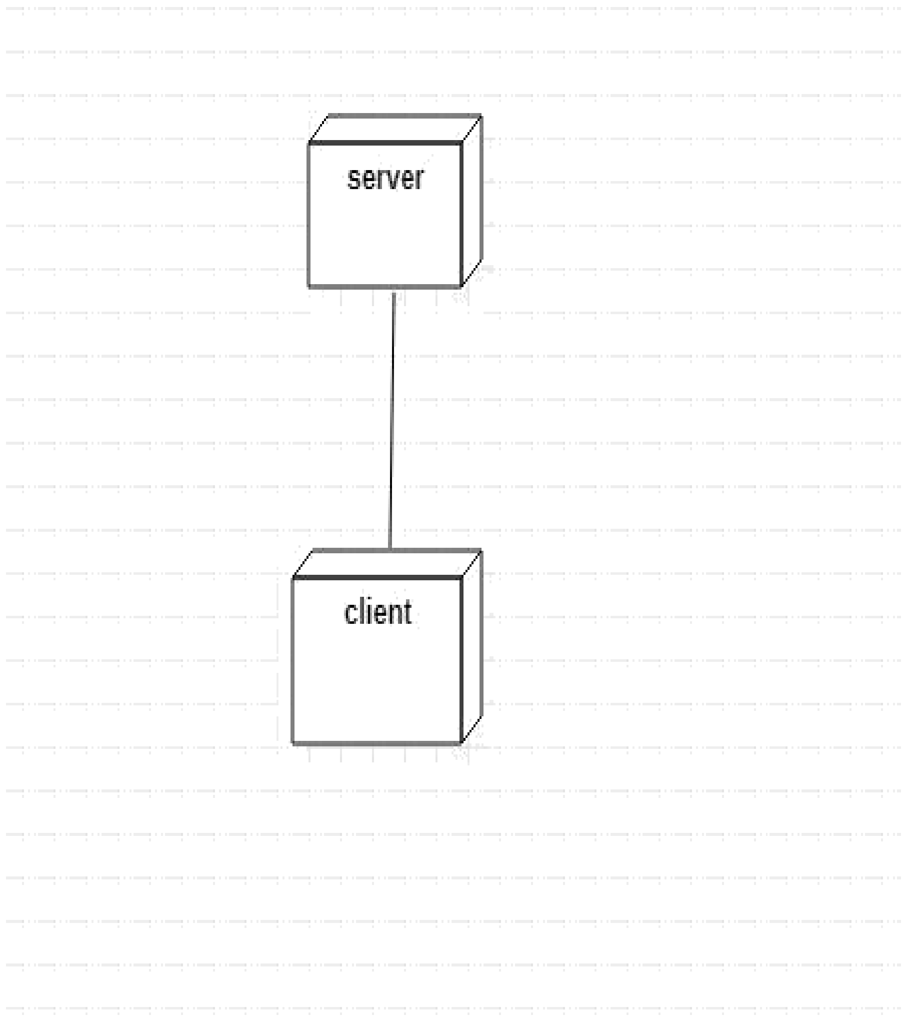
9.5 State chart Diagram For Library



9.6 Component Diagram for Library

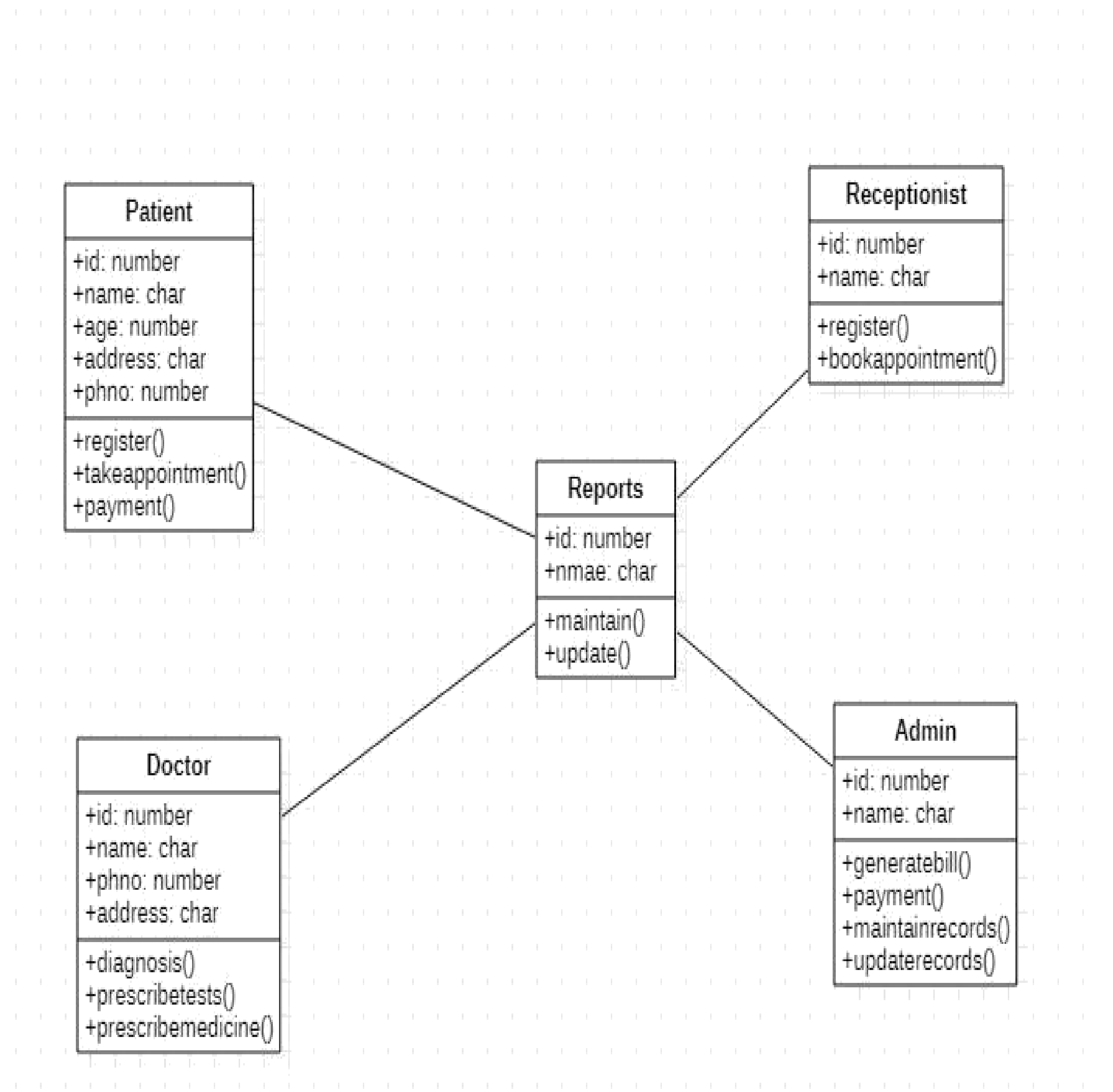


9.7 Deployment Diagram For Library

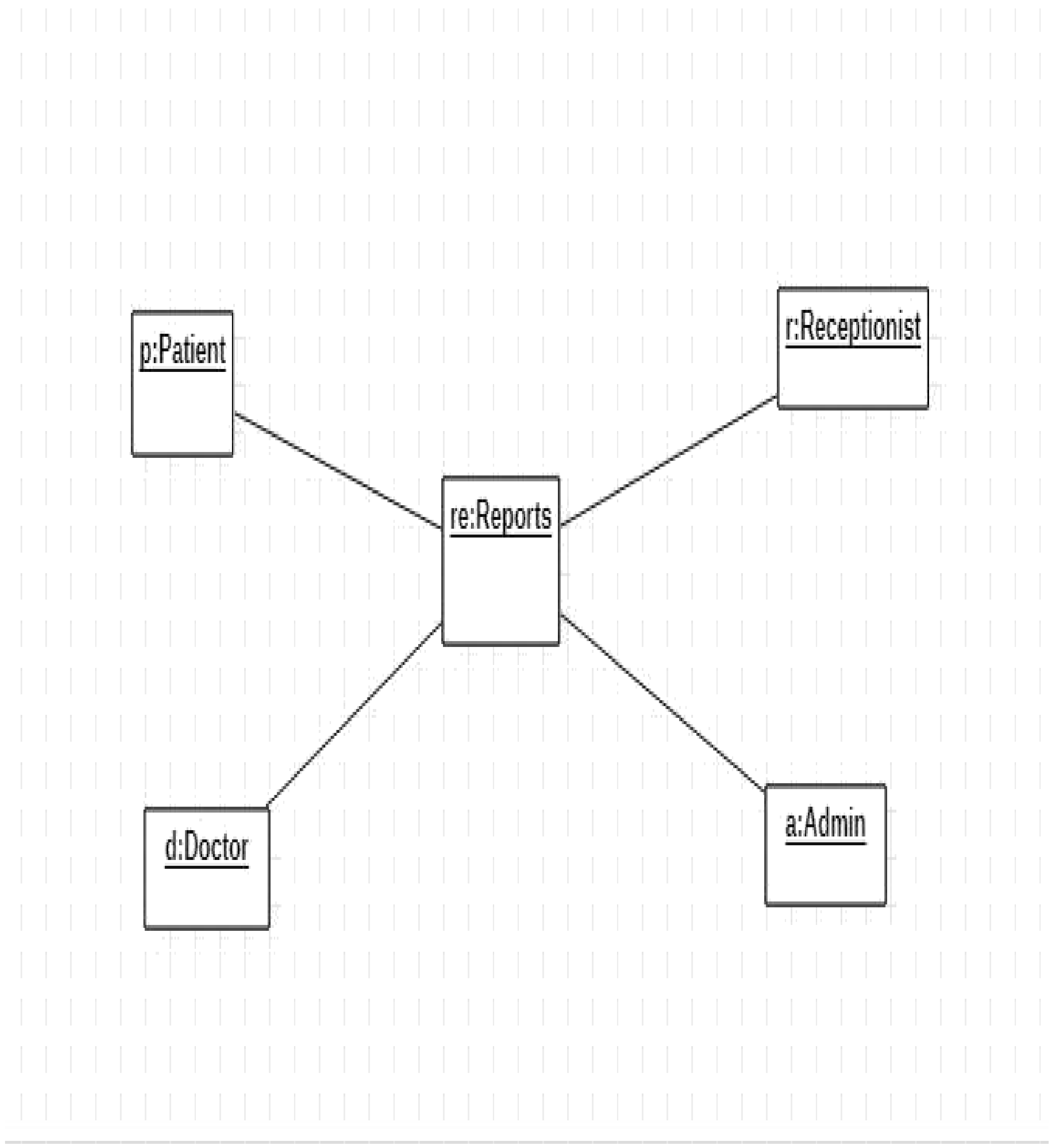


Task-10 Case study for Hospital Management System.

