

# LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)



## Department of Computer Science & Engineering

### 20IT55 – SOFTWARE ENGINEERING LAB

Name of the Student: \_\_\_\_\_

Registered Number: \_\_\_\_\_

Branch & Section: \_\_\_\_\_ & ../Sec

Academic Year: 2021-2022

# LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)



## CERTIFICATE

This is to certify that this is a bonafide record of the practical  
workdone Mr/Ms. ....,  
bearing Regd. Num.: 20761A05..... of B.Tech..... Semester, ..... Branch, .....  
Section in the 20IT55 – SOFTWARE ENGINEERING LAB during the

Academic Year: 2021-2022.

No. of Experiments/Modules held: 11

No. of Experiments Done: 11

Date: .... / \_\_\_\_ / 2022

Signature of the Faculty

INTERNAL EXAMINER

EXTERNAL EXAMINER

## **boarList of Experiments**

### **UML:**

Consider the following Case Studies:

- 1) Automated Teller Machine(ATM)
- 2) Library ManagementSystem
- 3) Railway Ticket ReservationSystem
- 4) Point-of-SaleTerminal
- 5) Customer Support ServiceOperations
- 6) Cab BookingService

### **Week-1: Analyze the Requirements for the following Case Studies**

- 1) Automated Teller Machine(ATM)
- 2) Library ManagementSystem 3) Railway Ticket ReservationSystem

### **Week-2: Analyze the Requirements for the following Case Studies**

- 1) Point-of-SaleTerminal
- 2) Customer Support ServiceOperations
- 3) Cab BookingService

### **Week-3: Basics of UML**

- 1) Introduction toUML
- 2) Familiarization with any one of the Software such as Rational RoseorUmbrello or Gliffy Diagrametc.

### **Week-4: For each case study given earlier, construct Use Case Diagram in the following manner:**

- 1) Identify and Analyze theActors
- 2) Identify theActions
- 3) Analyze the Relationships between Actors andActions
- 4) Sketch the Use CaseDiagram

### **Week-5 and Week-6: For each case study given earlier, Construct Class and Object Diagram in the following manner:**

- 1) Identify and Analyze the Classes related to yourproblem
- 2) Analyze the Attributes andOperations
- 3) Analyze the Relationships betweenClasses
- 4) Sketch the ClassDiagram

### **Week-7: For each case study given earlier, Construct Interaction Diagrams in the following manner:**

- 1) Identify the Objects participating inCommunication
- 2) Identify the Messages between theobjects
- 3) Give numbering tomessages
- 4) Use Flat Sequencing or Procedural Sequencing fornumbering

**Week-8: For each case study given earlier, Construct Activity Diagram in the following manner:**

- 1) Identify activities in your casestudy.
- 2) Identify relationships amongactivities.
- 3) Use Fork or Join, ifnecessary. 4) Sketch thediagram

**Week-9: For each case study given earlier, Construct State Chart Diagram in the following manner:**

- 1) Identify the different states in your case study.
- 2) List out the different sub-states present in the state.
- 3) Identify relationships among the state to state. 4) Sketch the diagram.

**Week-10:For each case study given earlier, Construct Component Diagram in the following manner:**

- 1) Identify the different components in your casestudy.
- 2) Create a visual for each of thecomponent.
- 3) Describe the organization and relationships between components using interfaces, portsetc.
- 4) Sketch the diagram

**Week-11: For each case study given earlier, construct Deployment Diagram in the following manner:**

- 1) Identify the nodes.
- 2) Identify the relationships among thenodes.
- 3) Sketch the Diagram.

## 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."

### Why use UML?

Two questions, really:

- 1) Why use a graphical notation of anysort?

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 beused?

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 Jacobson—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 for different 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:

- Class diagrams
- Object 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

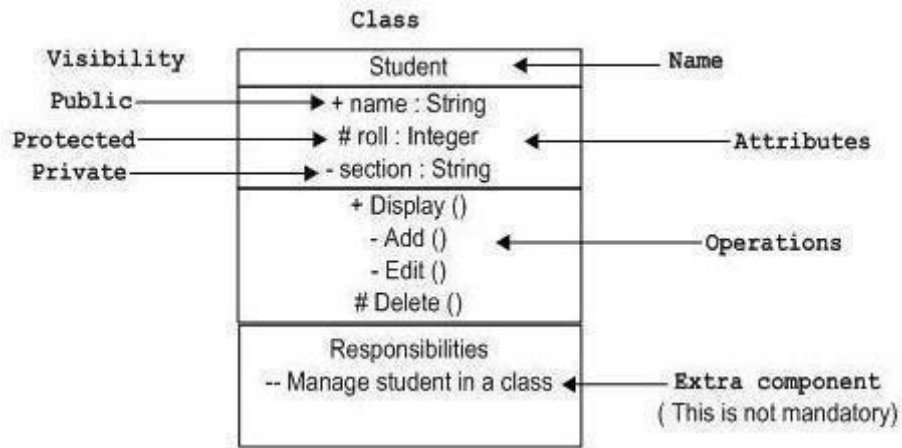
Graphical notations used in structural things are the most widely used in UML. These are considered as the nouns of UML models. Following are the list of structural things.

- Classes
- Interface
- Collaboration
- Usecase
- Activeclasses
- Components
- Nodes

### Class Notation:

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

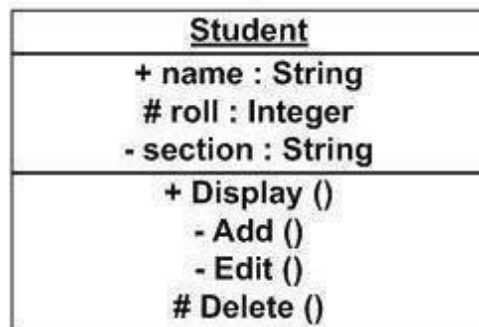
- The top section is used to name the class.
- The second one is used to show the attributes of the class.
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.



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:



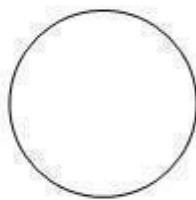
As object is the actual implementation of a class which is known as the instance of a class. So it has the same usage as the class.

#### 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

**Interface**

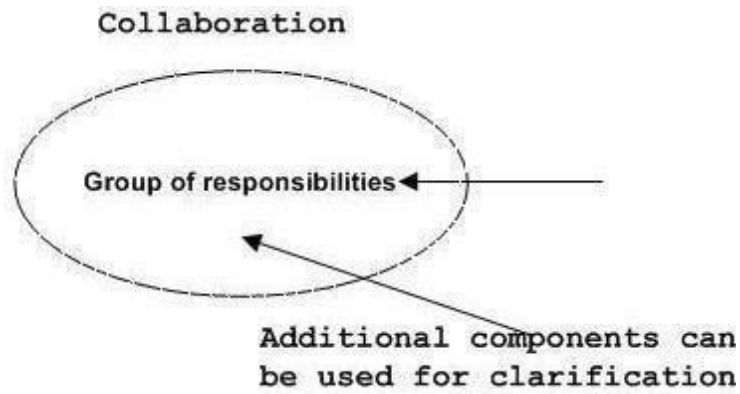


**StudentApplication** ← **Name**

you defined different functions not the implementation. When a class implements the interface it also implements the functionality as per the requirement.

#### Collaboration Notation:

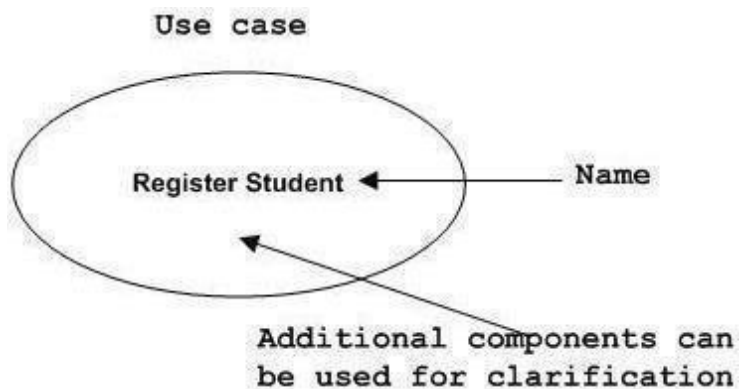
Collaboration is represented by a dotted ellipse as shown below. It has a name written inside the ellipse.



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:**

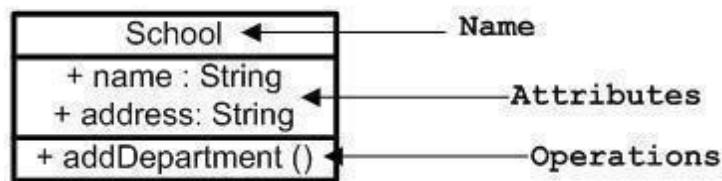
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.



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

**Active class Notation:**

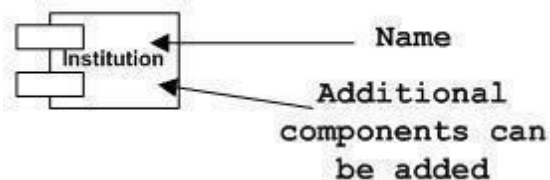
Active class looks similar to a class with a solid border. Active class is generally used to describe concurrent behavior of a system.

**Active Class**

Active class is used to represent concurrency in a system.

**Component Notation:**

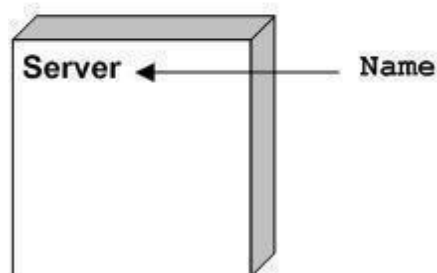
A component in UML is shown as below with a name inside. Additional elements can be added wherever required.

**Component**

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**

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

**Behavioural Things:**

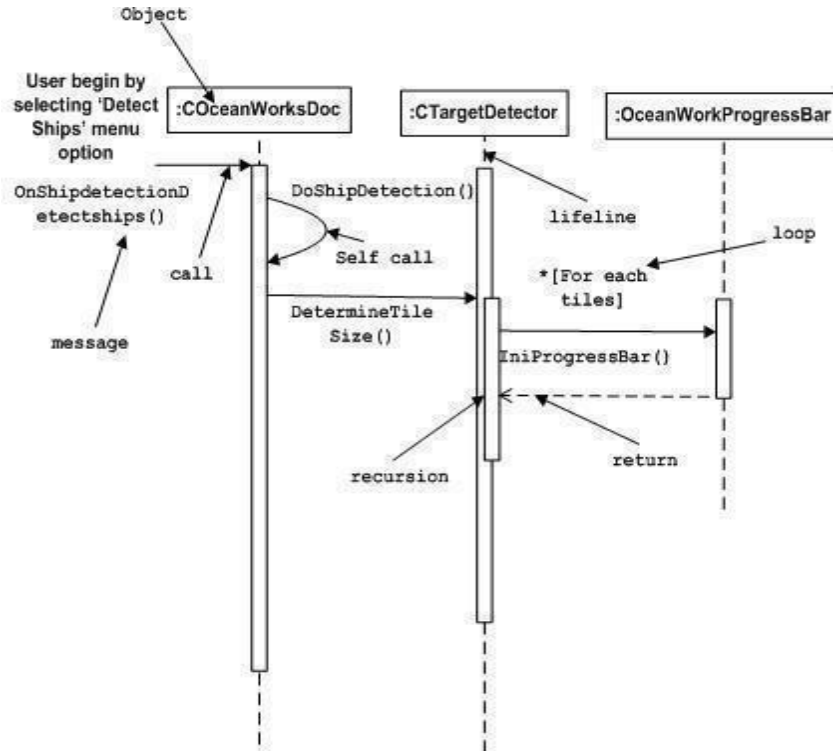
Dynamic parts are one of the most important elements in UML. UML has a set of powerful features to represent the dynamic part of software and non-software systems. These features include interactions and state machines.

Interactions can be of two types:

- Sequential (Represented by sequencediagram)
- Collaborative (Represented by collaborationdiagram)

#### Interaction Notation:

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

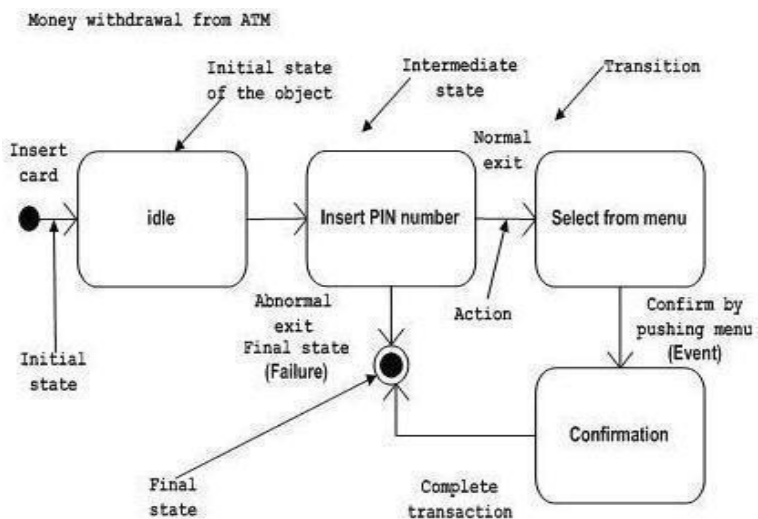


Interaction is used to represent communication among the components of a system.

#### State machine Notation:

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

Statemachine issued todescribedifferentstatesofasystemcomponent.Thestatecanbeactive,idle or any other depending upon thesituation.

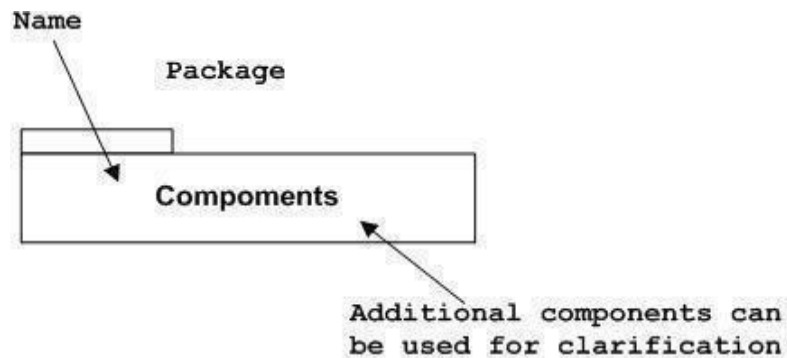


## Grouping Things:

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.

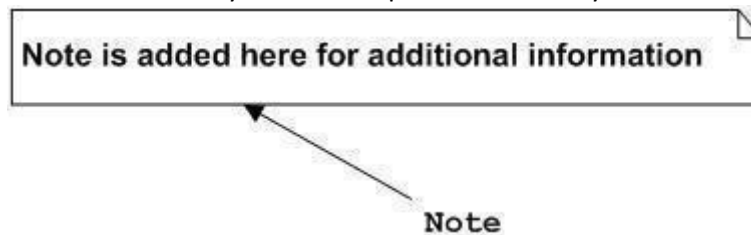


## Annotational Things:

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.



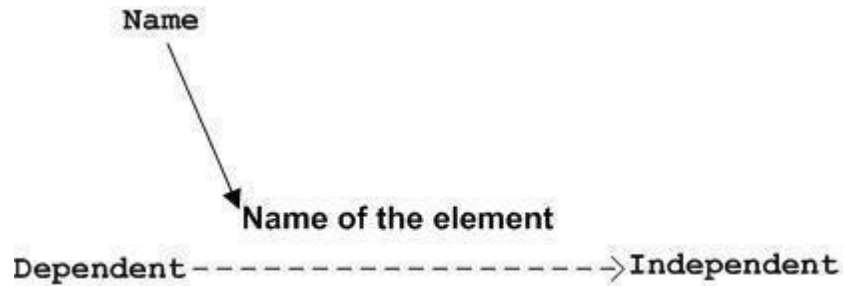
## Relationships

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.

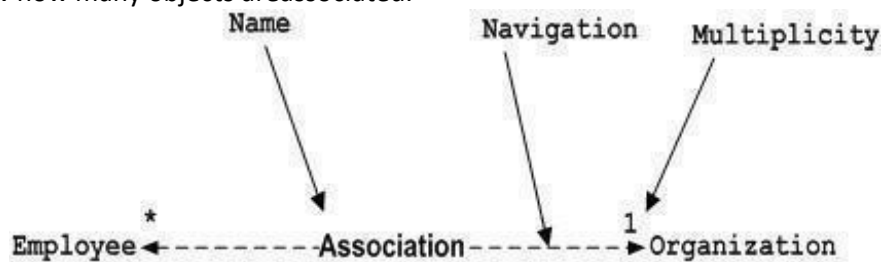


Dependency is used to represent dependency between two elements of a system.

#### Association Notation:

Association describes how the elements in a UML diagram are associated. In simple words, 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 end represents the parent element and the other end child element.