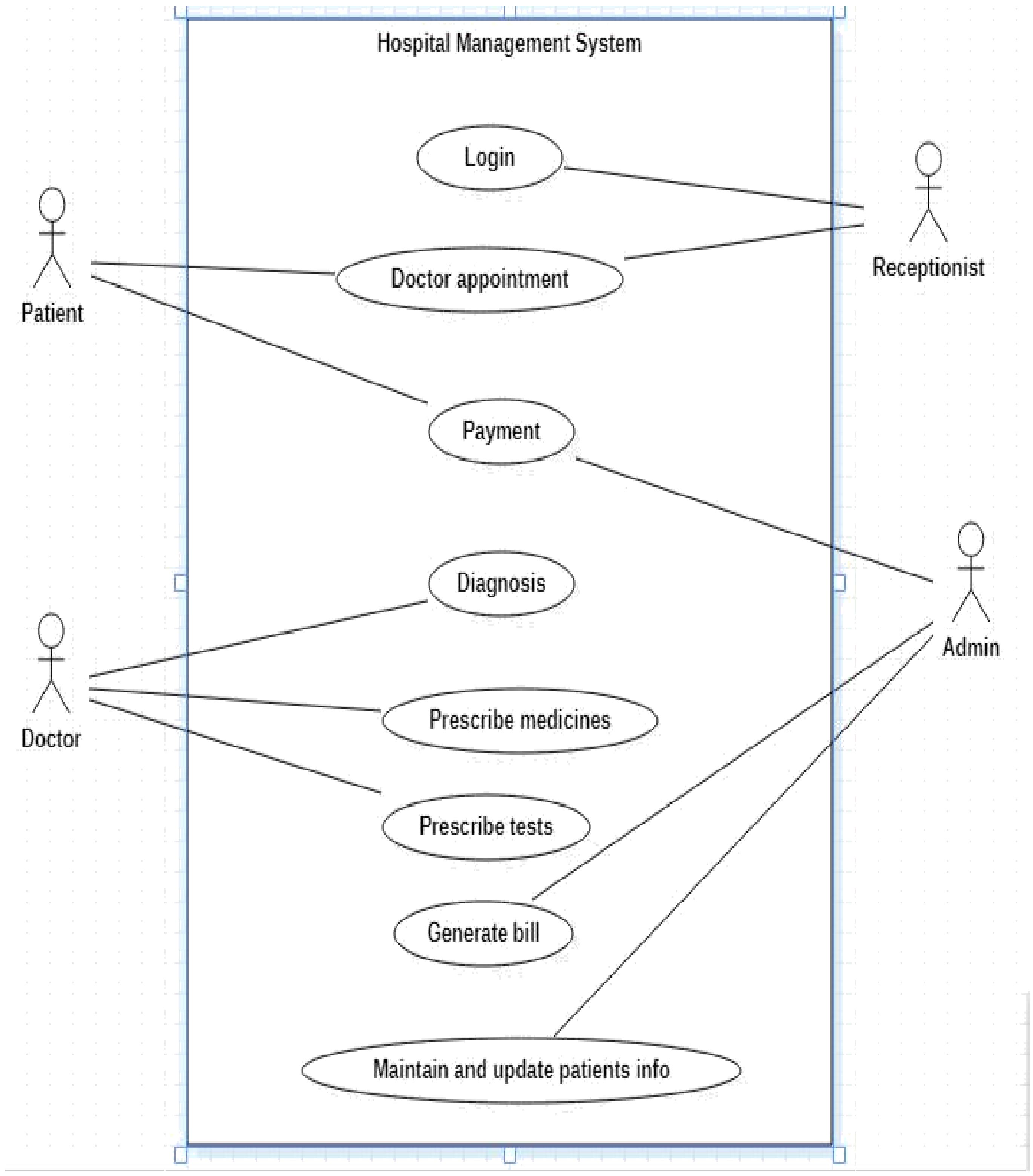
10.1.Class Diagram for Hospital Mgmt



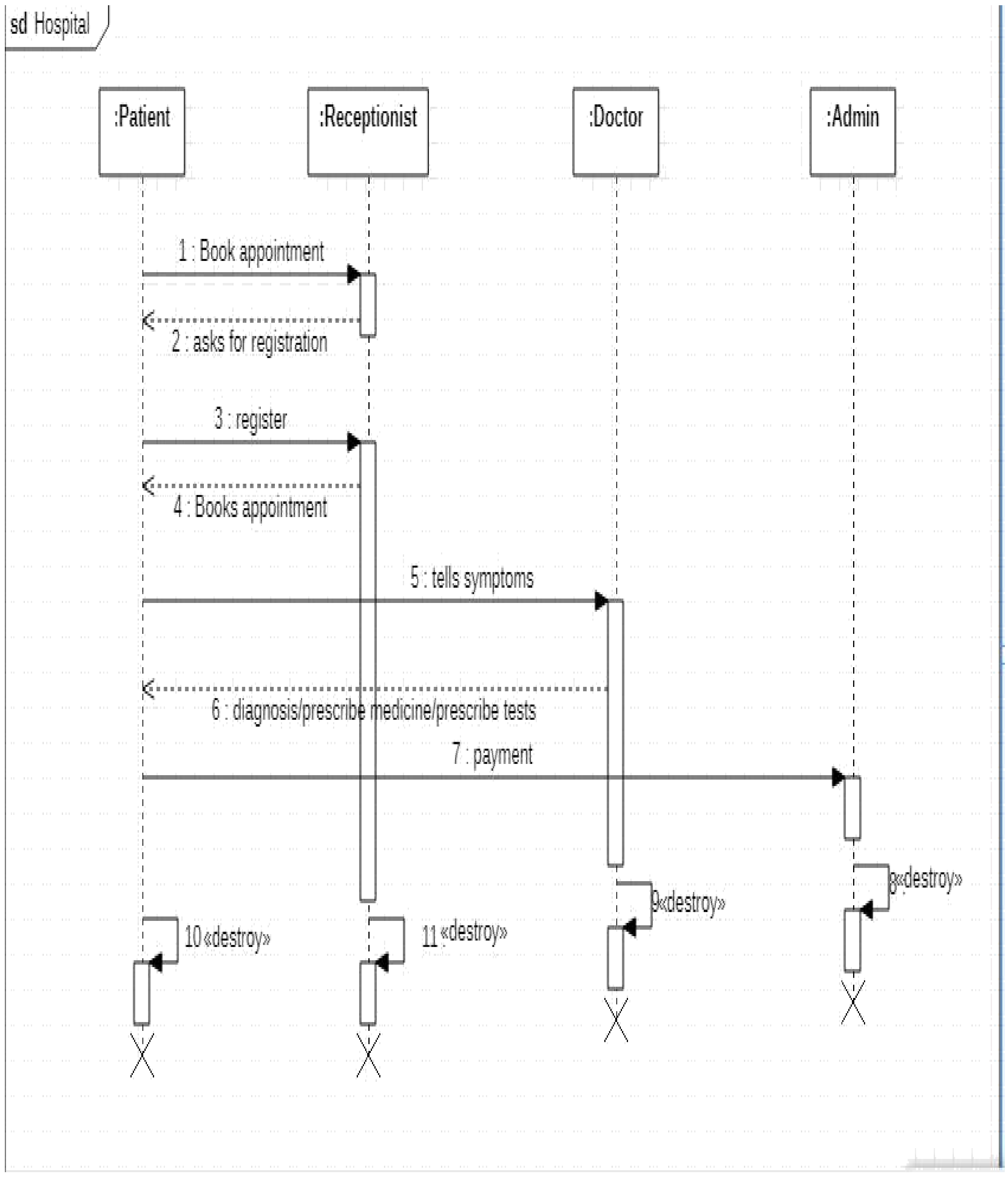
Object Diagram for Hospital Mgmt



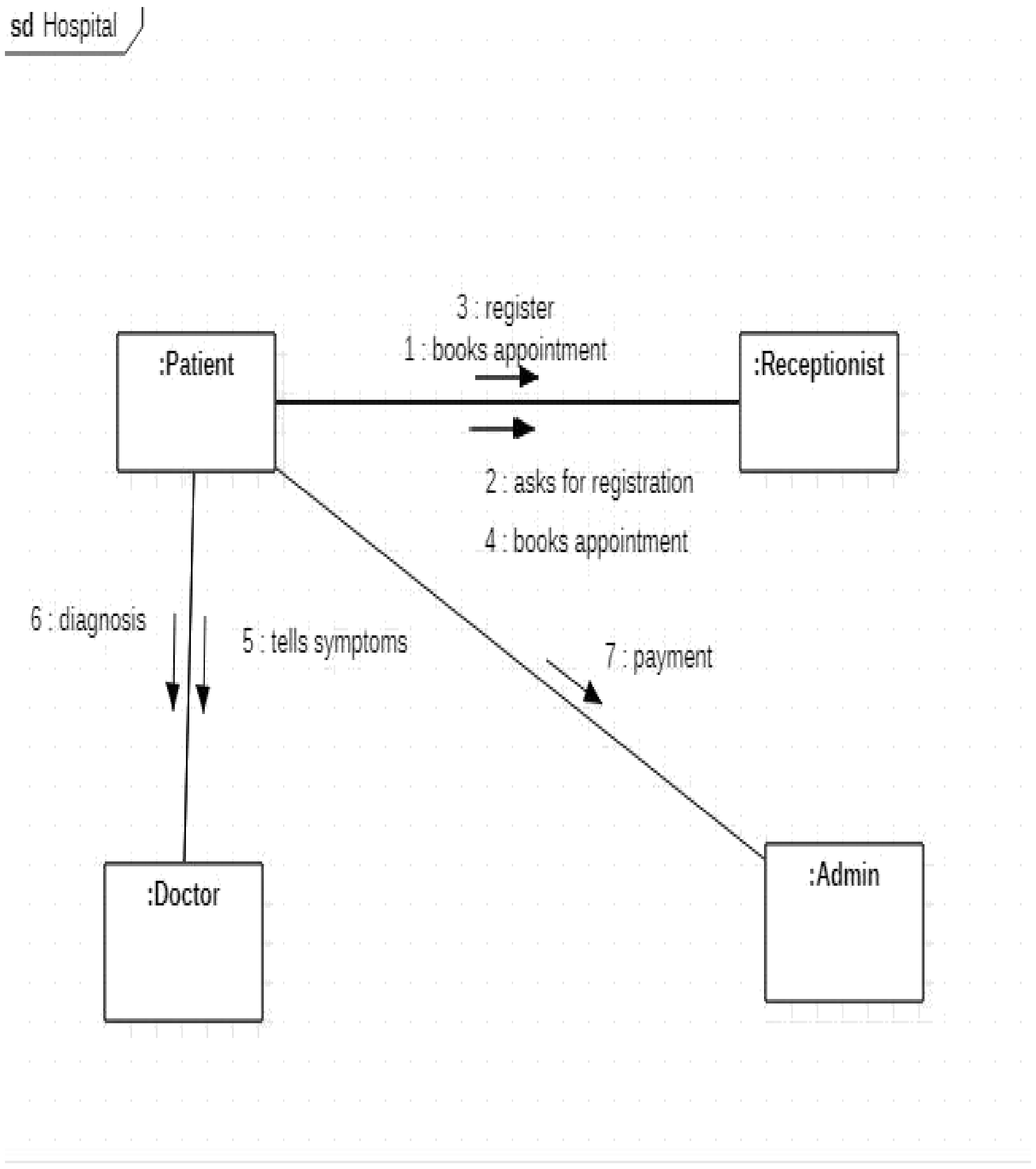