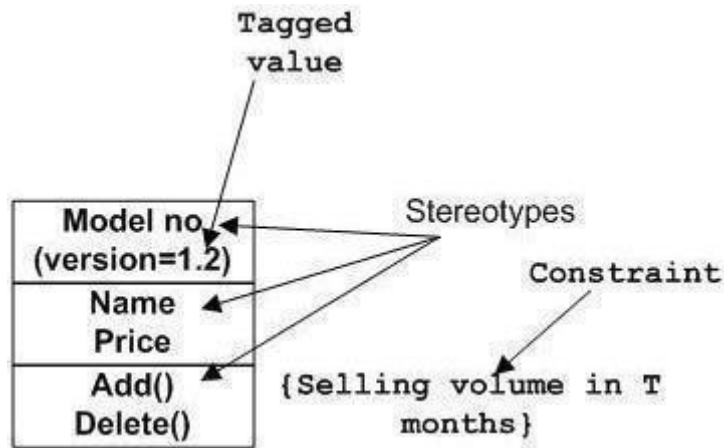


Generalization is used to describe parent-child relationship of two elements of a system.

#### 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 behavior of the system. These extra behaviors 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 relationship 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 end of 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 relationship because each customer can have multiple orders.
- We would keep Order class as an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive().

## 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 behavior. 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 behaviour 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 diagram the following things should be remembered and understood clearly:

- Object diagrams 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.
- Values 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 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 reside 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 need 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 an 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 files, 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 a completely different purpose.

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 show 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 hardware 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.

## 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 and 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 use 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 usecase
- 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 function 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. So if we look into the diagram then we will find three use cases (Order, Special Order and Normal Order) and one actor which is customer.

The Special Order and Normal Order 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 described 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 snapshot 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. These 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 be 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 whereas 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, a 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 lifetime of a reactive system.
- To describe different states of an object during its lifetime.
- 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 from 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 deal 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 sometime considered as the flow chart. Although the diagram 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 mentally out of the entire flow. This mentally out 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:

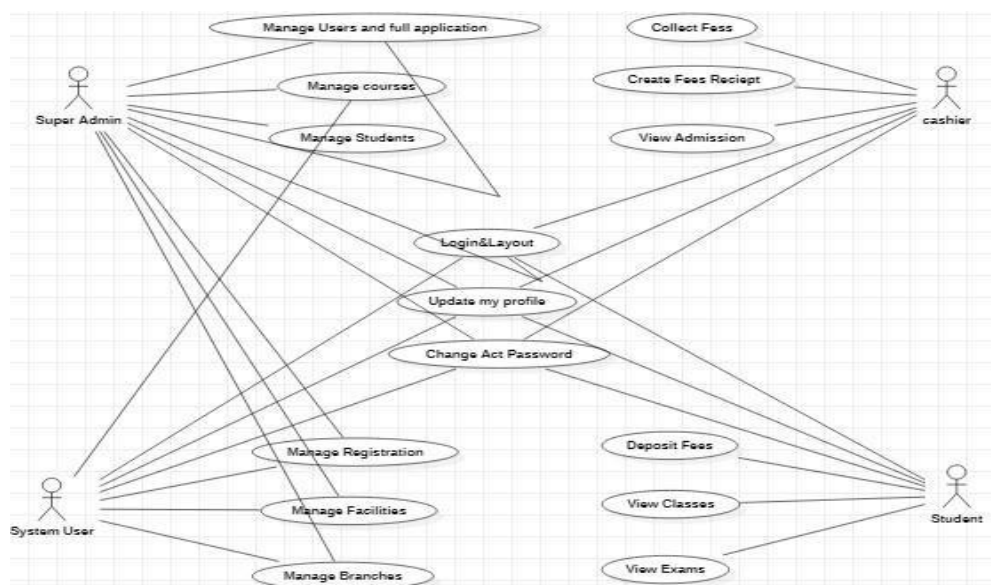
- Send order by the customer
- Receipt of the order
- Confirm order
- Dispatch order

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.

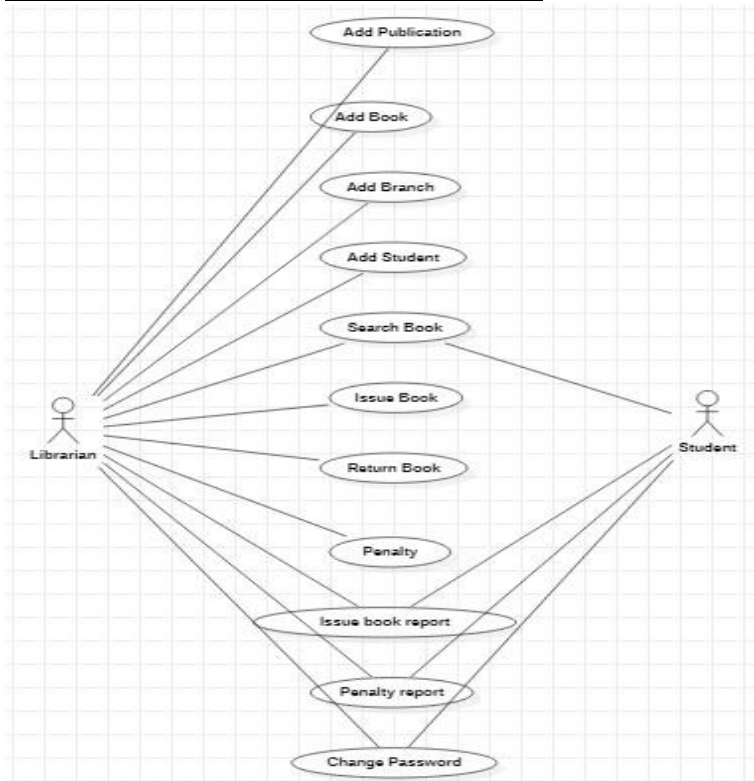
**Week-4: For each case study given earlier, construct Use Case Diagram in the following manner:**

### USE CASE DIAGRAMS:

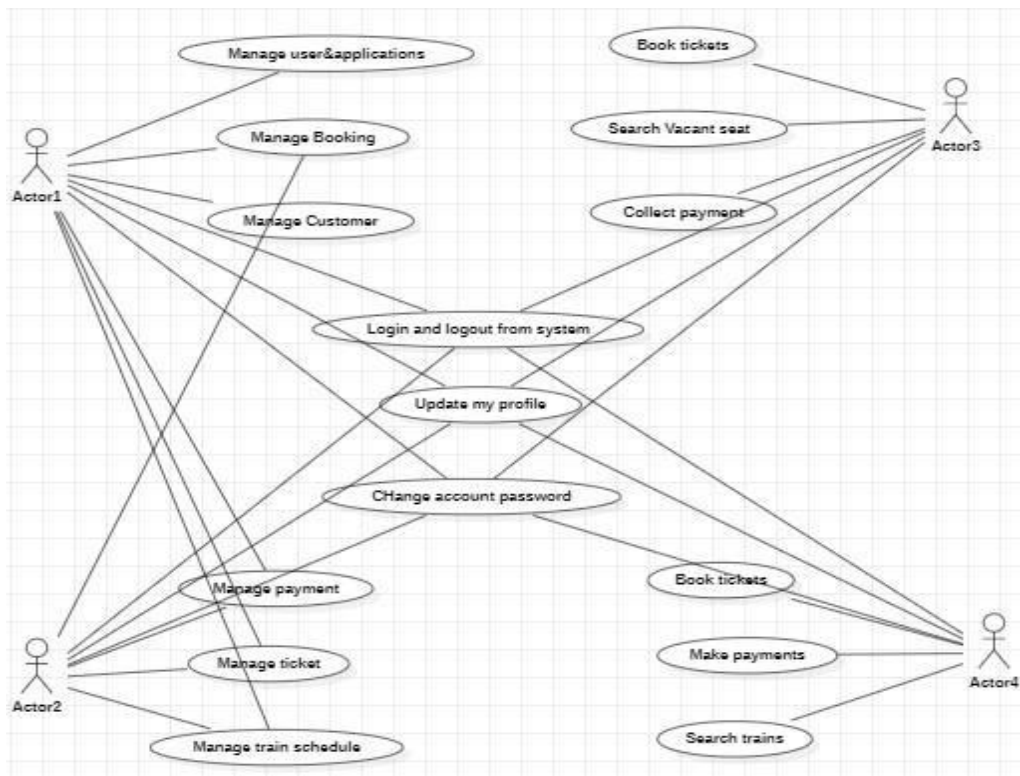
### COLLEGE MANAGEMENT SYSTEM:



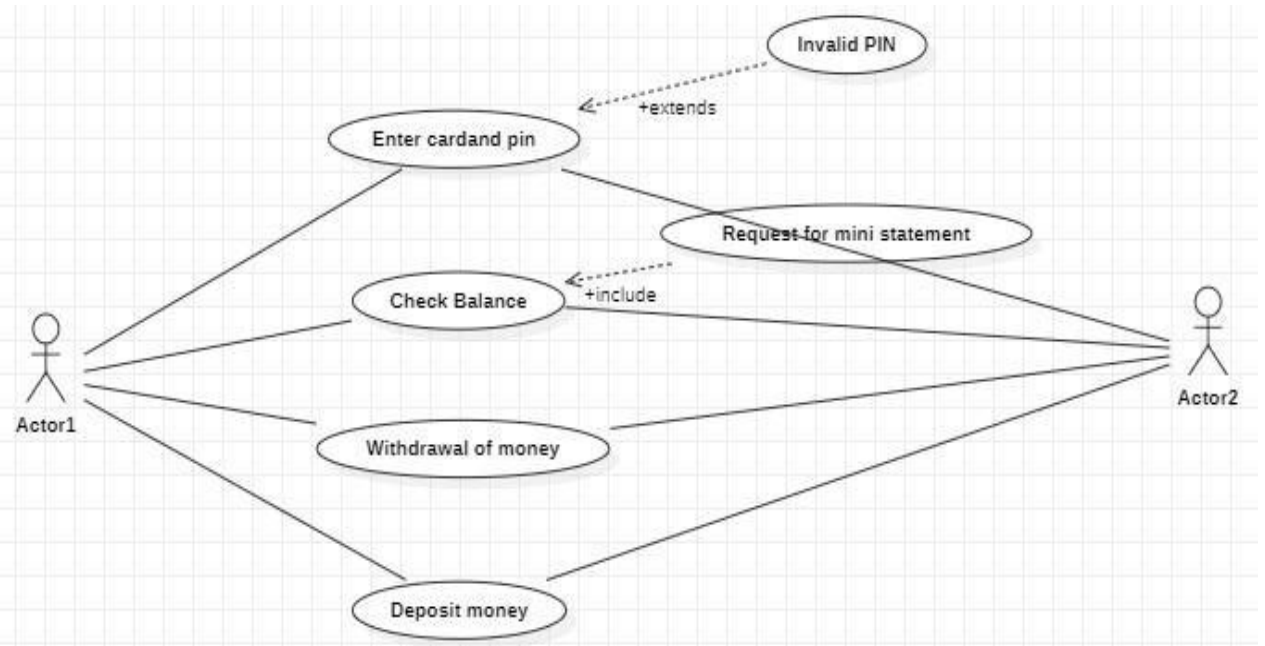
## LIBRARY MANAGEMENT SYSTEM



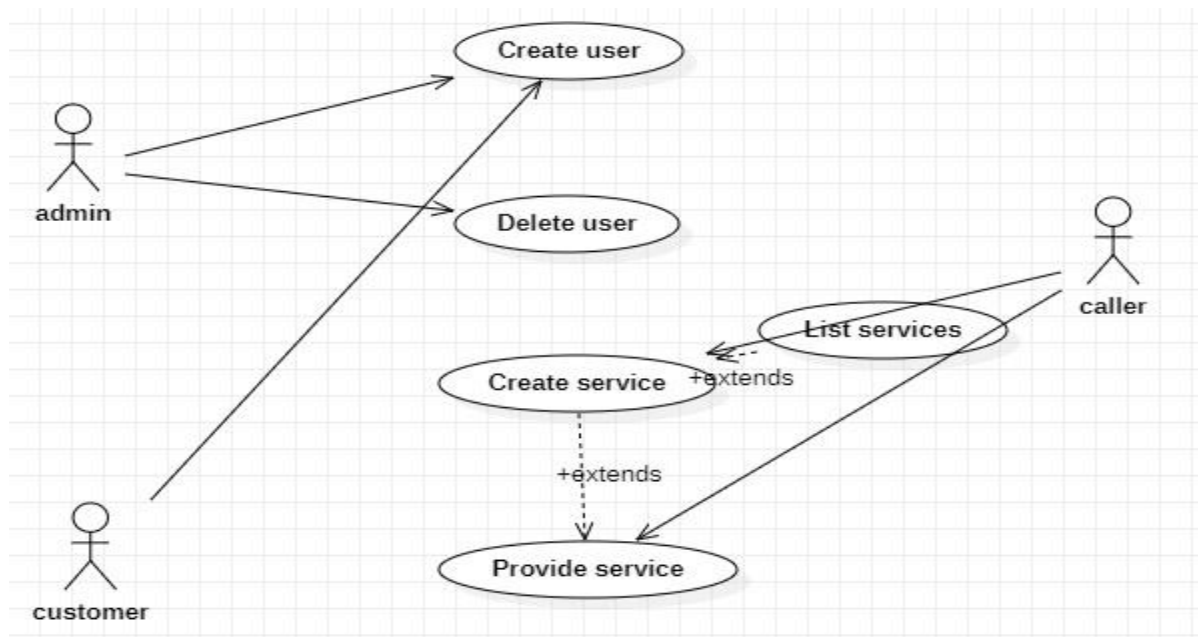
## RAILWAY RESERVATION SYSTEM:



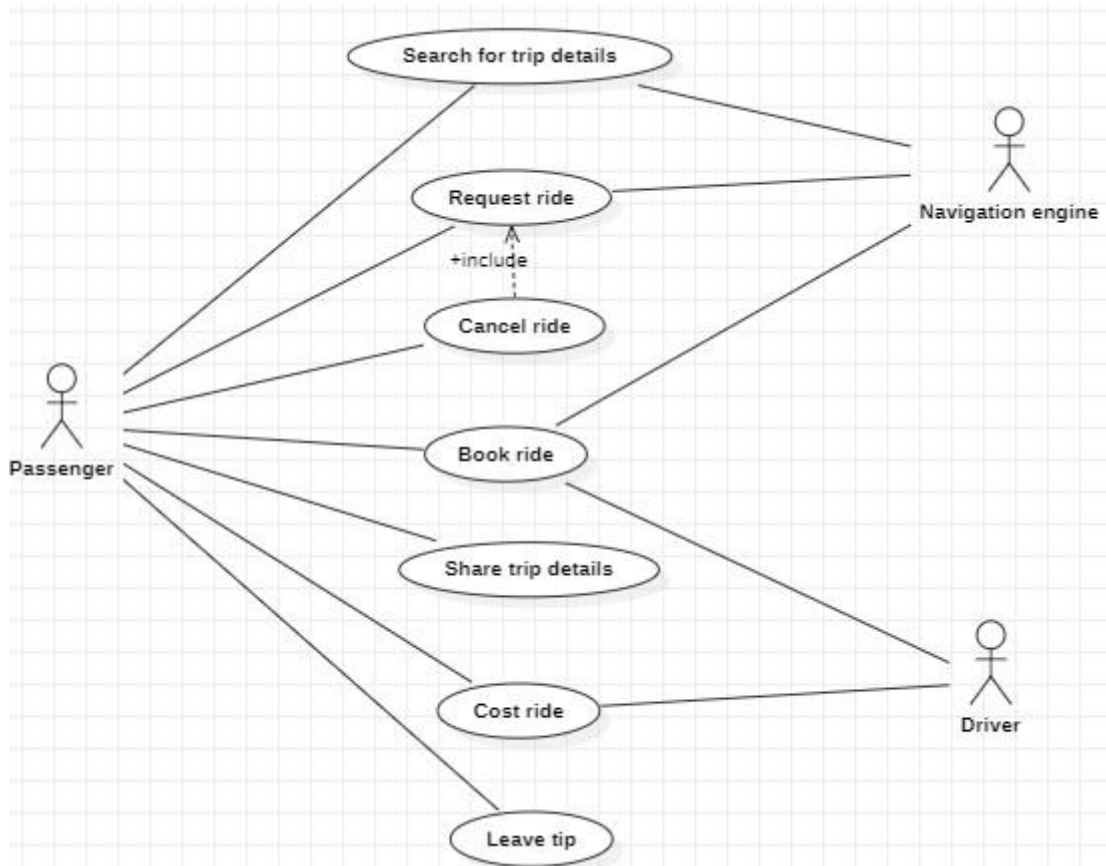
## ATM:



### **CUSTOMER SUPPORT SYSTEM:**

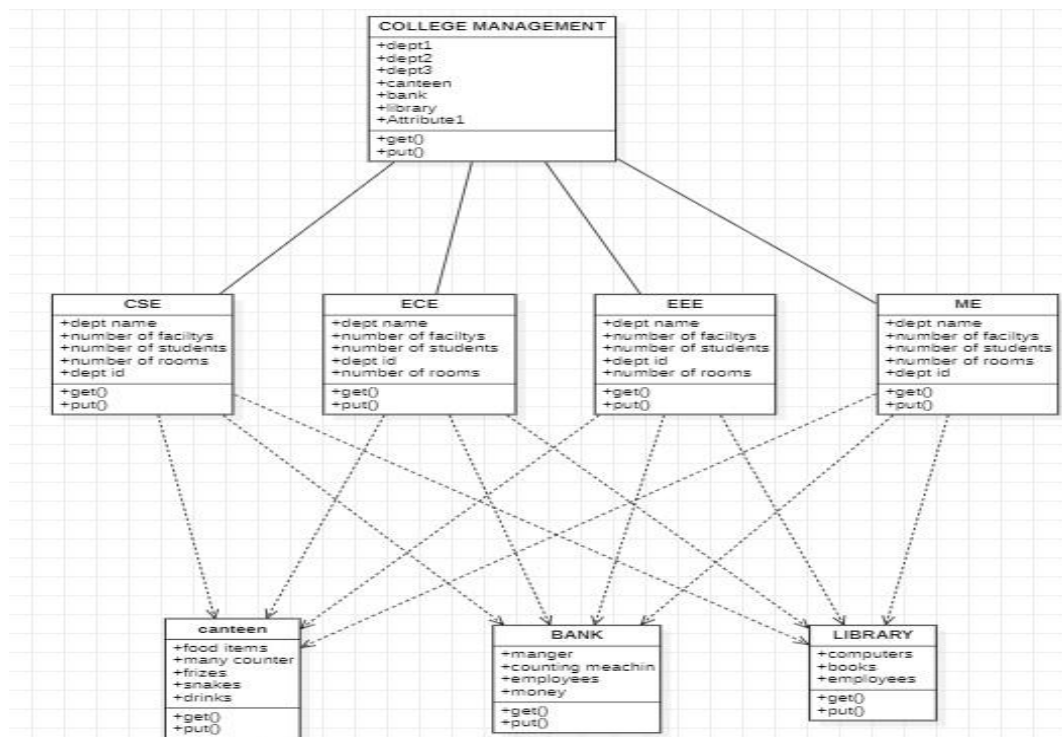


### **UBER CAB SYSTEM:**



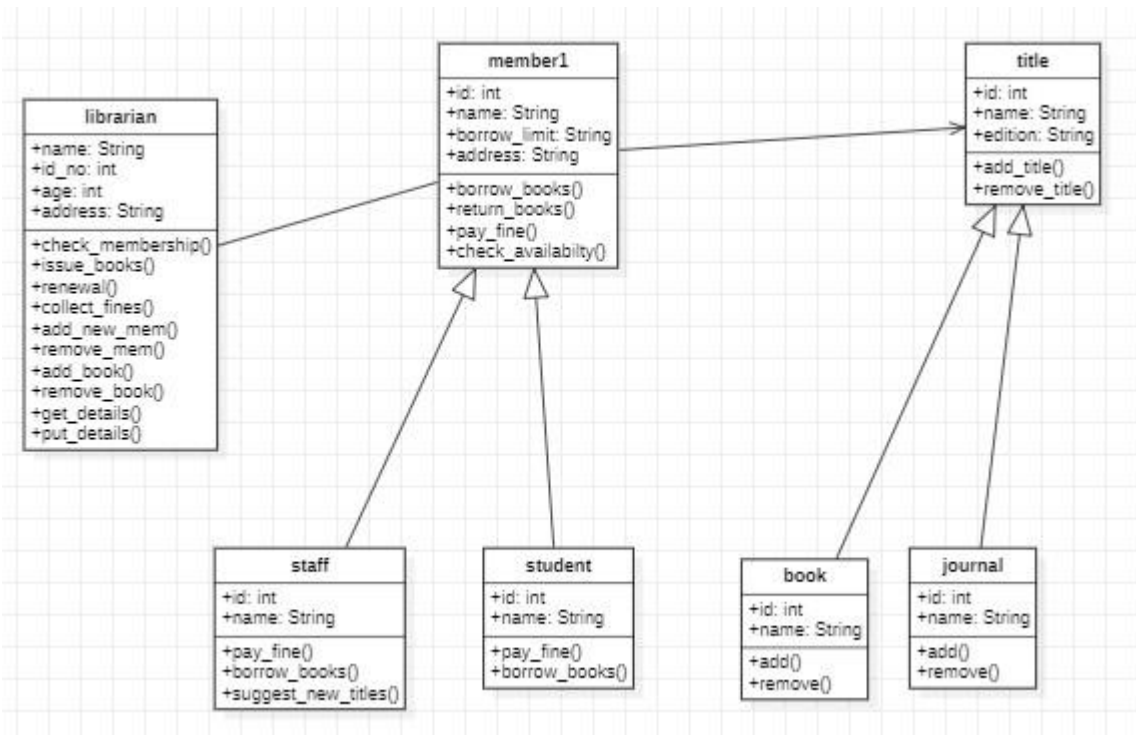
**Week-5 and Week-6: For each case study given earlier, Construct CLASS Diagram in the following manner:**

**CLASS DIAGRAMS: COLLEGE MANAGEMENT SYSTEM:**

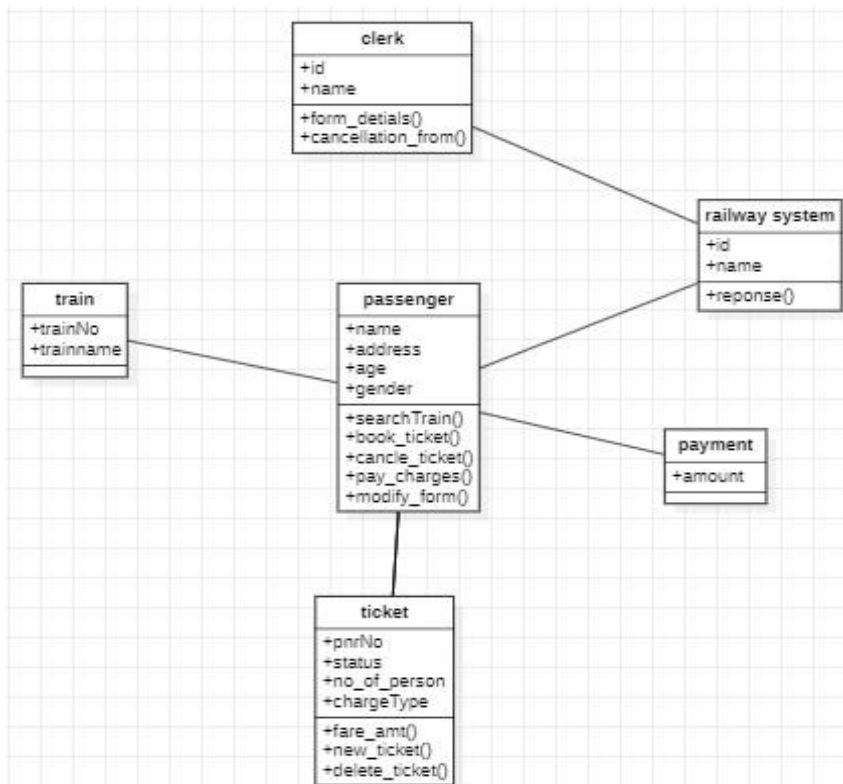


**LIBRARY MANAGEMENT SYSTEM:**

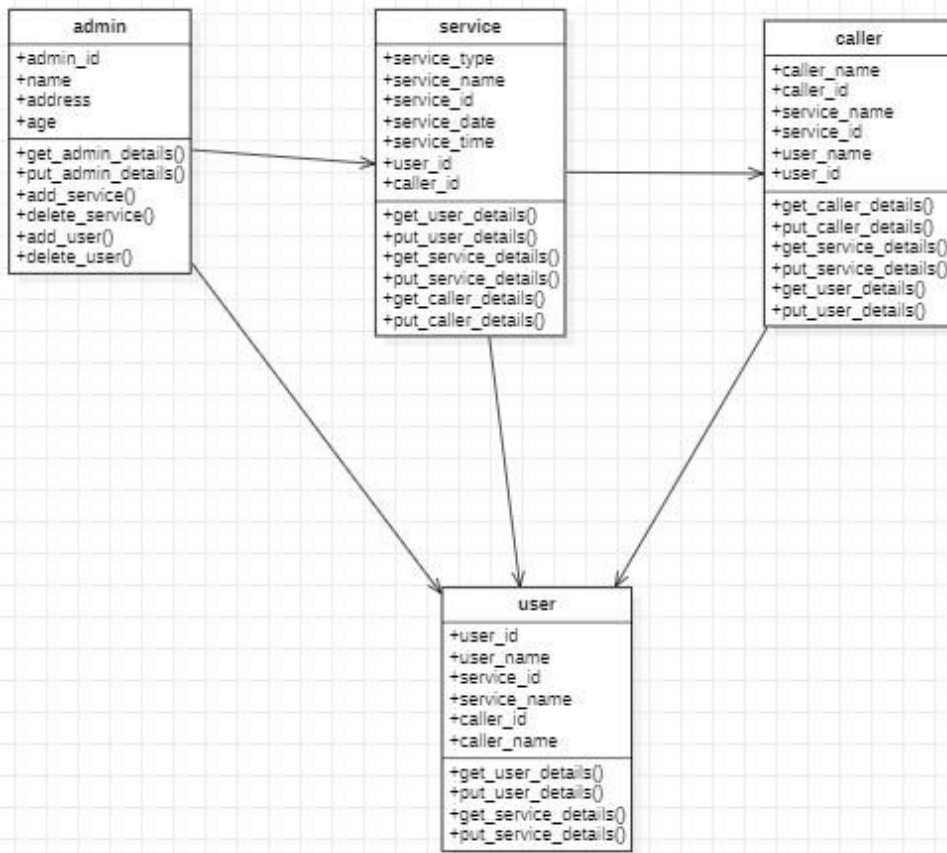




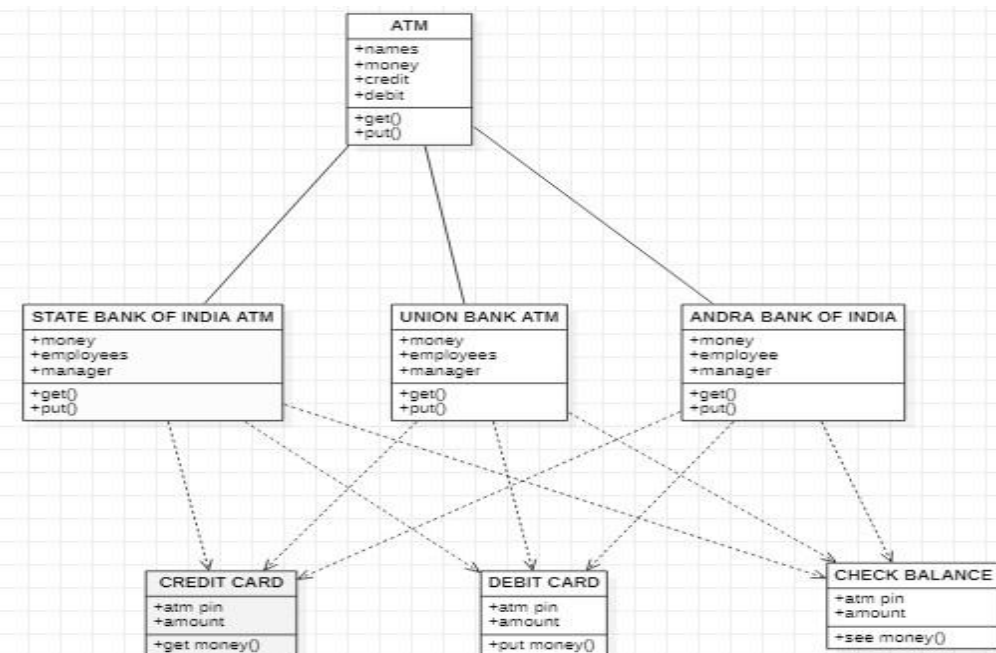
### RAILWAY RESERVATION SYSTEM:



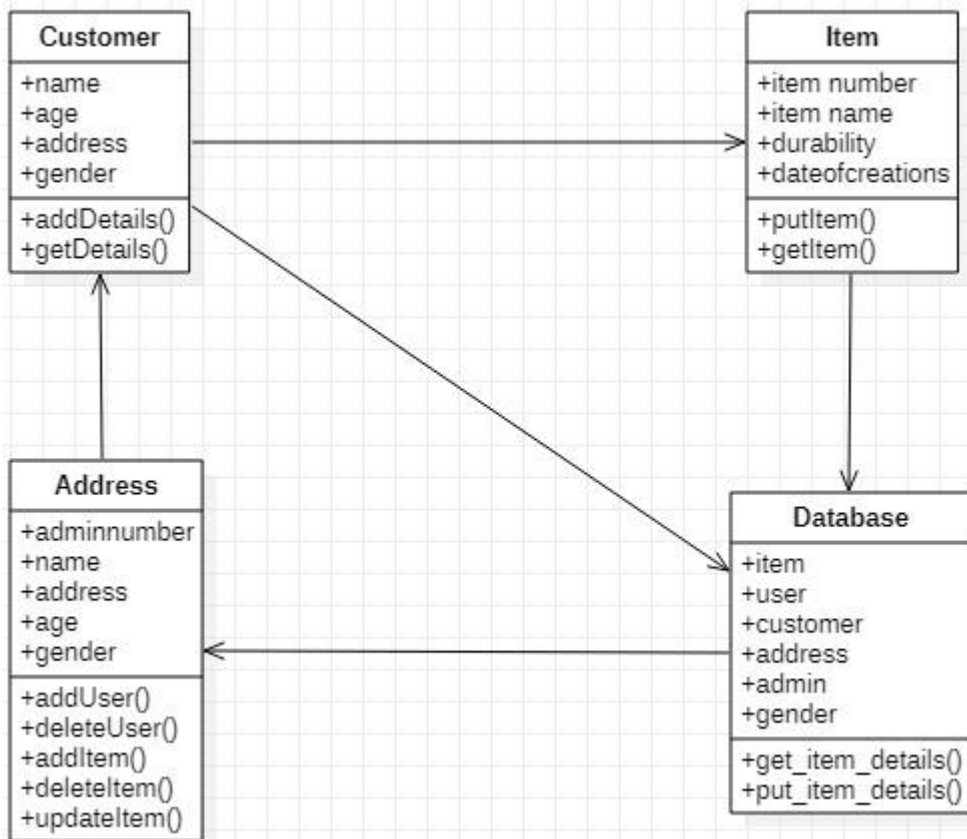
### CUSTOMER SUPPORT SYSTEM:



### ATM:



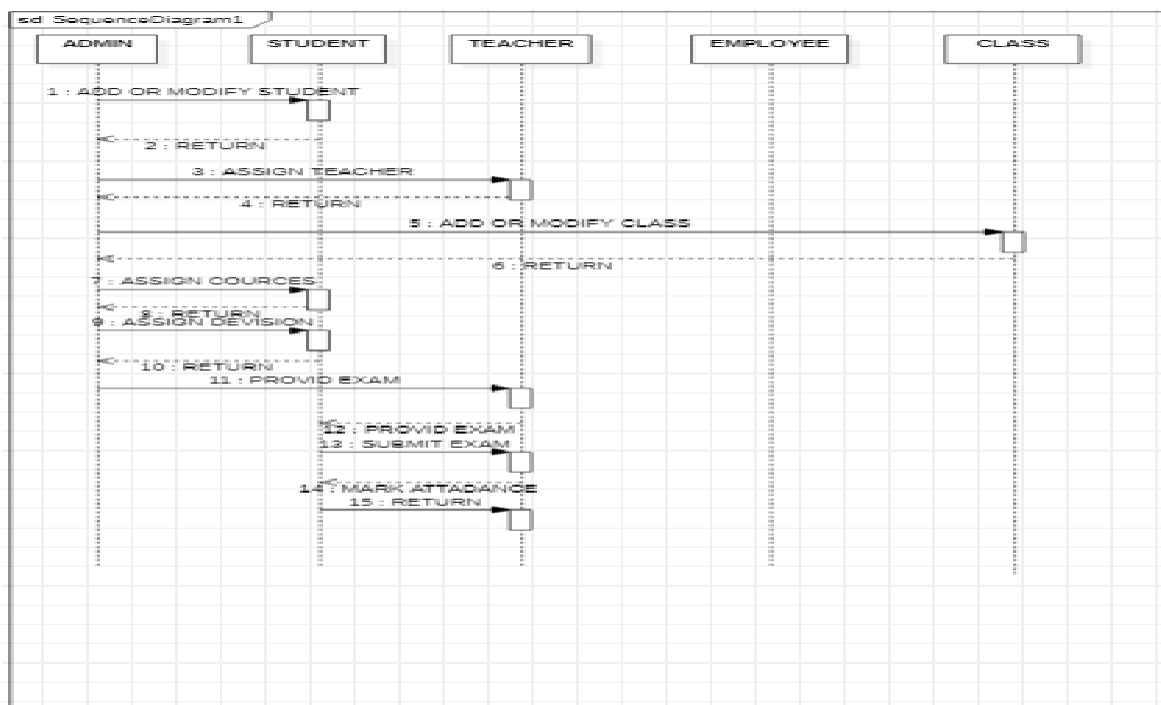
### POINTOFSALES:



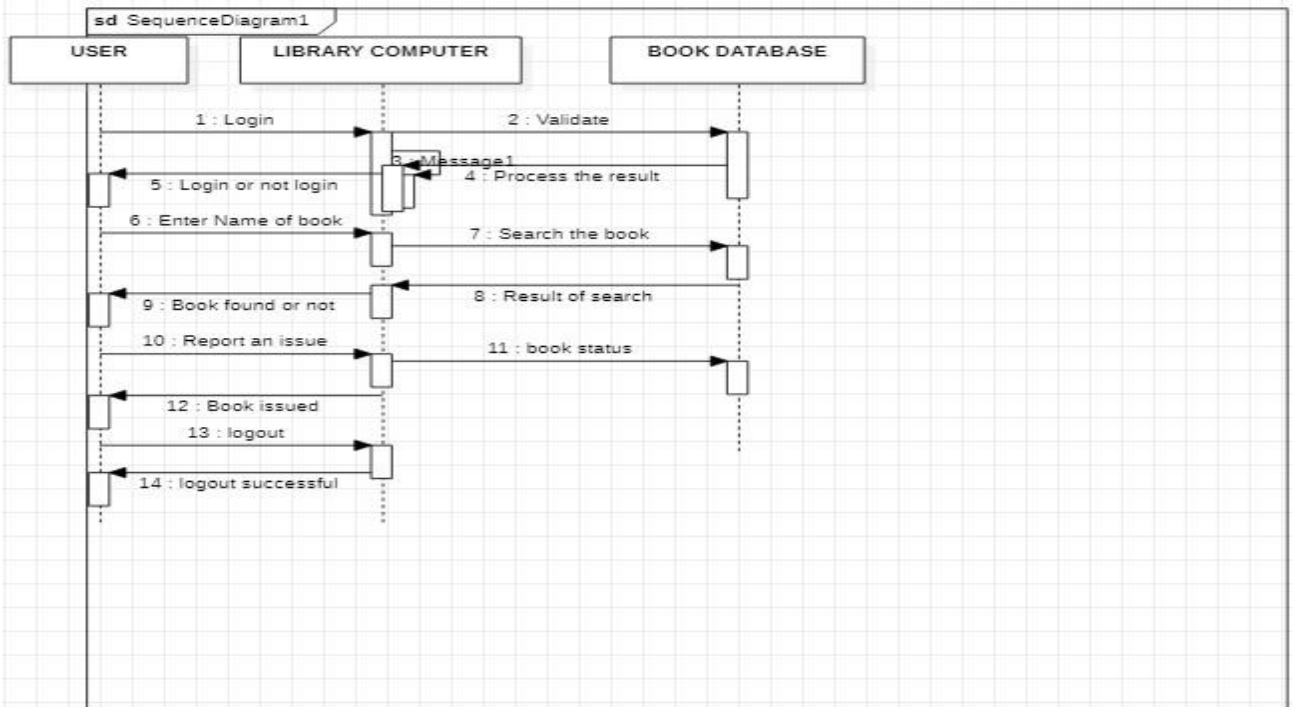
**Week-7: For each case study given earlier, Construct Interaction Diagrams in the following manner:**

### SEQUENCE DIAGRAMS: COLLEGE

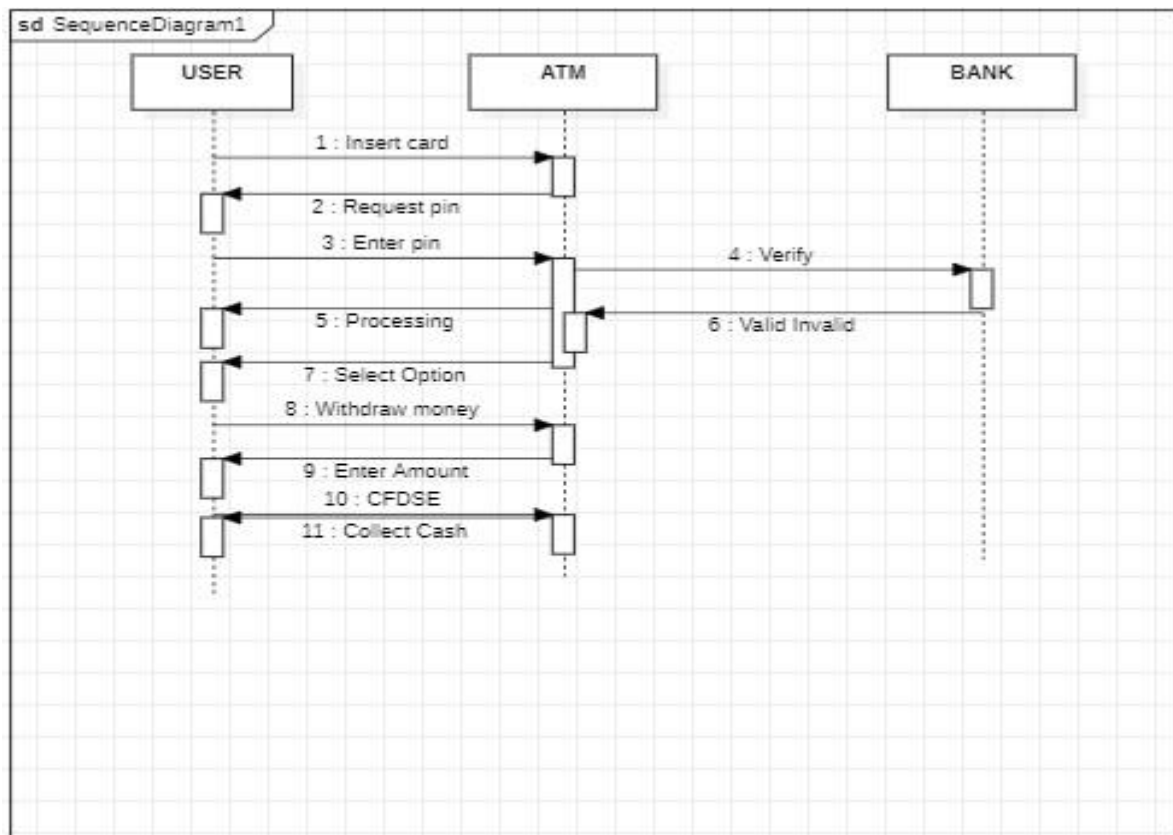
### MANAGEMENT SYSTEM:



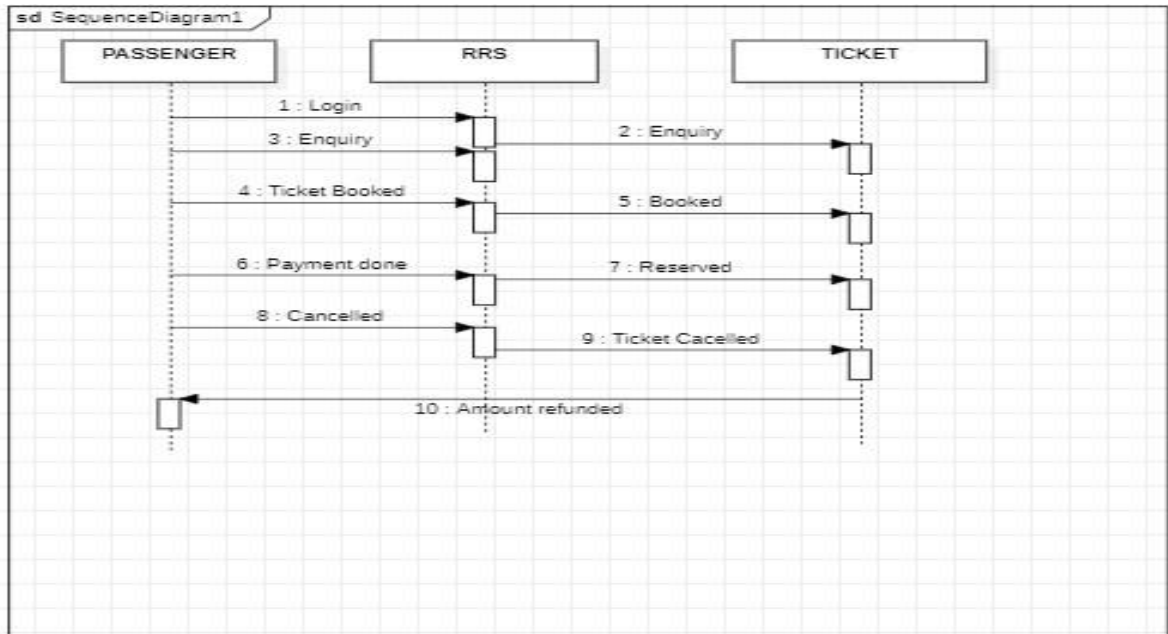
### LIBRARY MANAGEMENT SYSTEM:



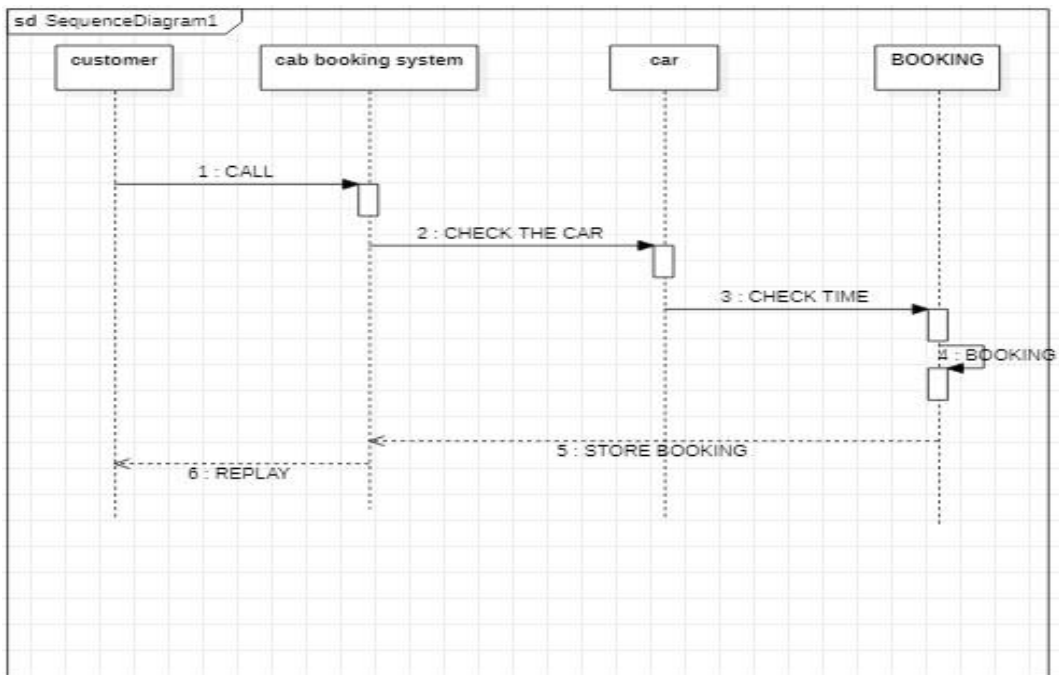
### ATM:



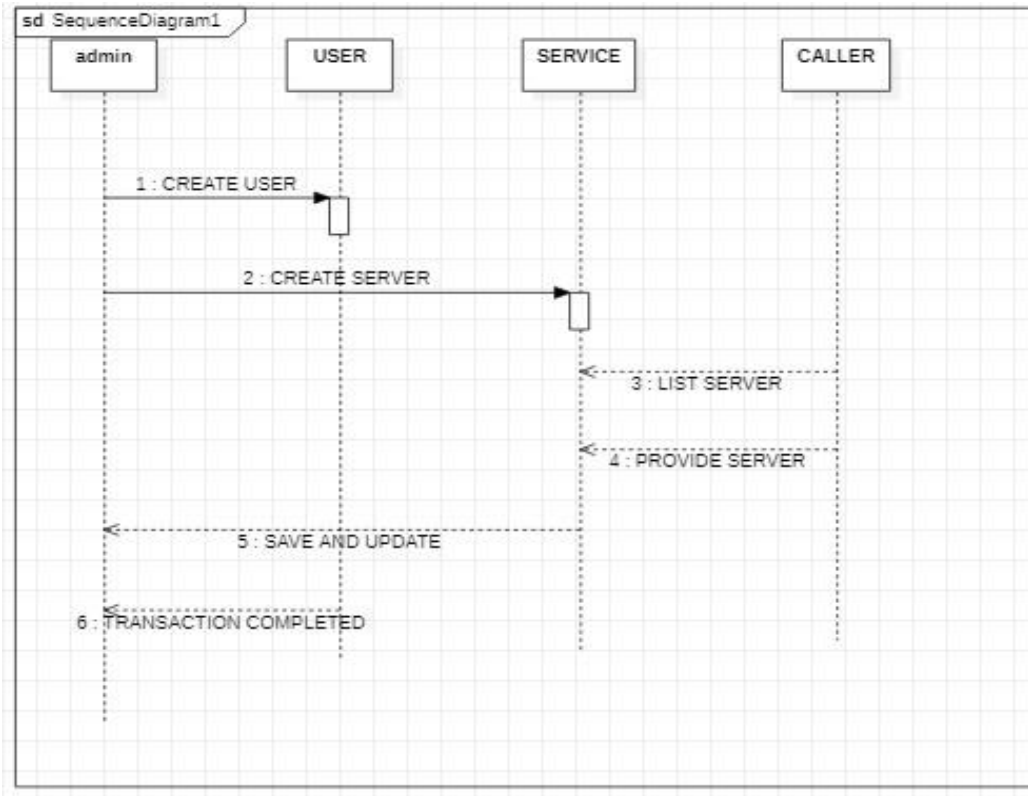
### RAILWAY RESERVATION SYSTEM:



### **CAB BOOKING SYSTEM:**

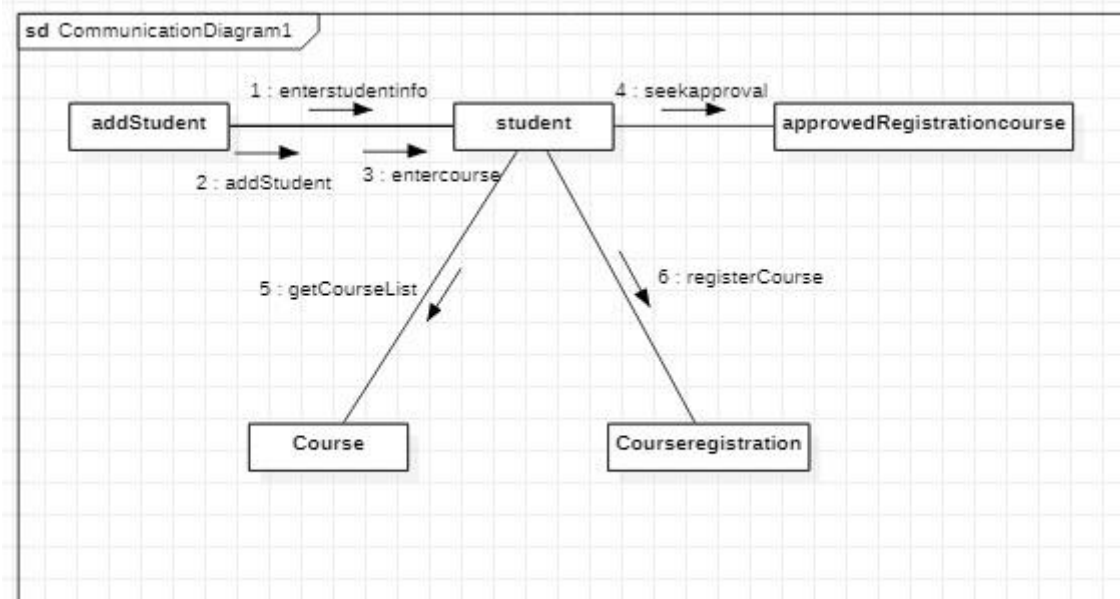


### **CUSTOMER SUPPORT SYSTEM:**

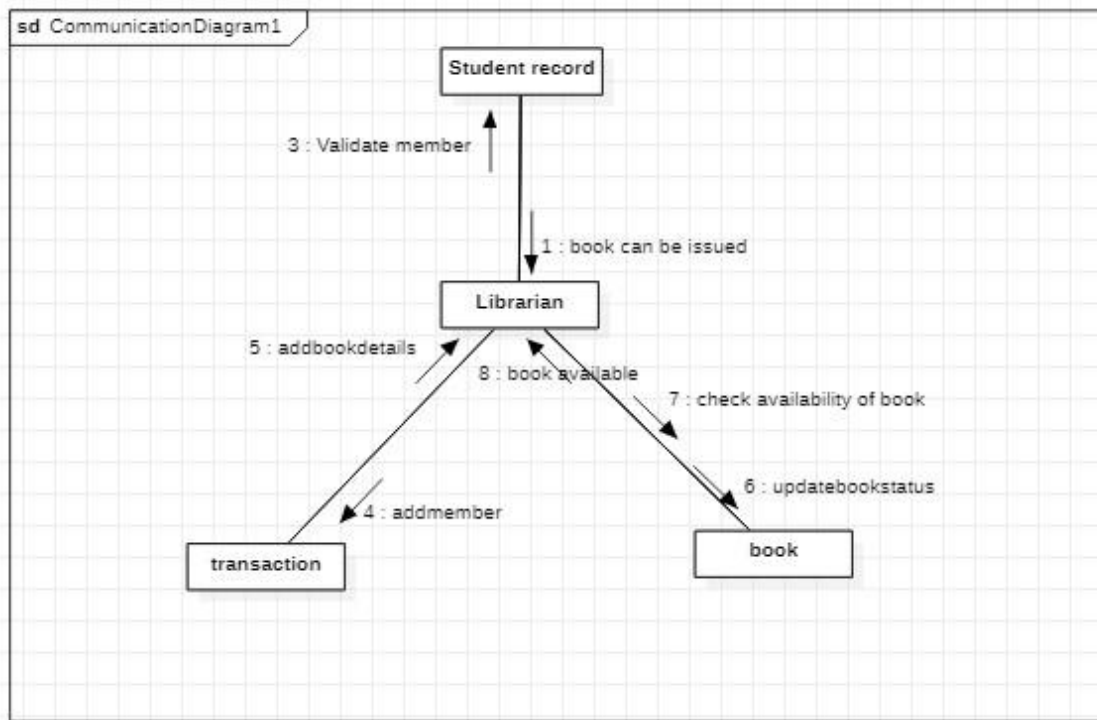


### COLLABORATION DIAGRAMS:

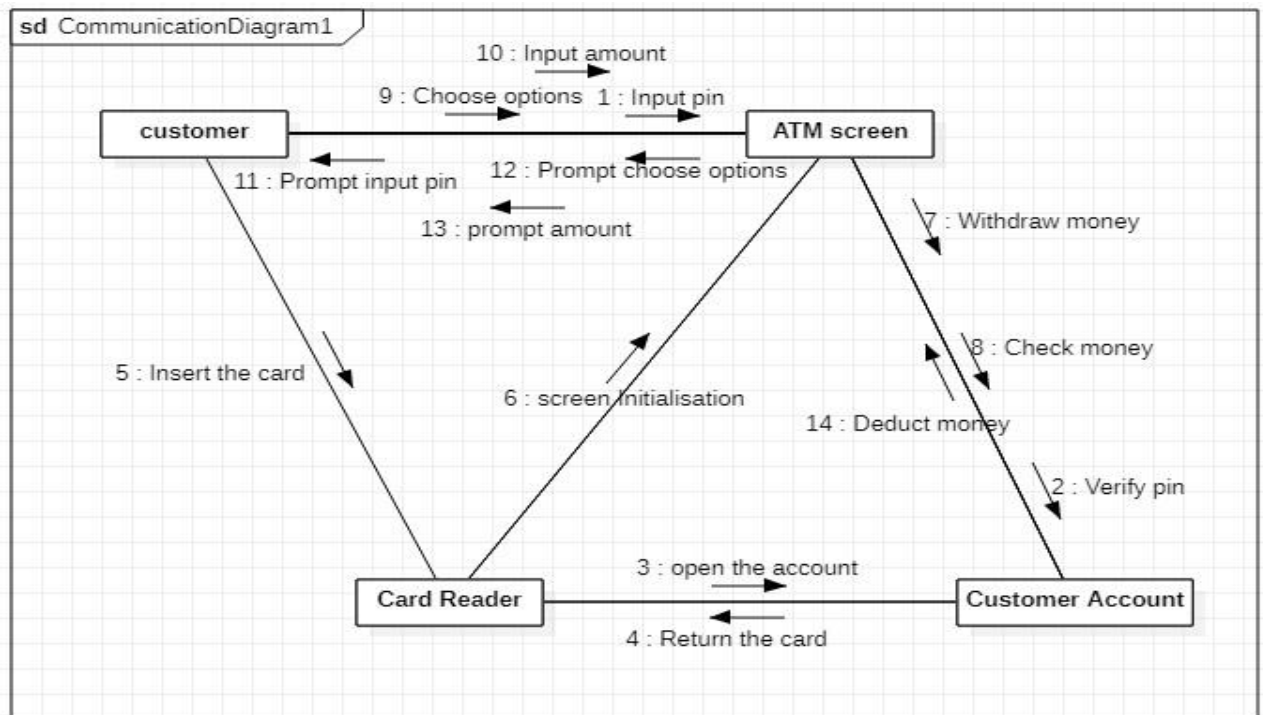
### COLLEGE MANAGEMENT SYSTEM:



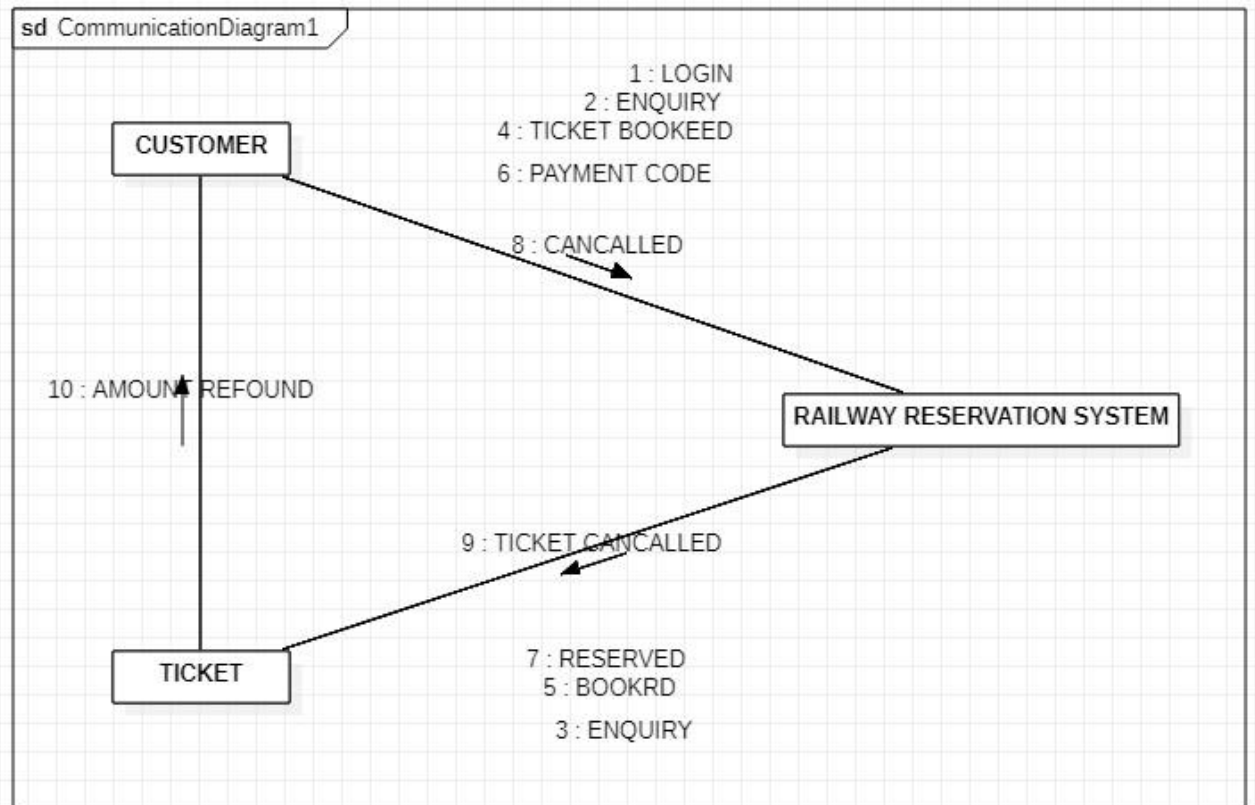
### LIBRARY MANAGEMENT SYSTEM:



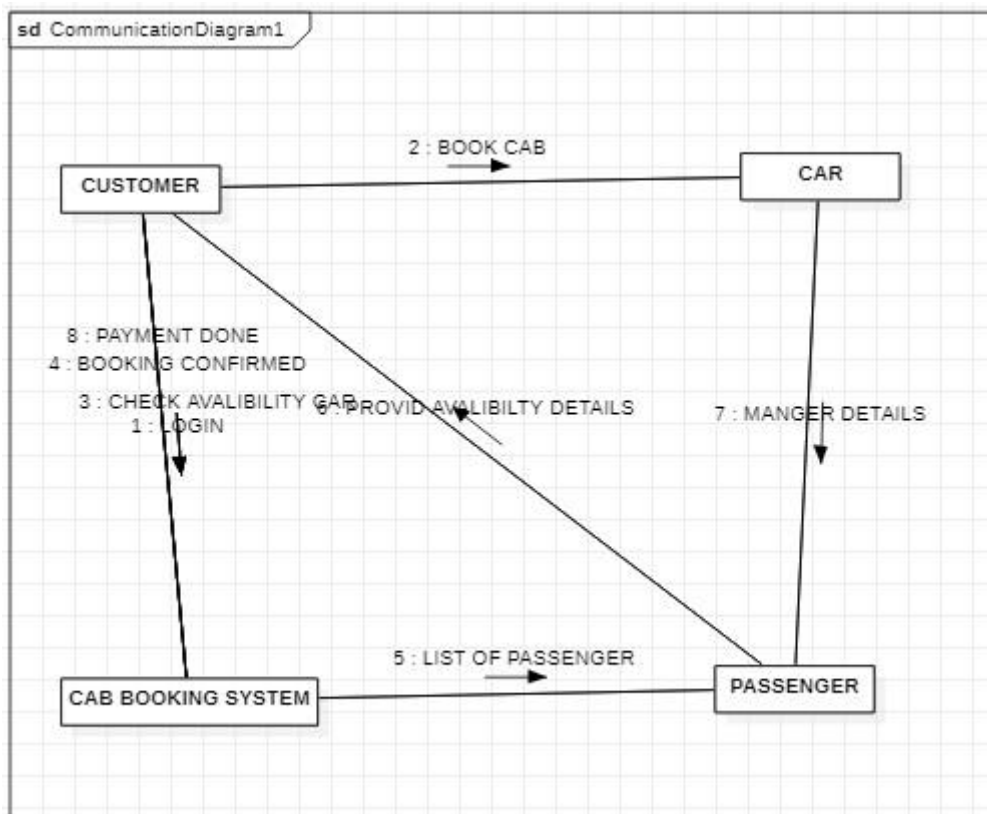
### ATM:



### RAILWAY RESERVATION SYSTEM:

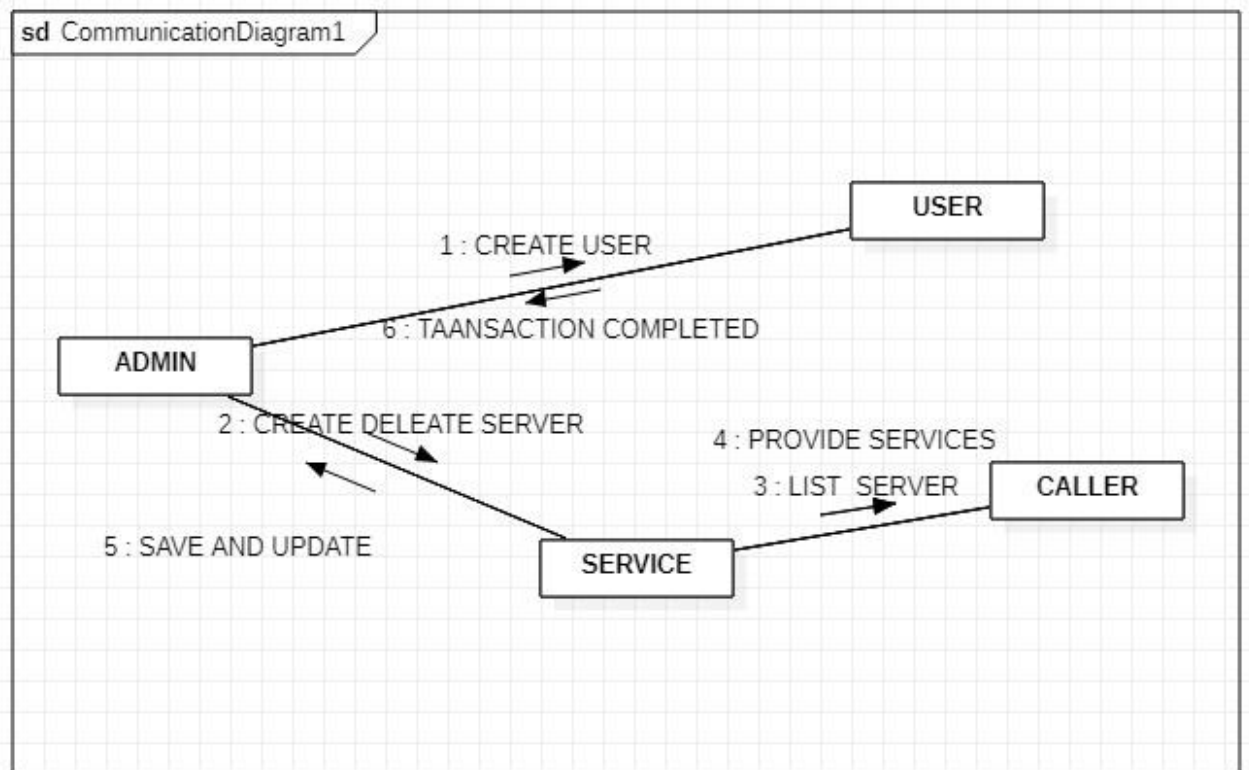


### **CAB BOOKING SYSTEM:**



### **CUSTOMER SUPPORT SYSTEM:**

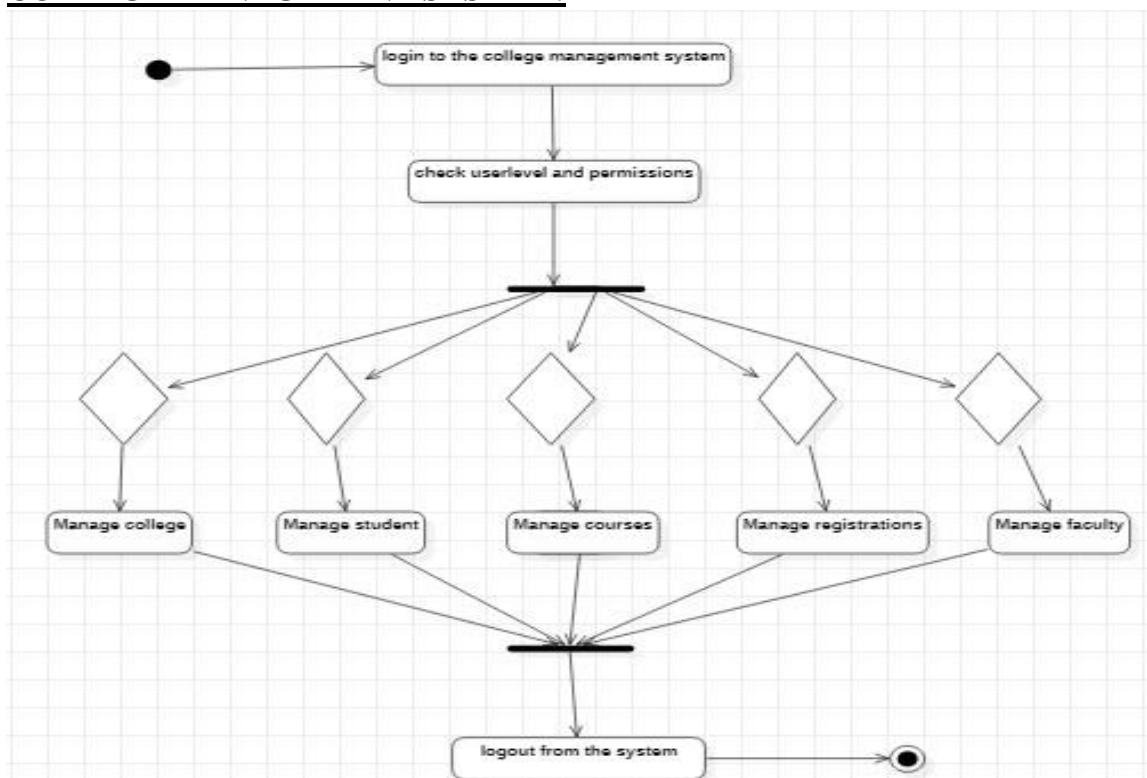




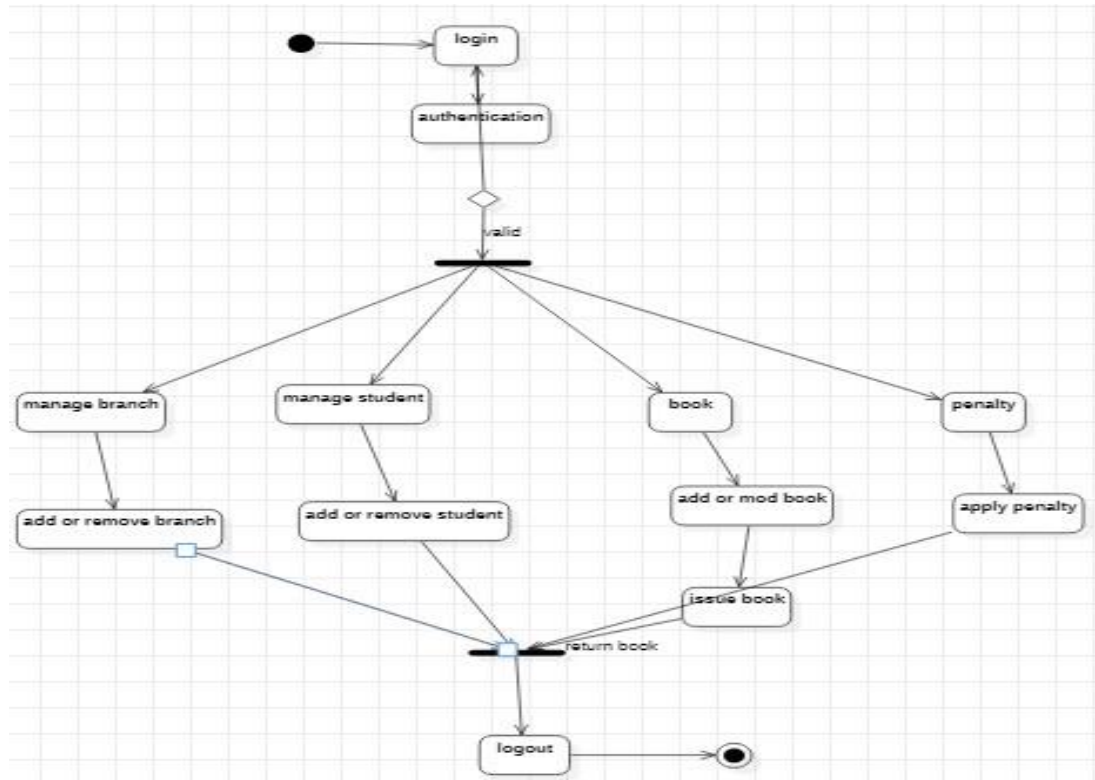
**Week-8: For each case study given earlier, Construct Activity Diagram in the following manner:**

### ACTIVITY DIAGRAMS:

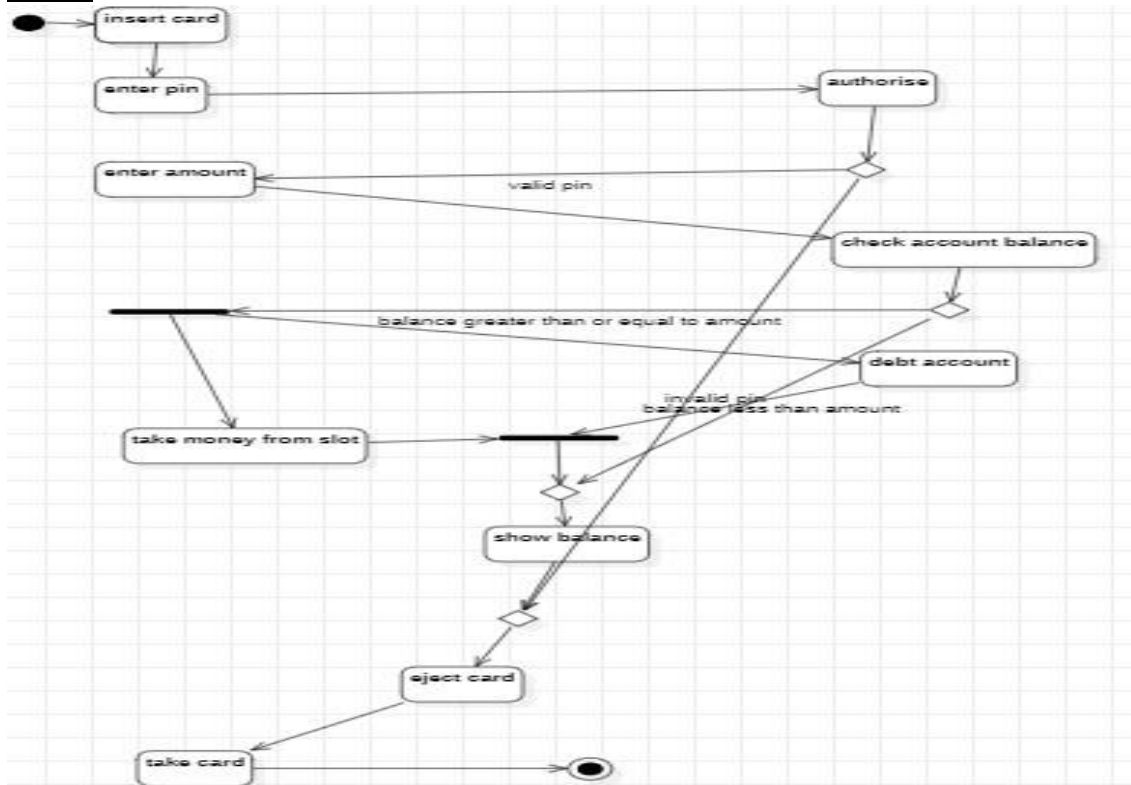
#### COLLEGE MANAGEMENT SYSTEM:



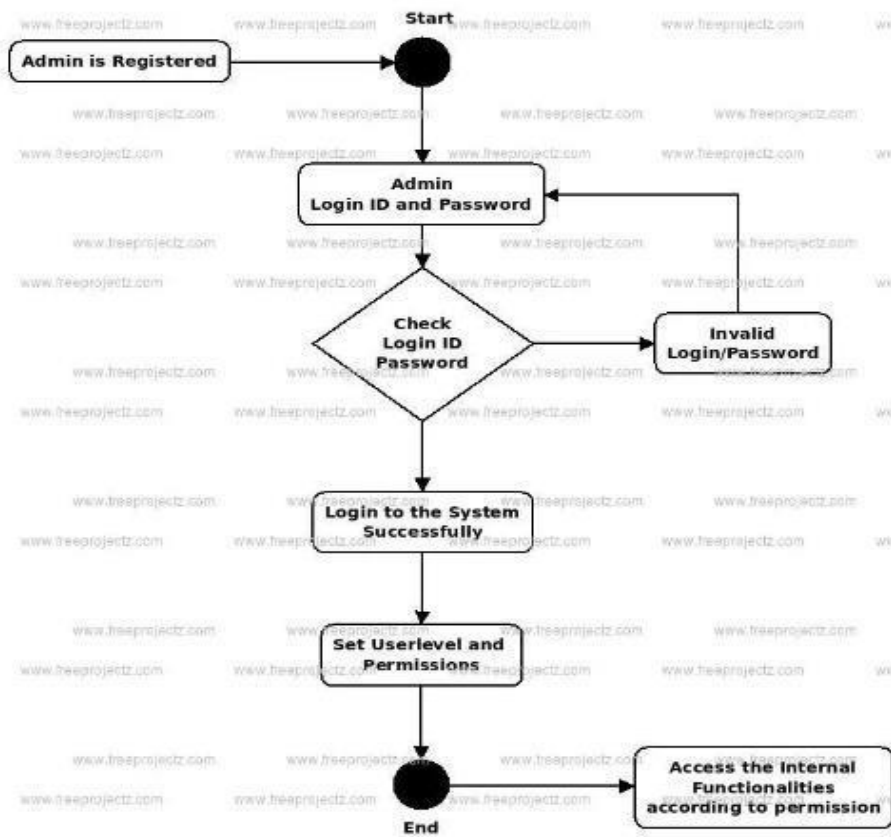
#### LIBRARY MANAGEMENT SYSTEM:



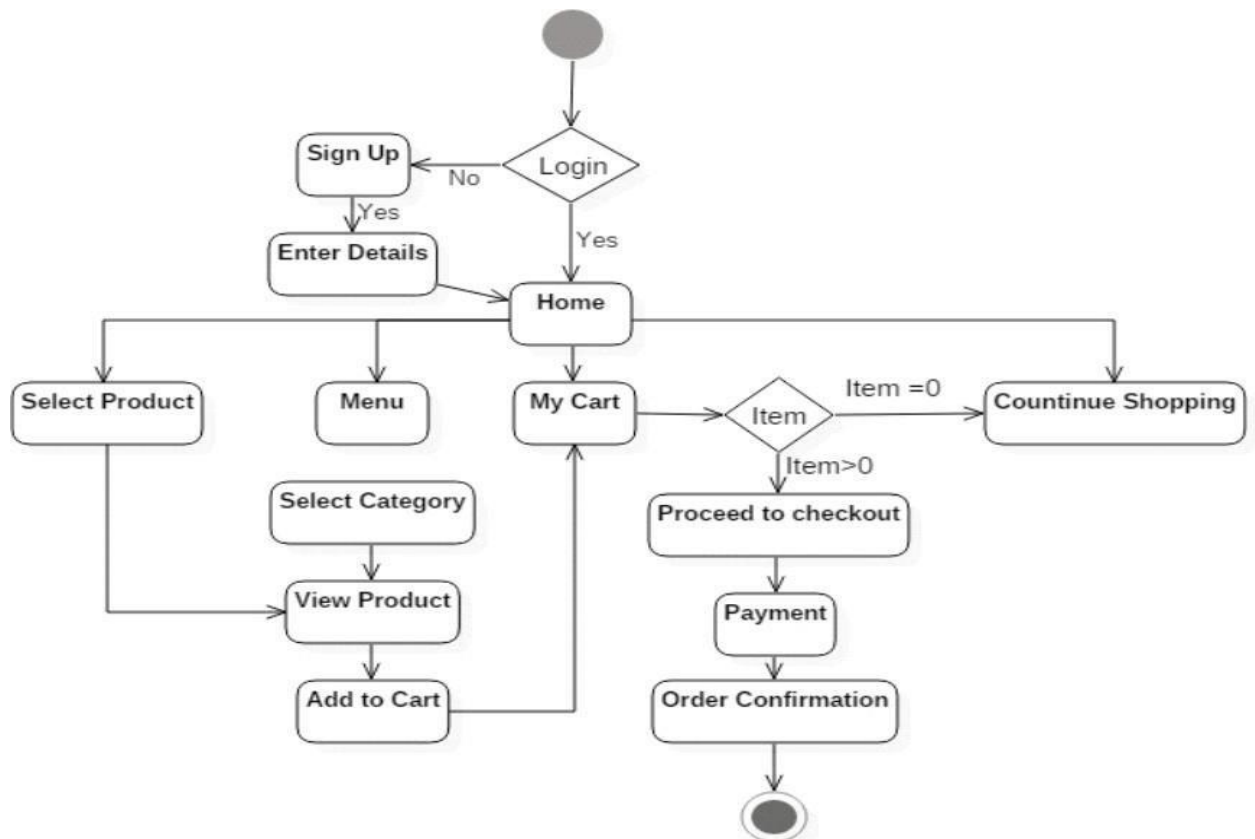
### ATM:



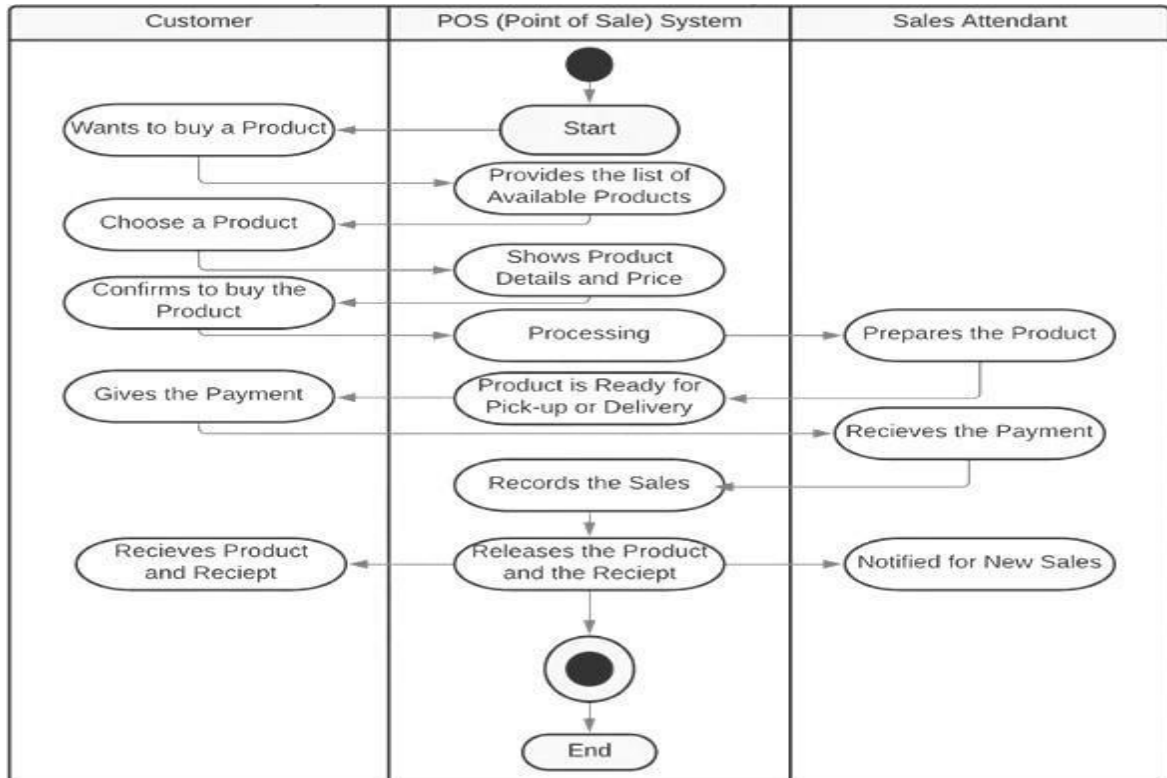
### RAILWAY RESERVATION SYSTEM:



### CUSTOMER SUPPORT SYSTEM:



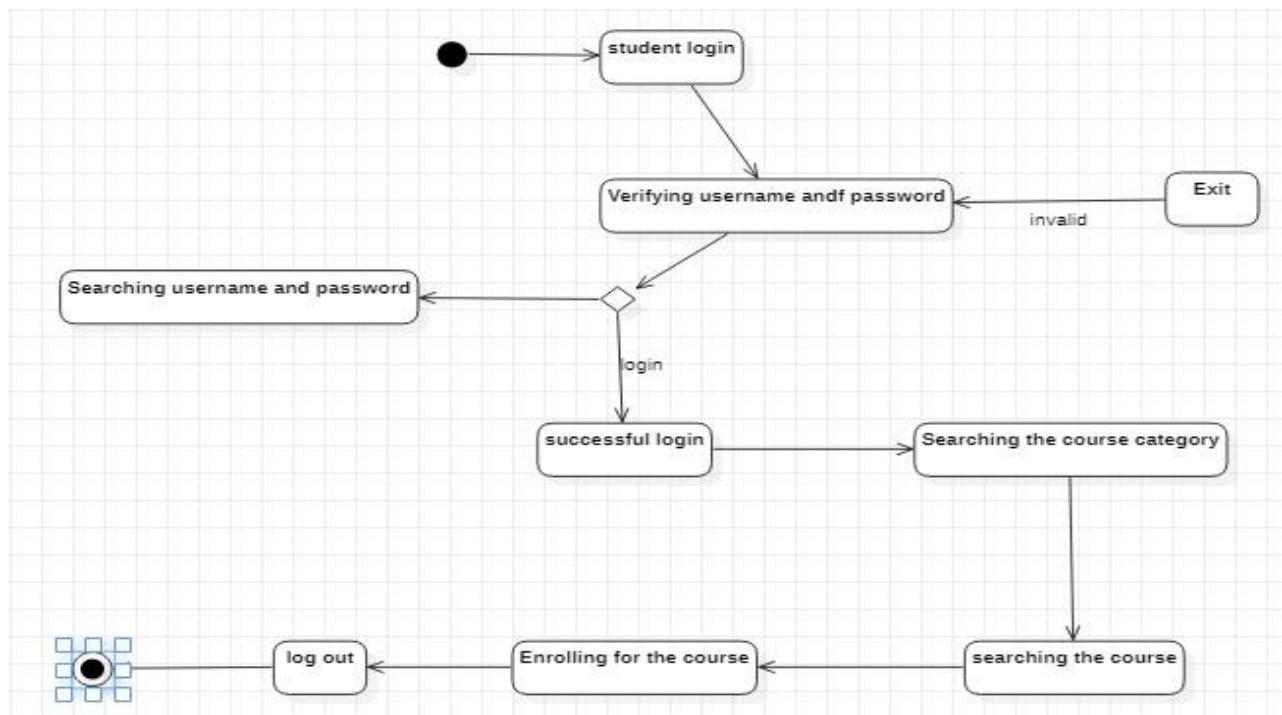
### POINT OF SALES:



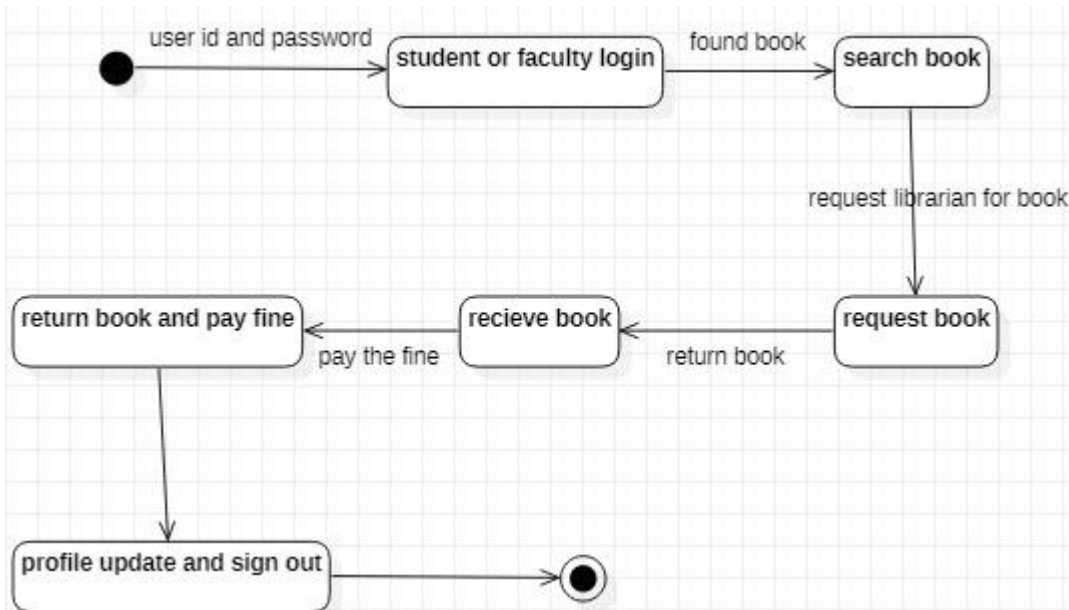
**Week-9: For each case study given earlier, Construct State Chart Diagram in the following manner:**

### STATE CHART DIAGRAMS:

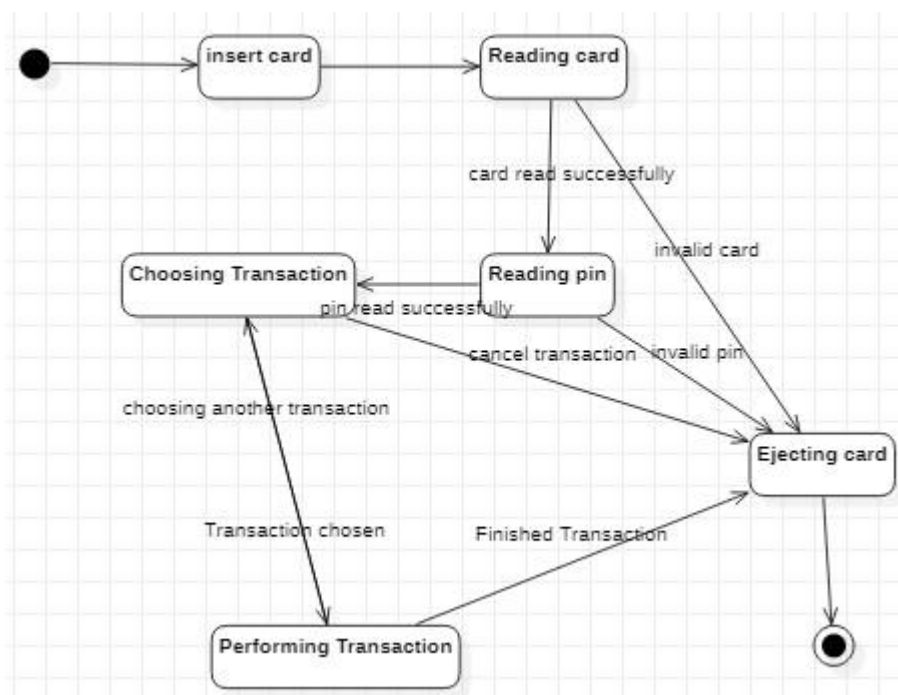
### COLLEGE MANAGEMENT SYSTEM:



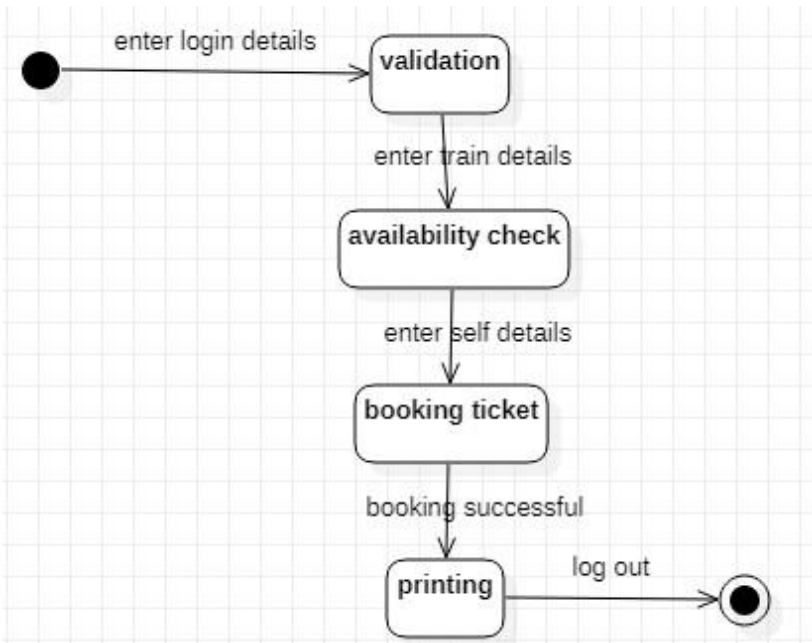
### LIBRARY MANAGEMENT SYSTEM:



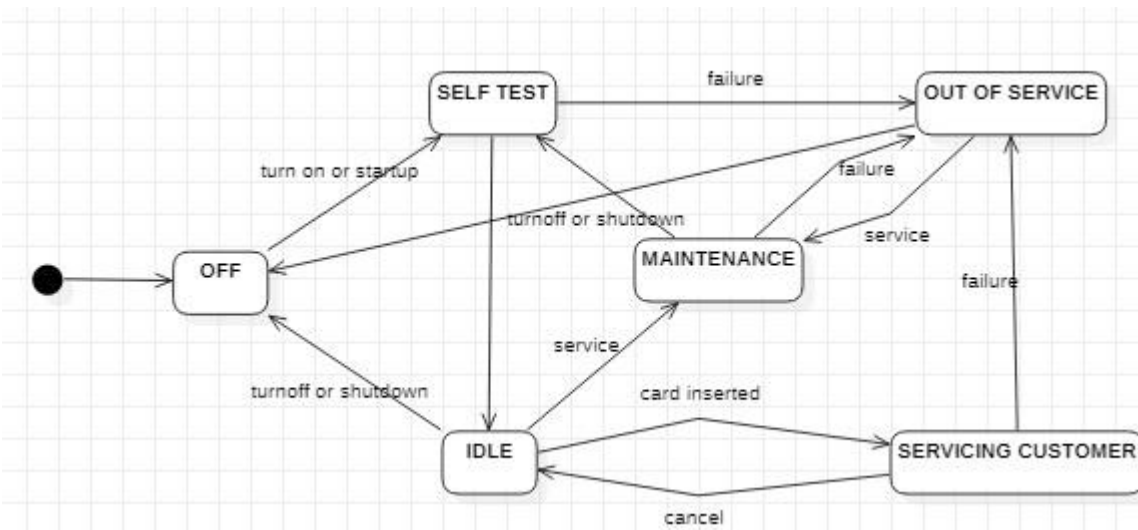
### ATM:



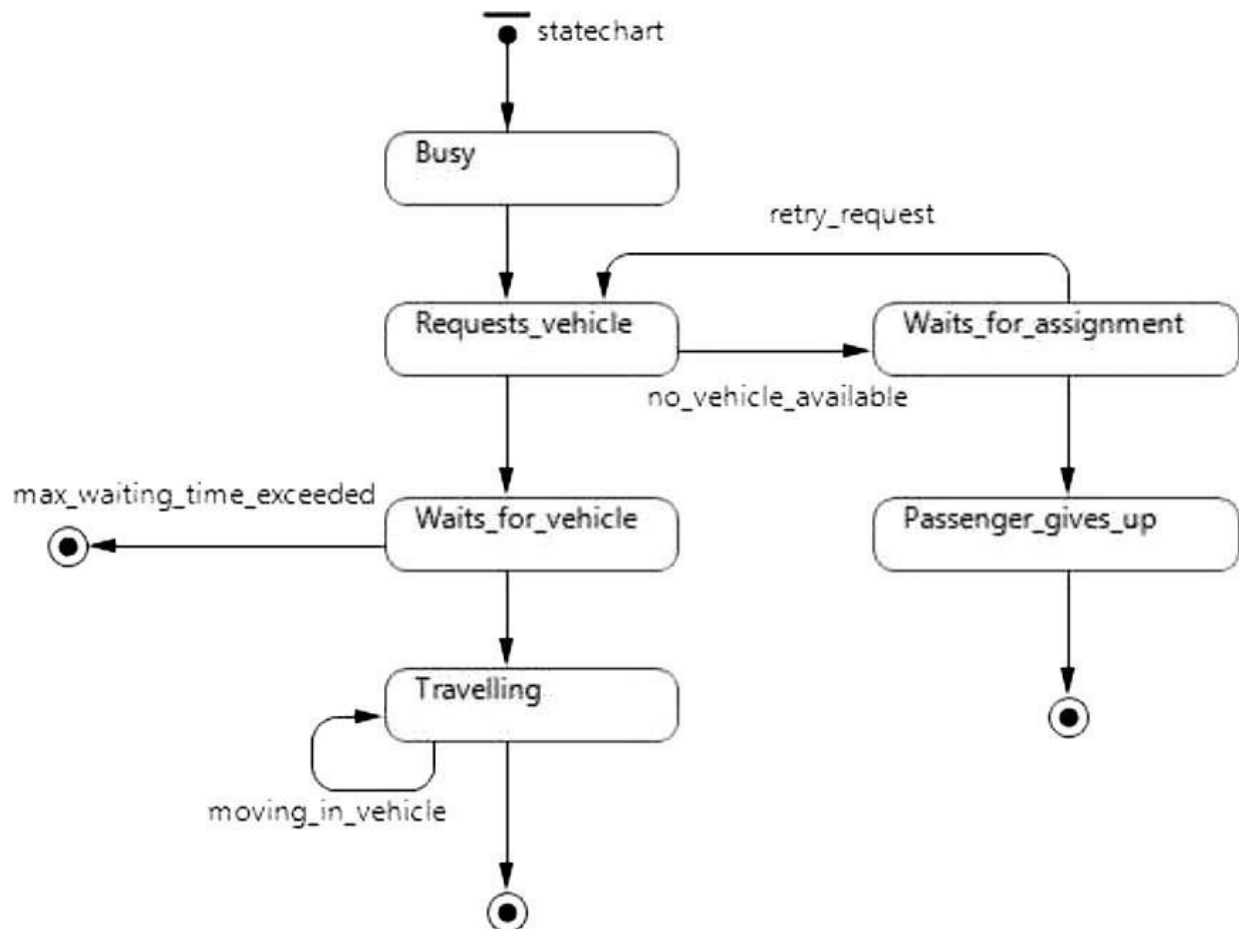
### RAILWAY RESERVATION SYSTEM:



### **CUSTOMER SUPPORT SYSTEM:**



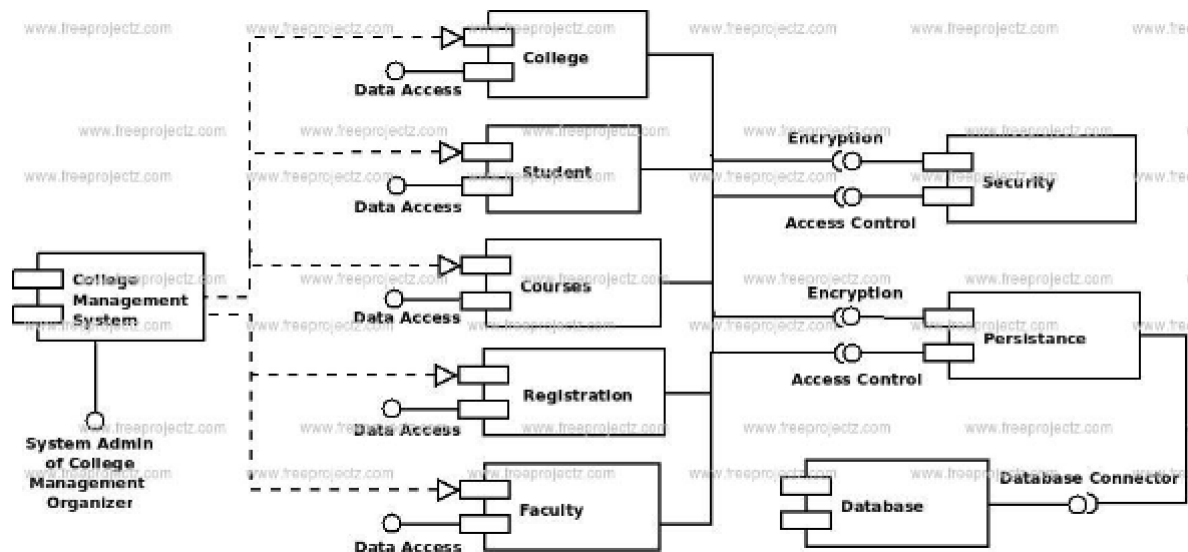
### **UBER CAB SYSTEM:**



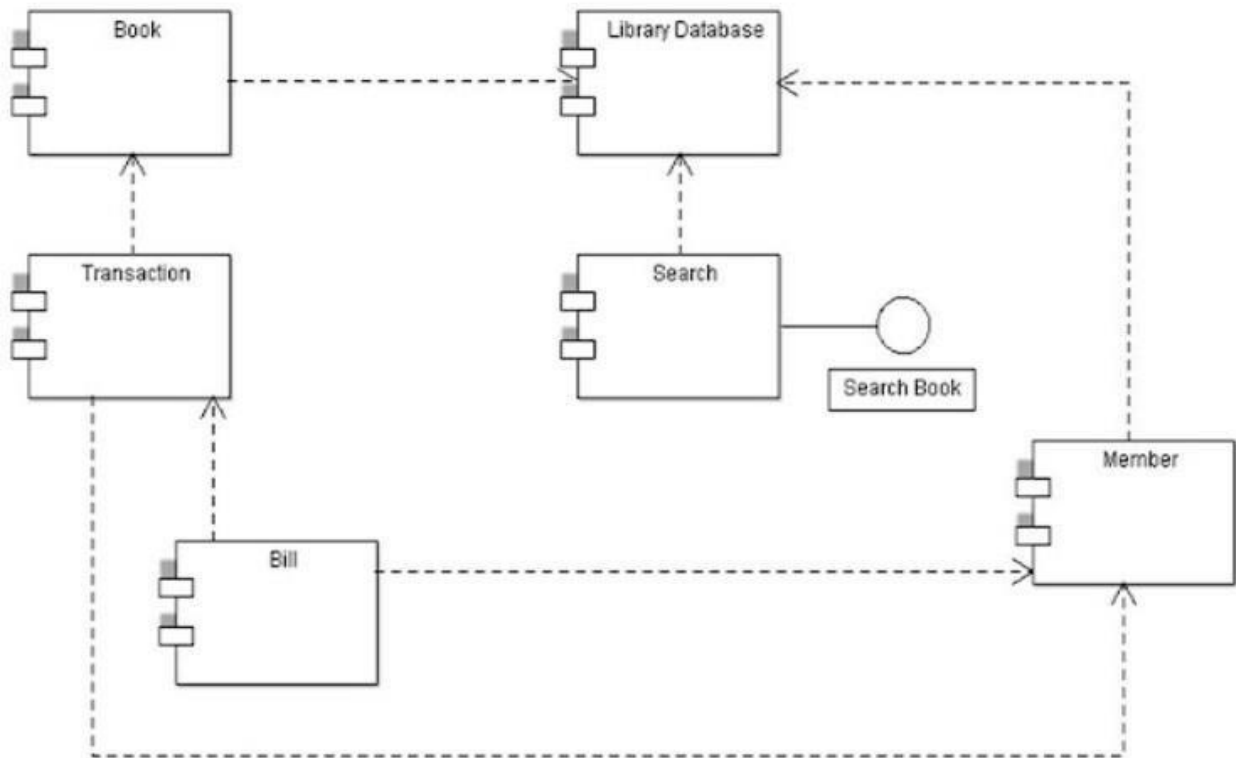
**Week-10: For each case study given earlier, Construct Component Diagram in the following manner:**