10.2.Use Case Diagram for Hospital Mgmt



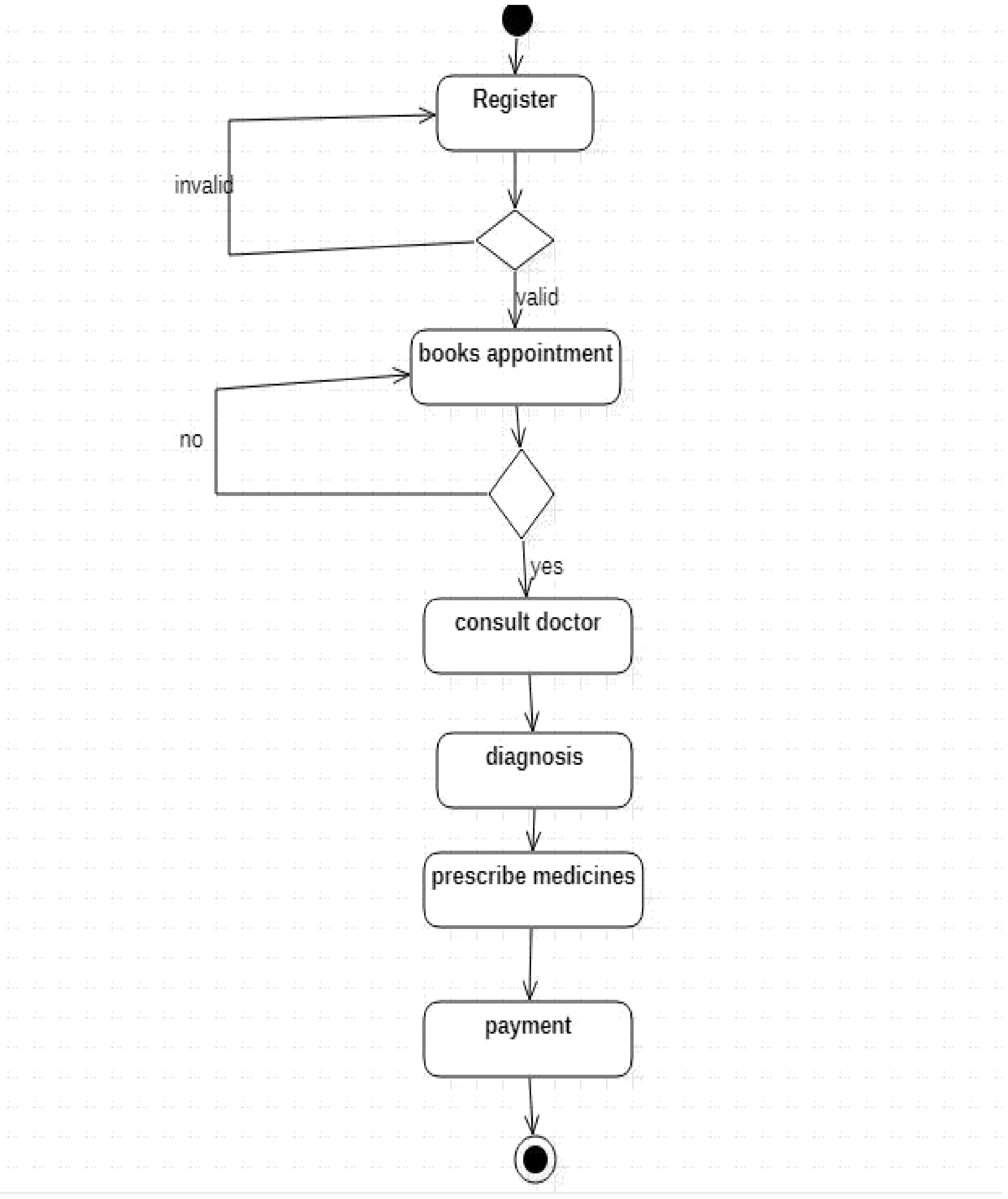
10.3.sequence Diagram for Hospital Mgmt



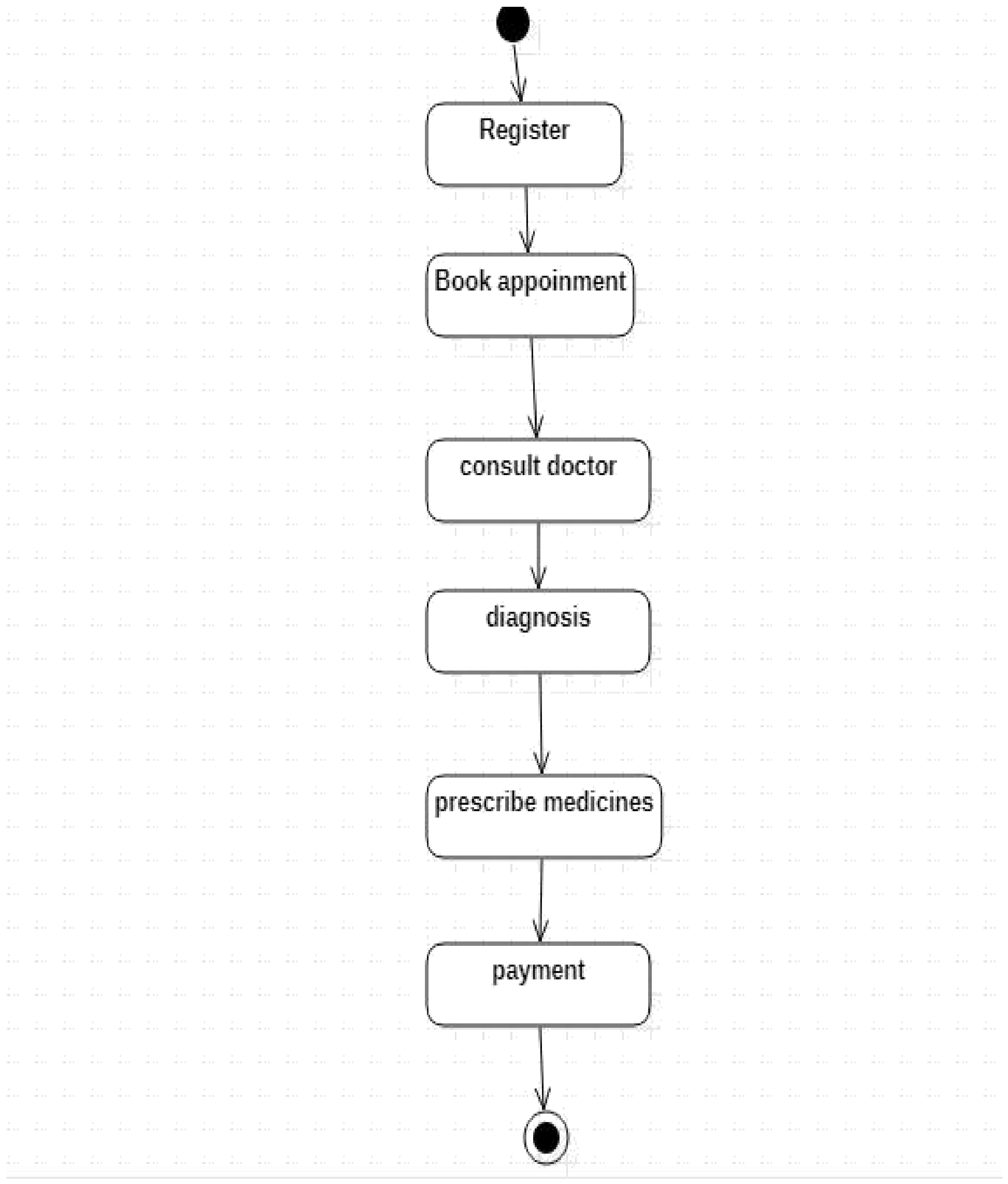
Collaboration Diagram For Hospital Mgmt.



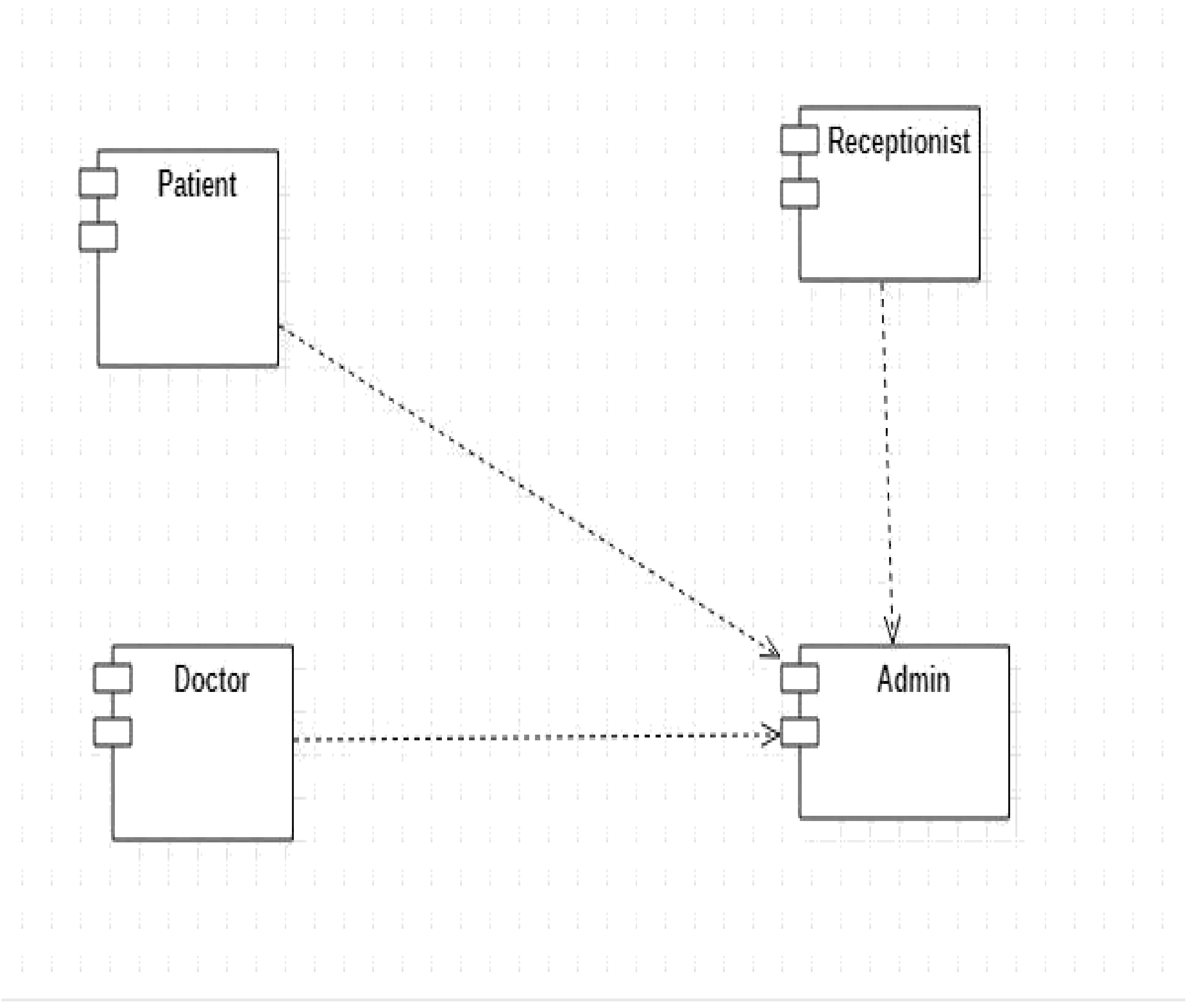
10.4.Activity Diagram For Hospital Mgmt



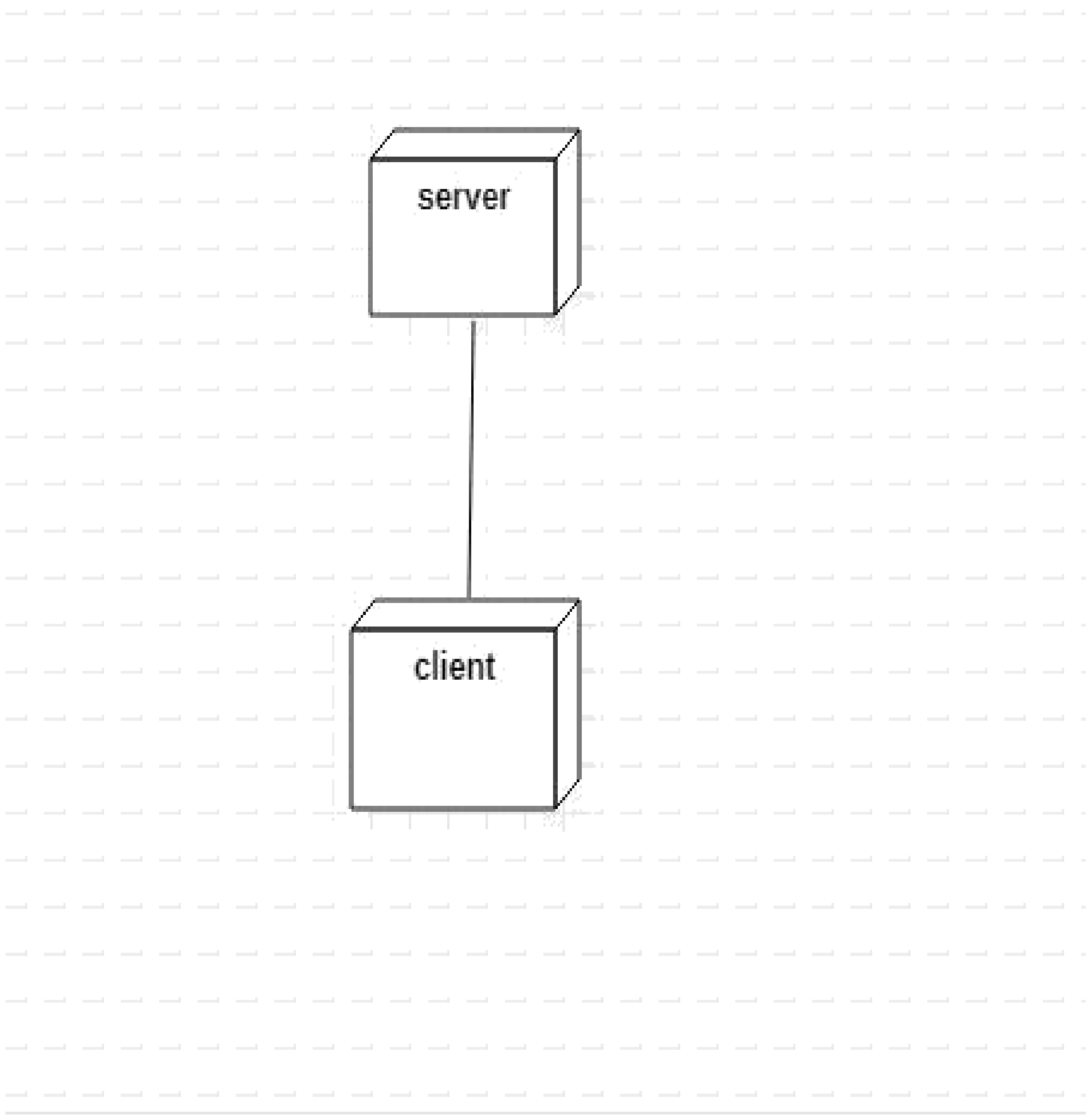
10.5. State Chart Diagram For Hospital Mgmt



10.6.Component Diagram For Hospital Mgmt

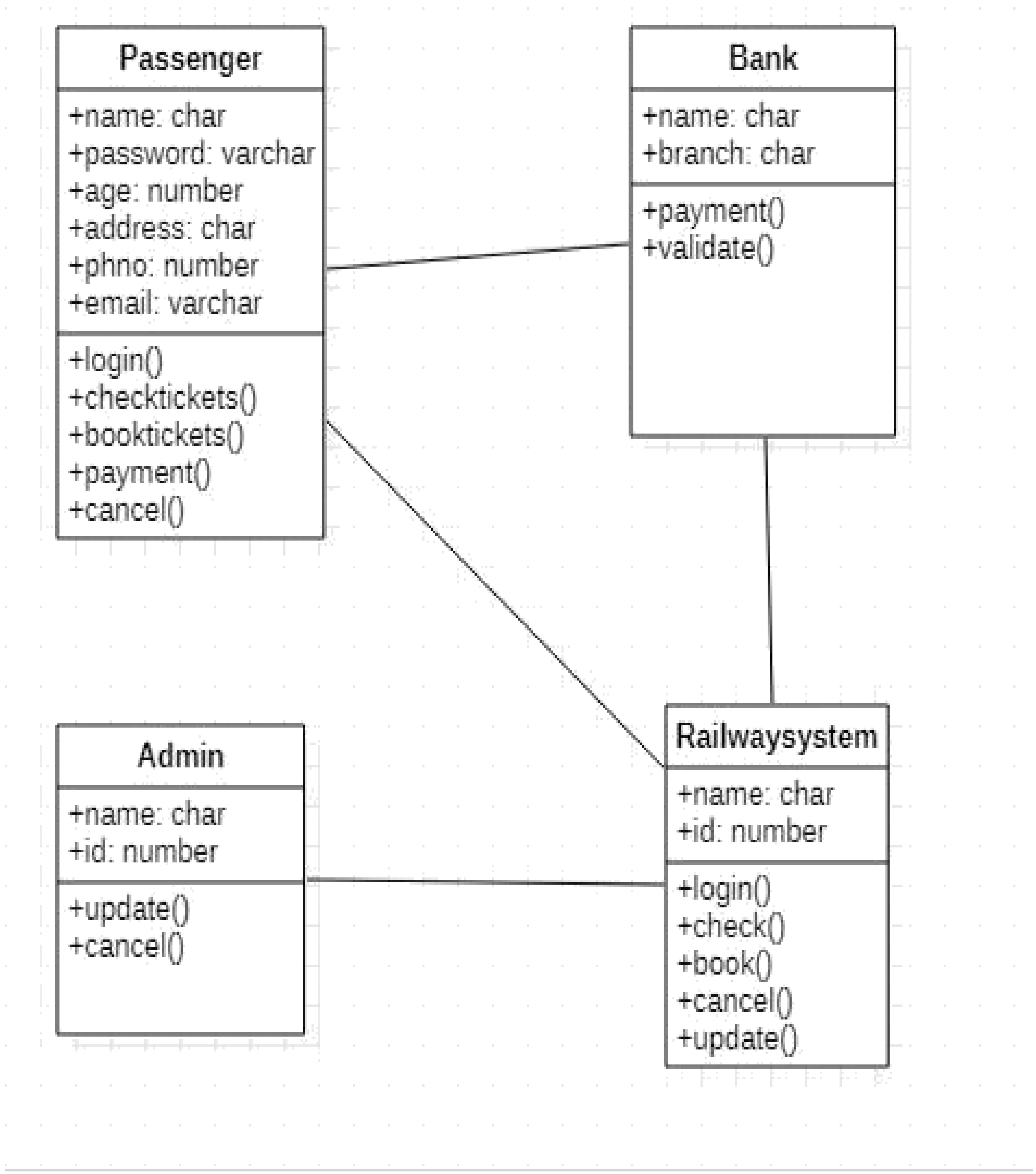


10.7.Deployment Diagram For Hospital Mgmt

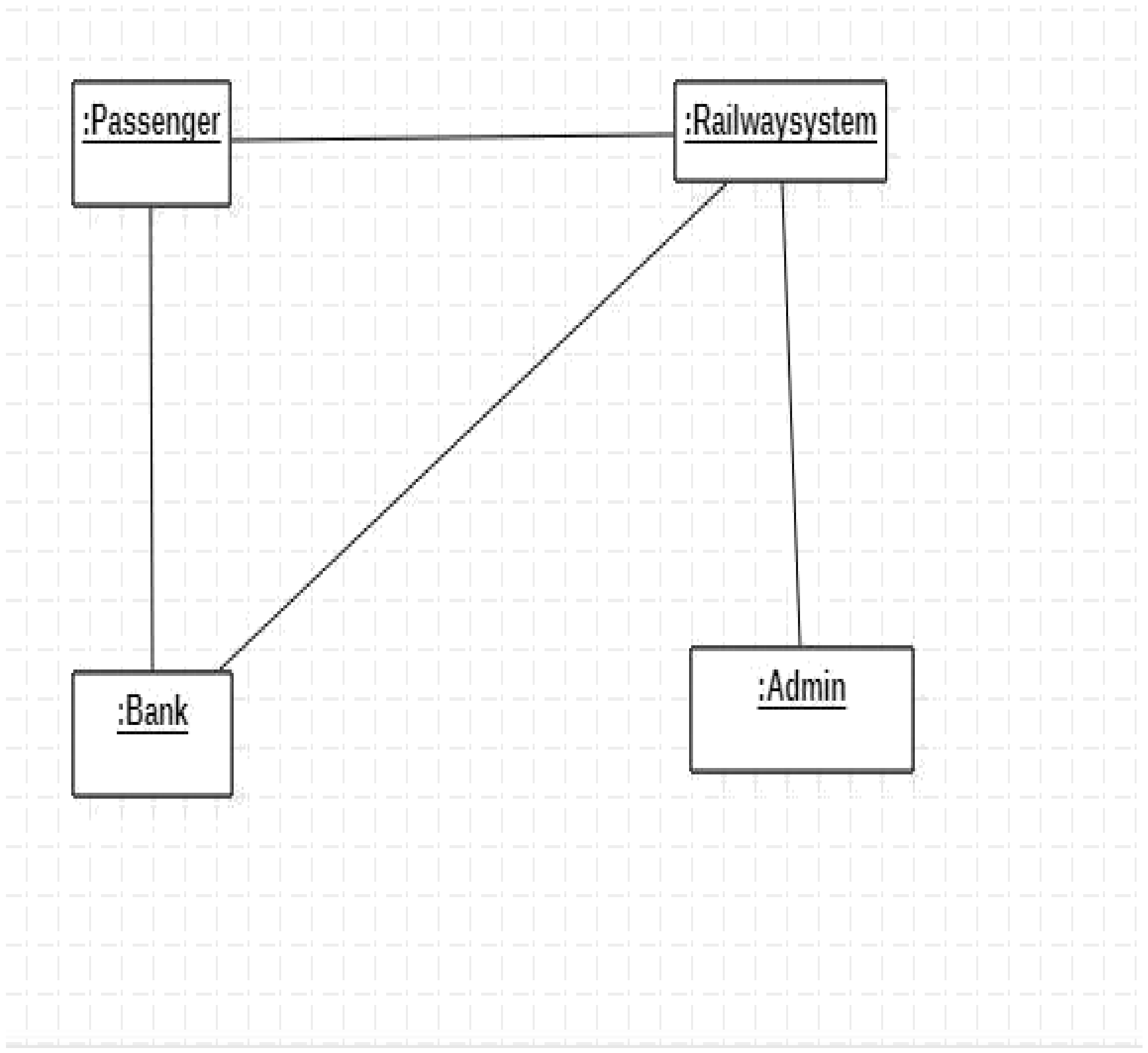


Task-11 Case study for Railway Reservation System.

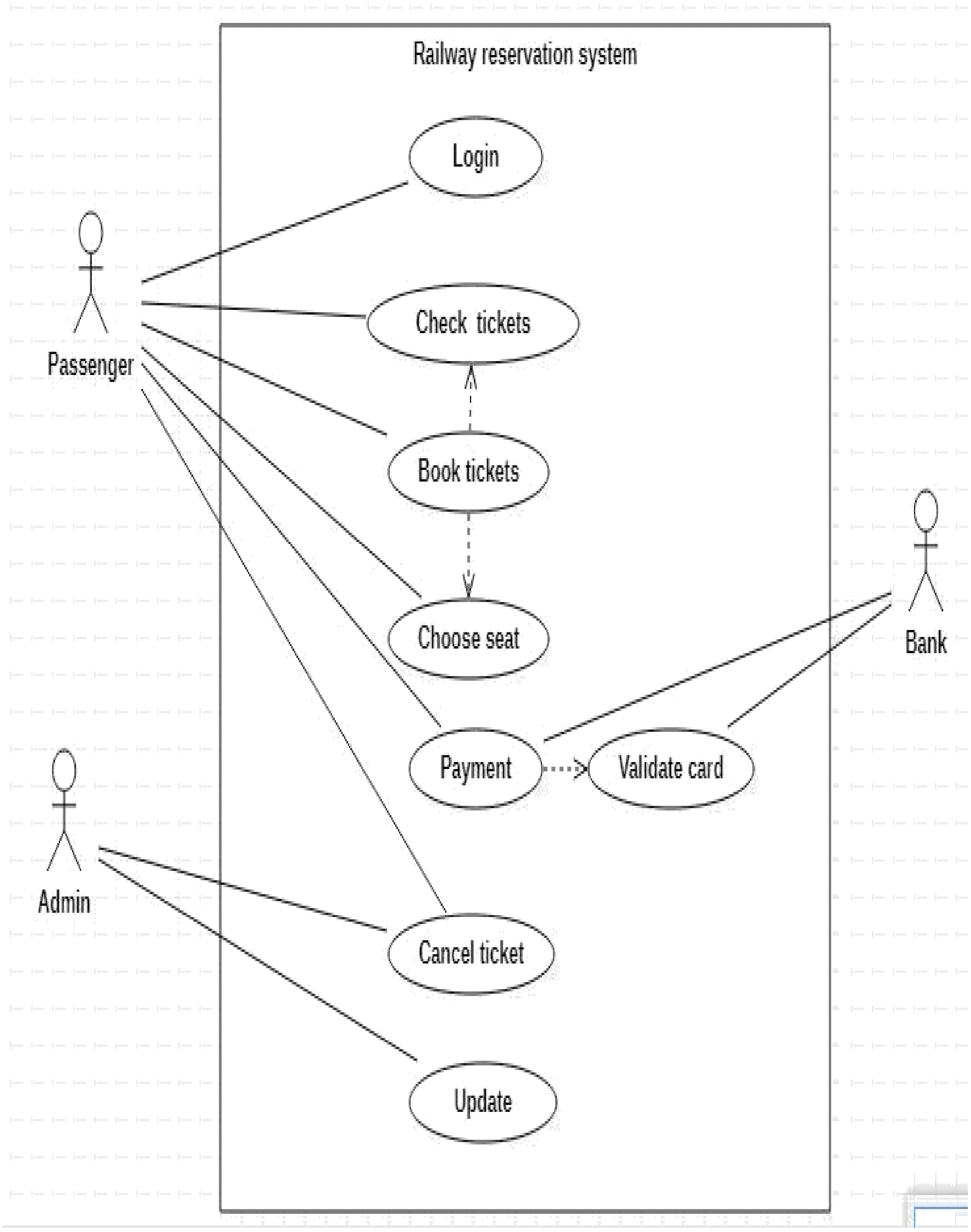
11.1 Class Diagram For Railway Reservation System



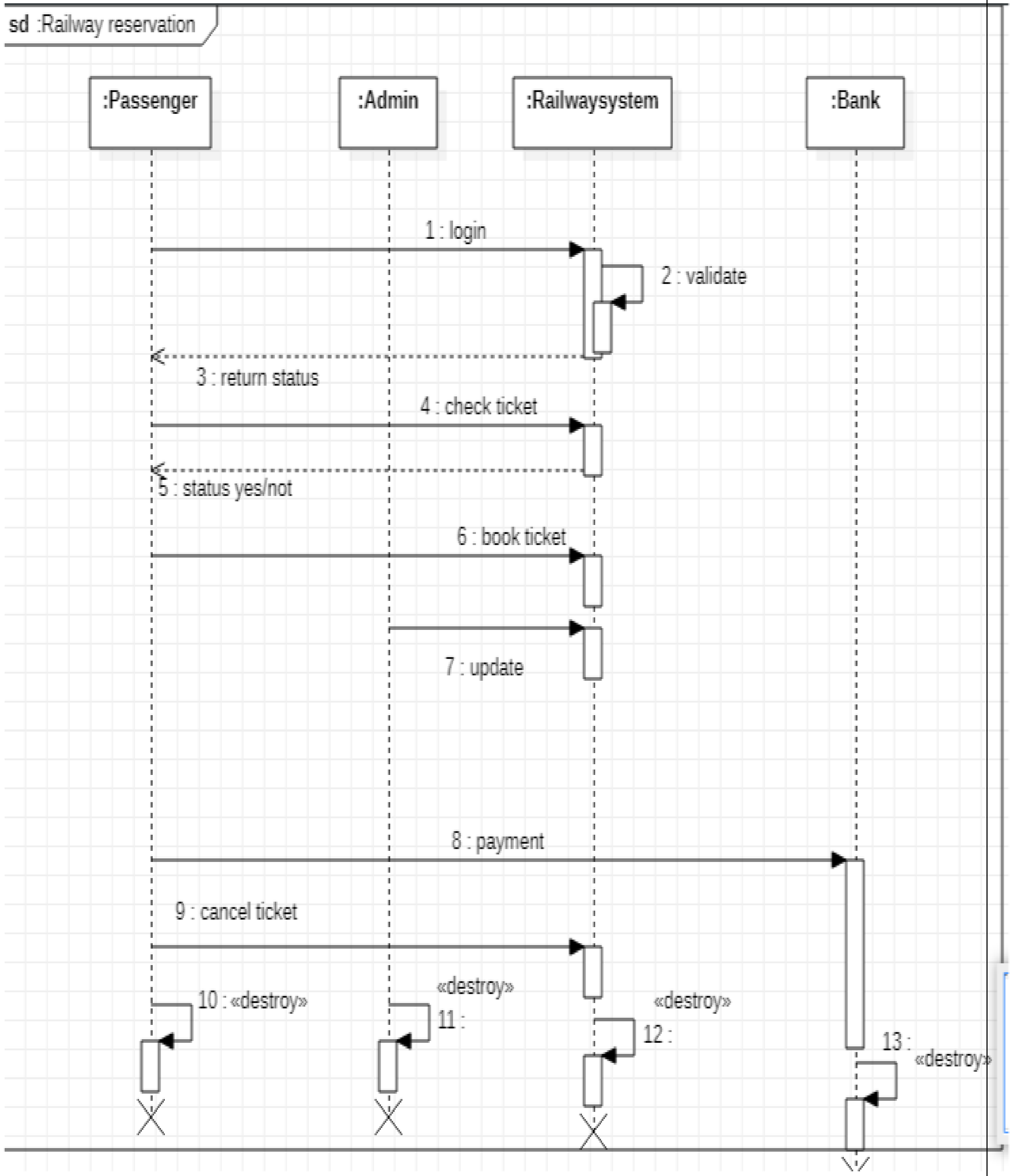
Object Diagram For Railway Reservation System



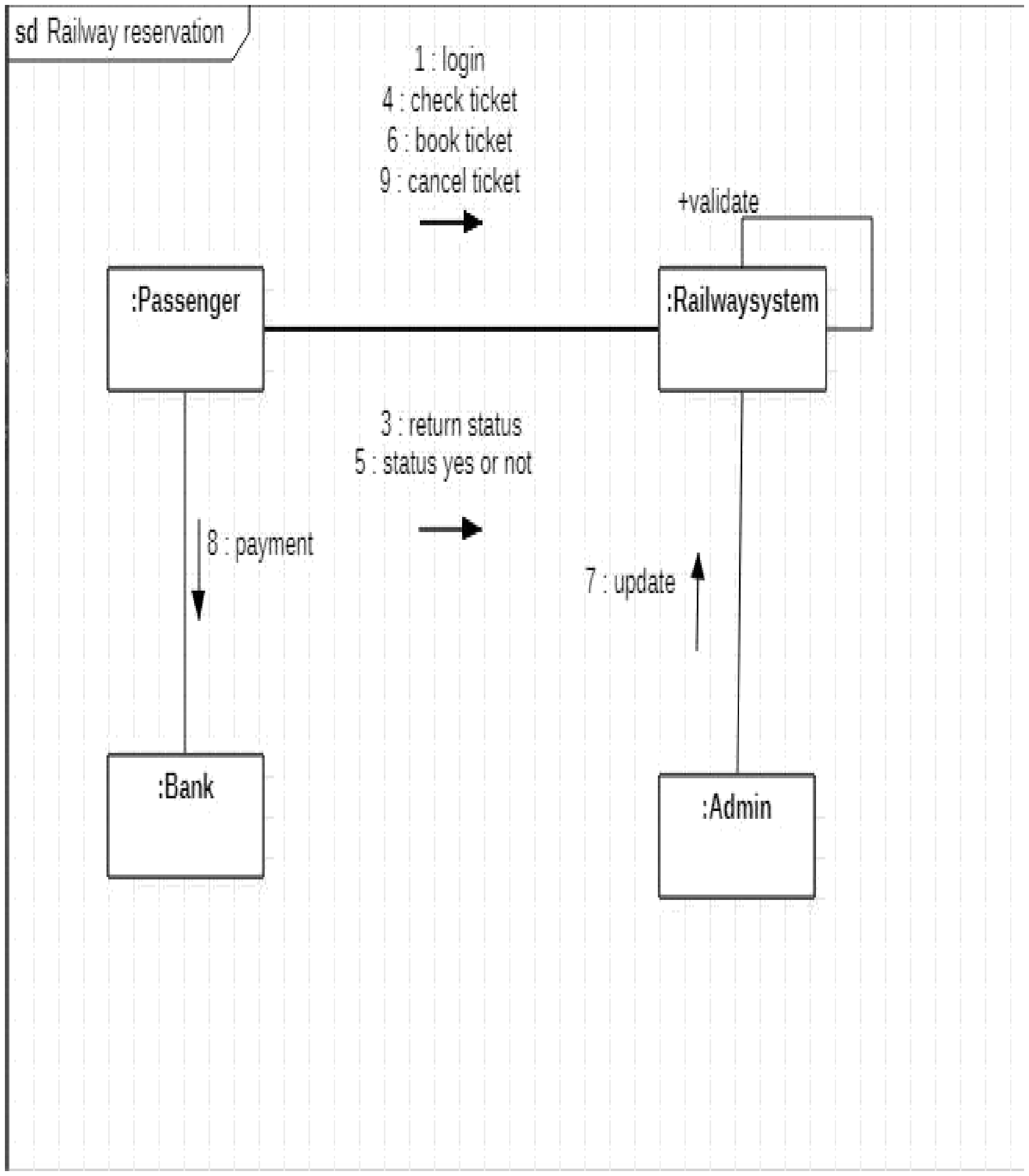
11.2 Usecase Diagram For Railway Reservation System



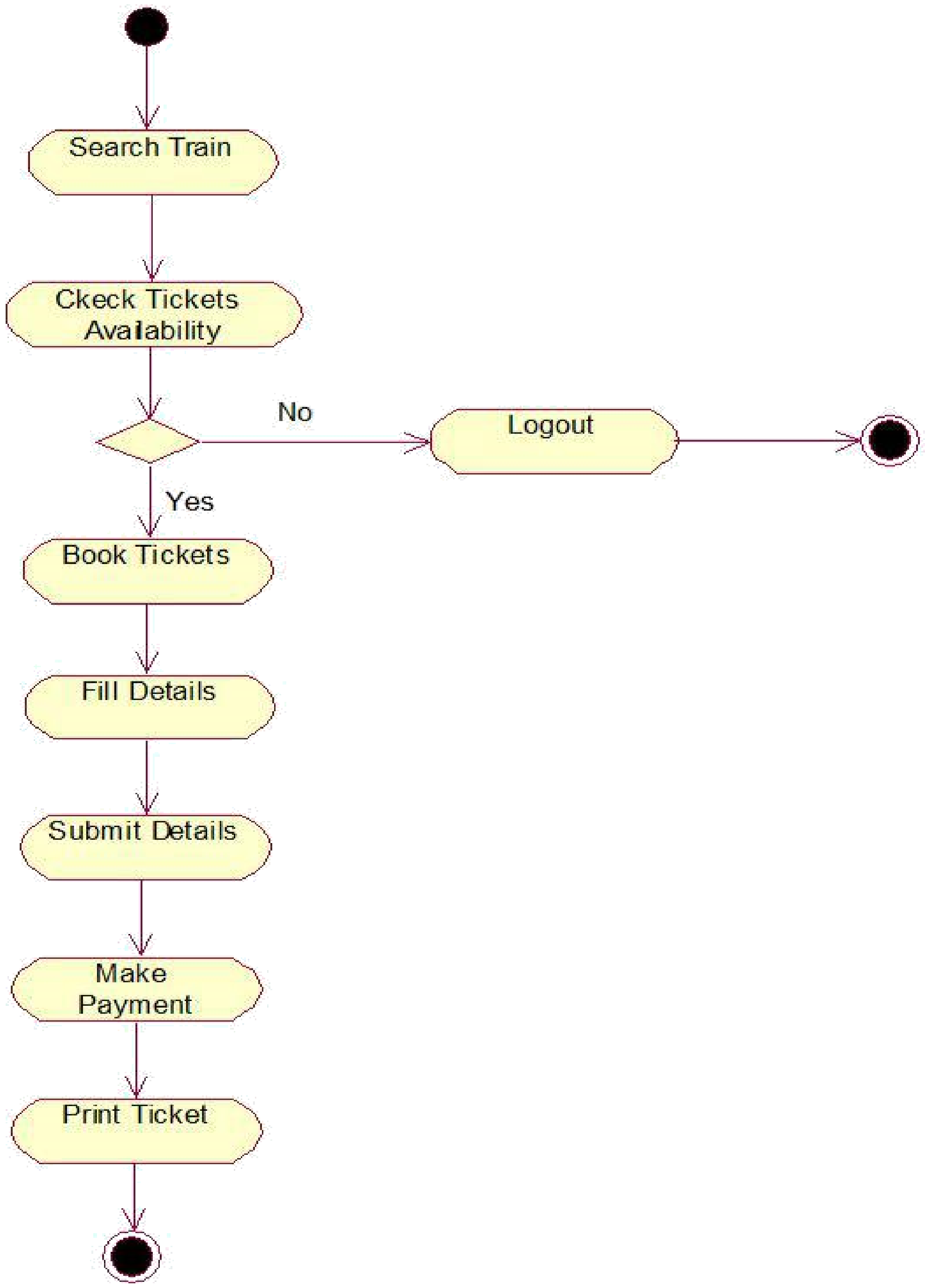
11.3 Sequence Diagram For Railway Reservation System



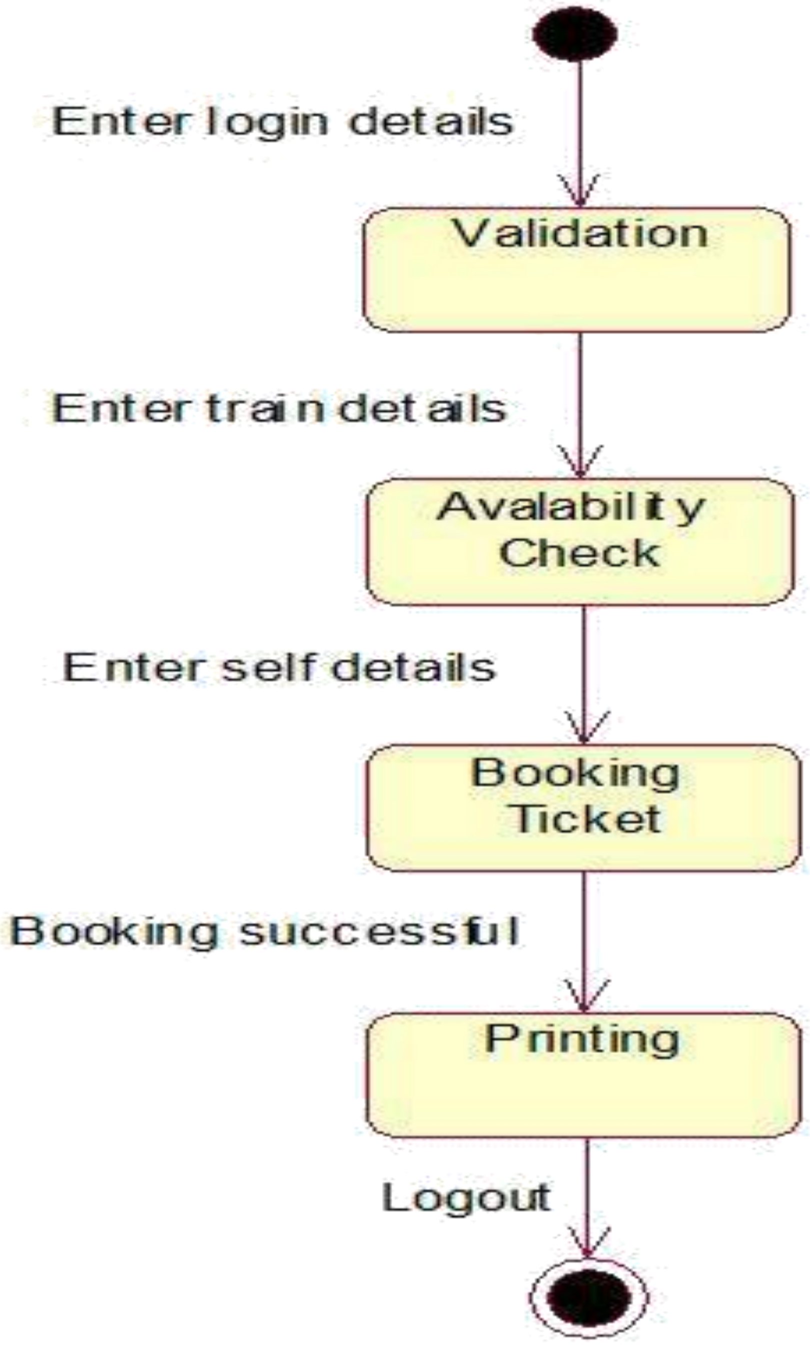
Collaboration Diagram For Railway Reservation System



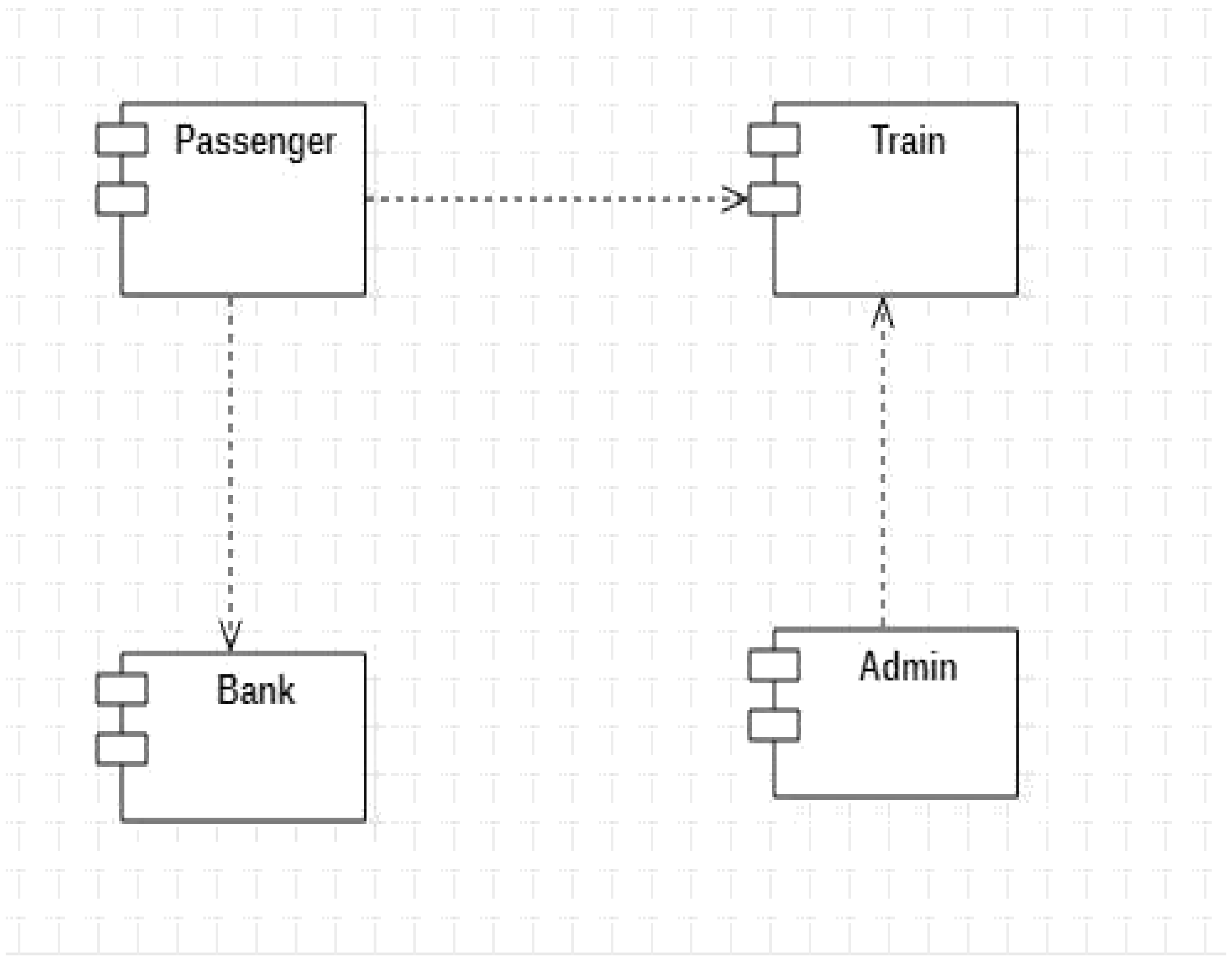
11.4 Activity Diagram For Railway Reservation System



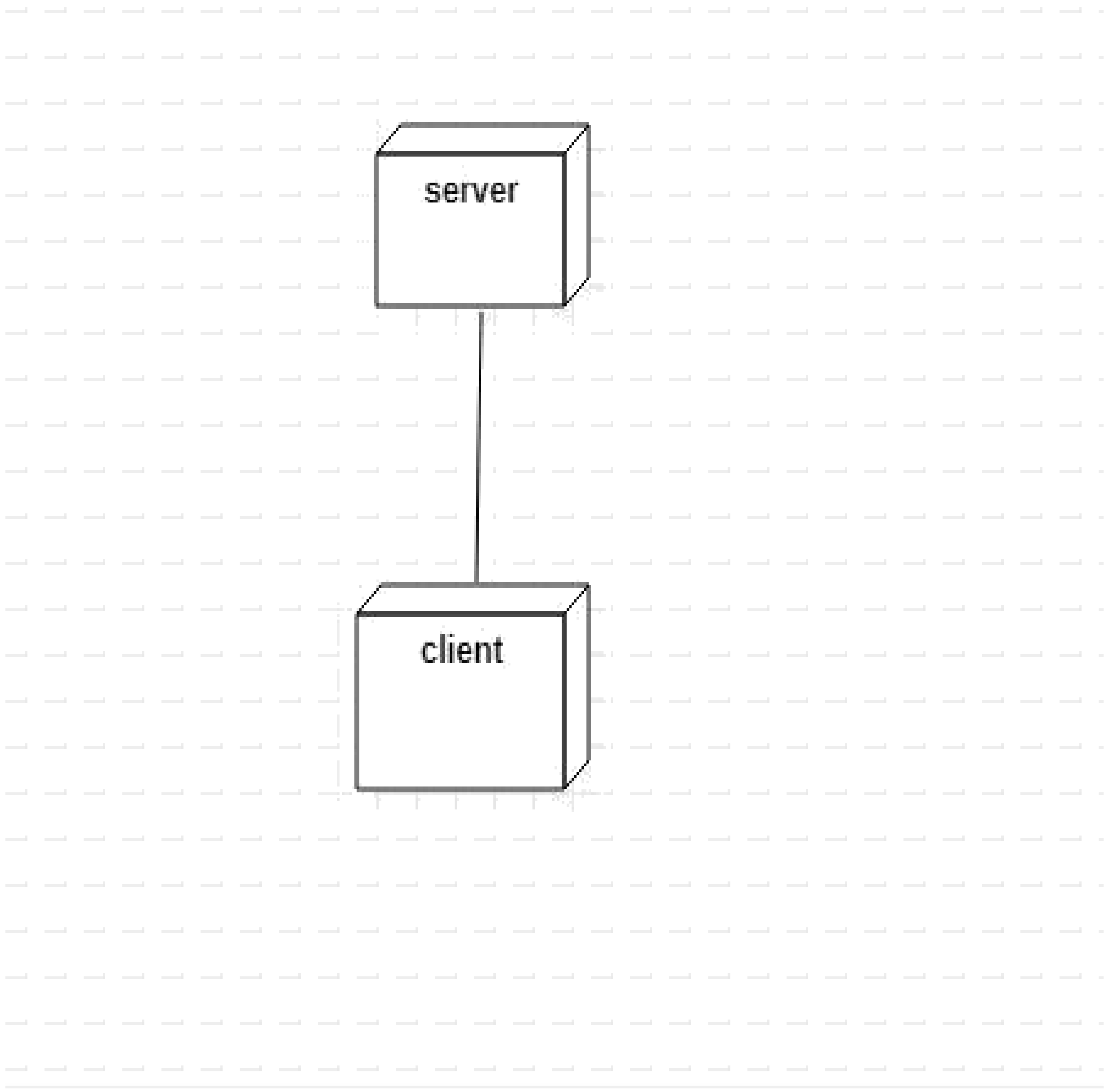
11.5 State chart Diagram For Railway Reservation System



11.6 Component Diagram For Railway Reservation System



11.7 Deployment Diagram For Railway Reservation System


server
client