### COMPONENT DIAGRAMS:

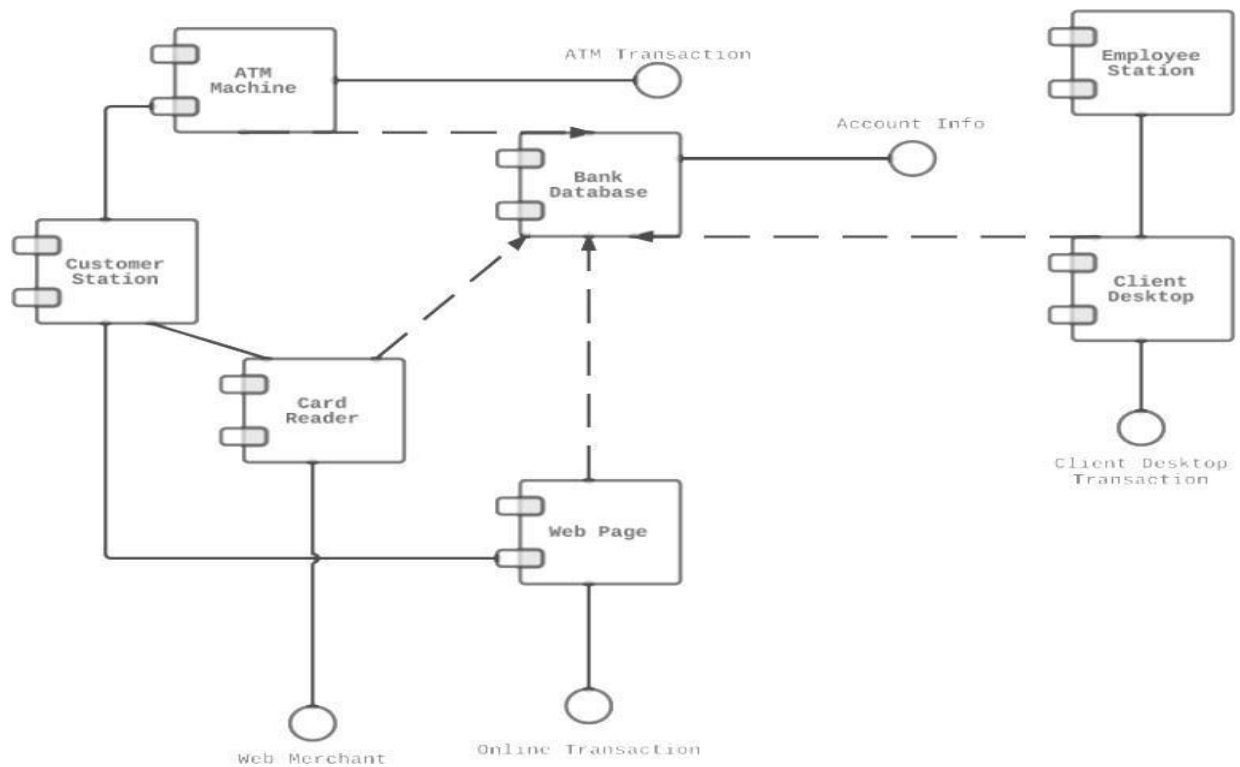
#### COLLEGE MANAGEMENT SYSTEM:



#### LIBRARY MANAGEMENT SYSTEM:

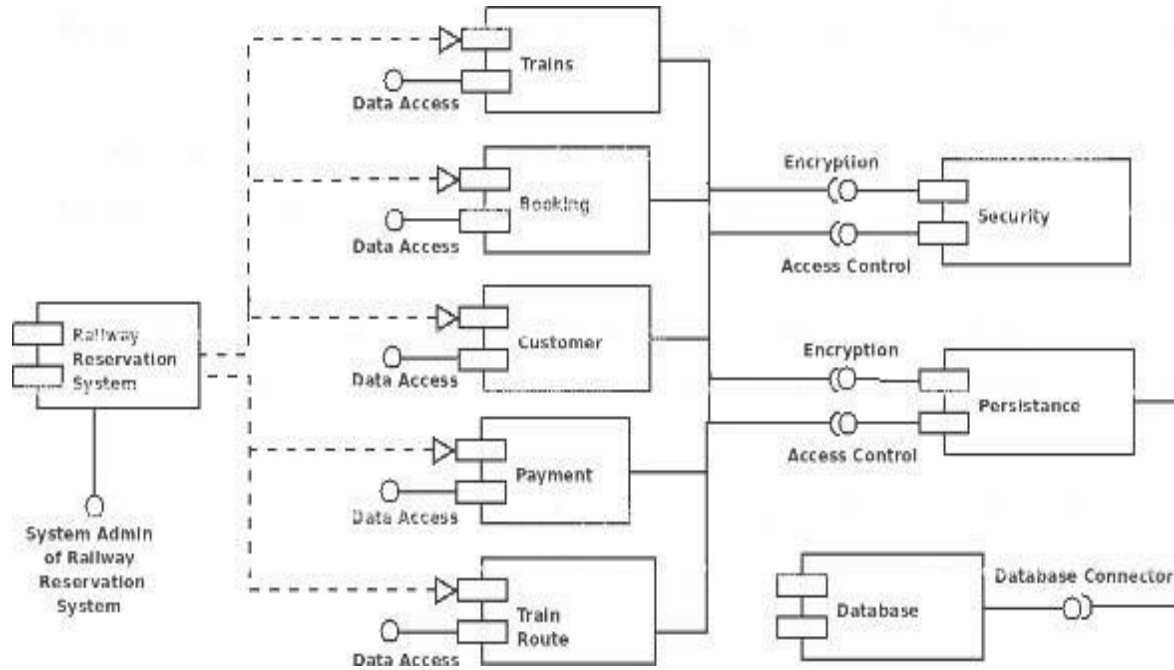


### ATM:

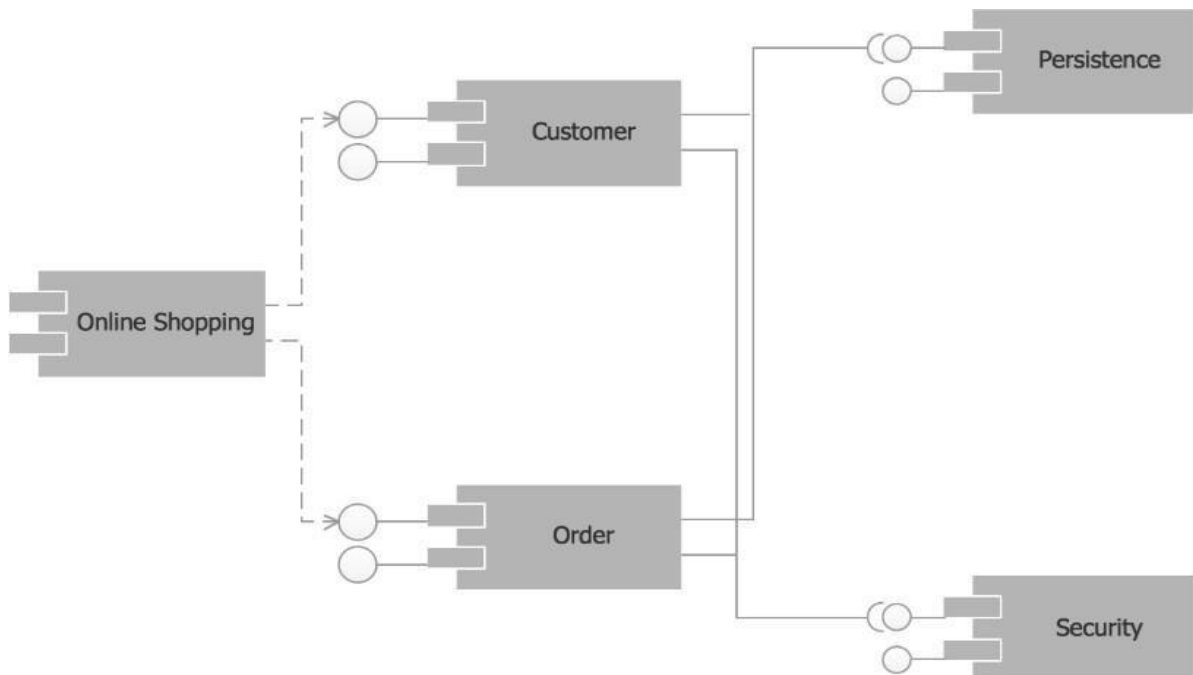


### RAILWAY RESERVATION SYSTEM:

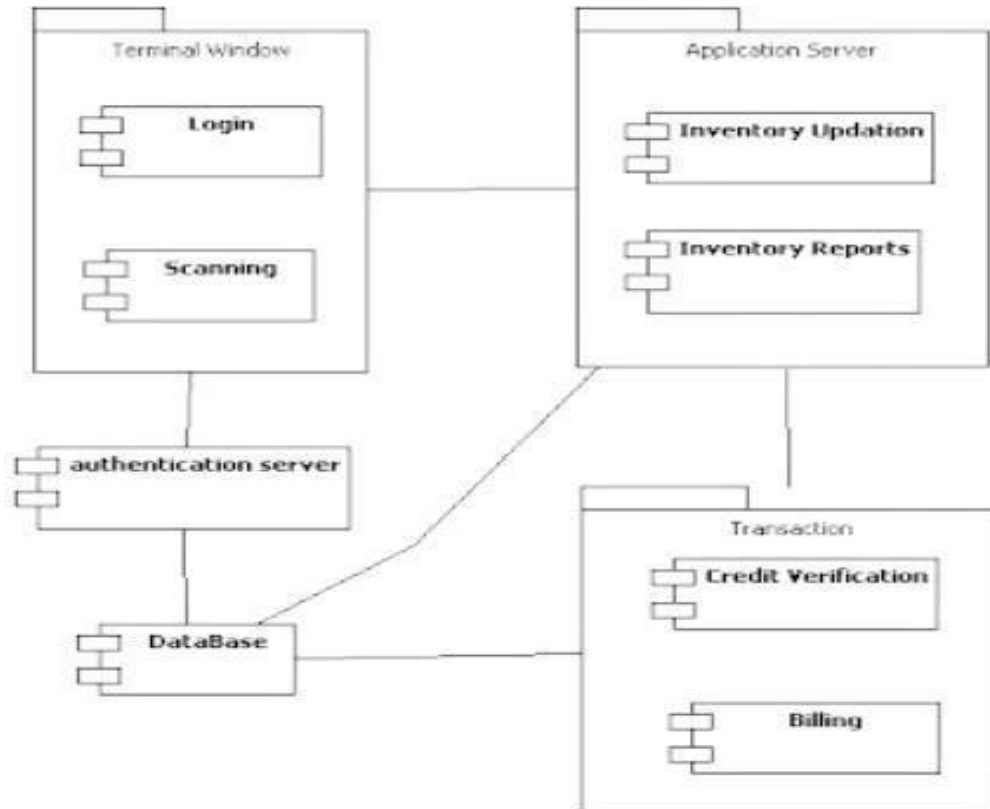




### CUSTOMER SUPPORT SYSTEM:



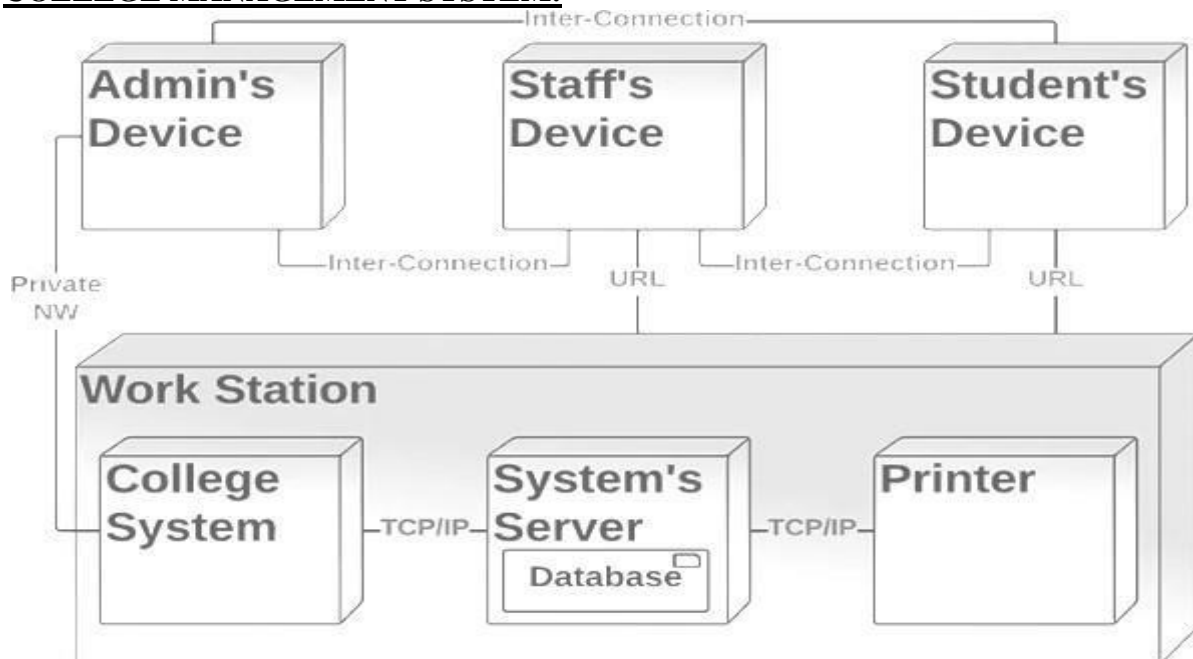
### POINT OF SALES:



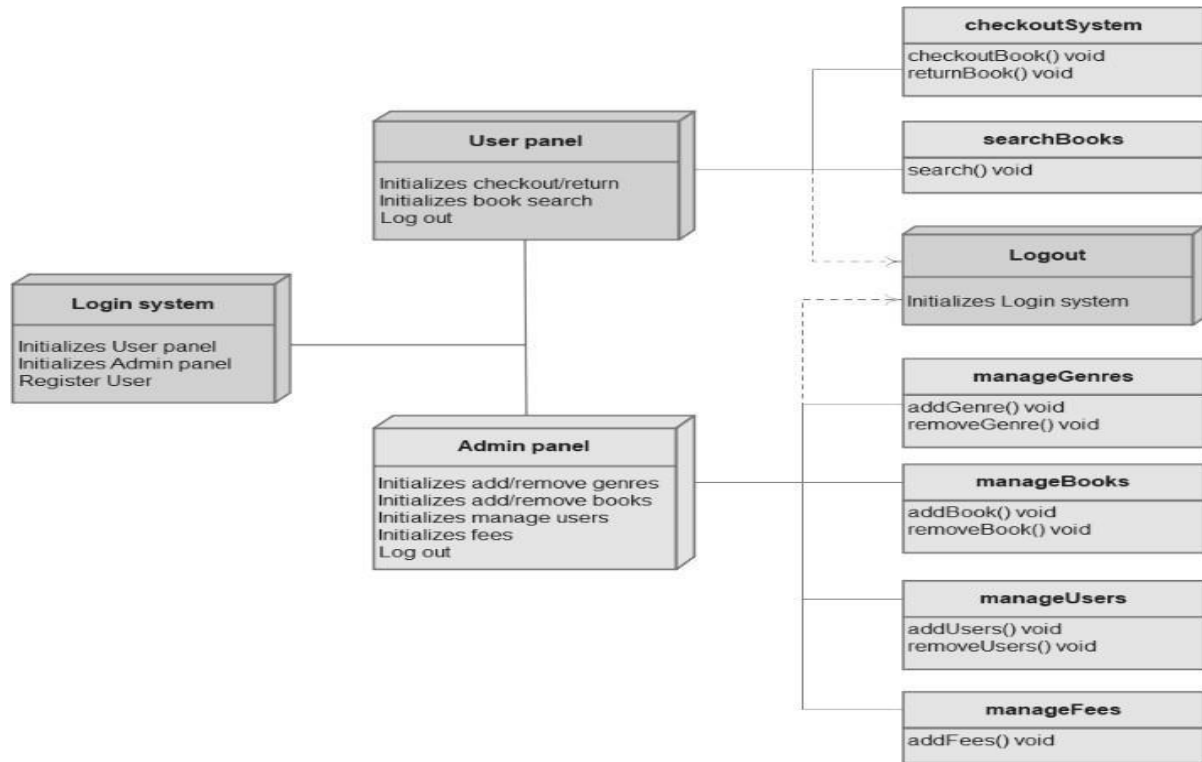
**Week-11: For each case study given earlier, construct Deployment Diagram in the following manner:**

### **DEPLOYMENT DIAGRAMS:**

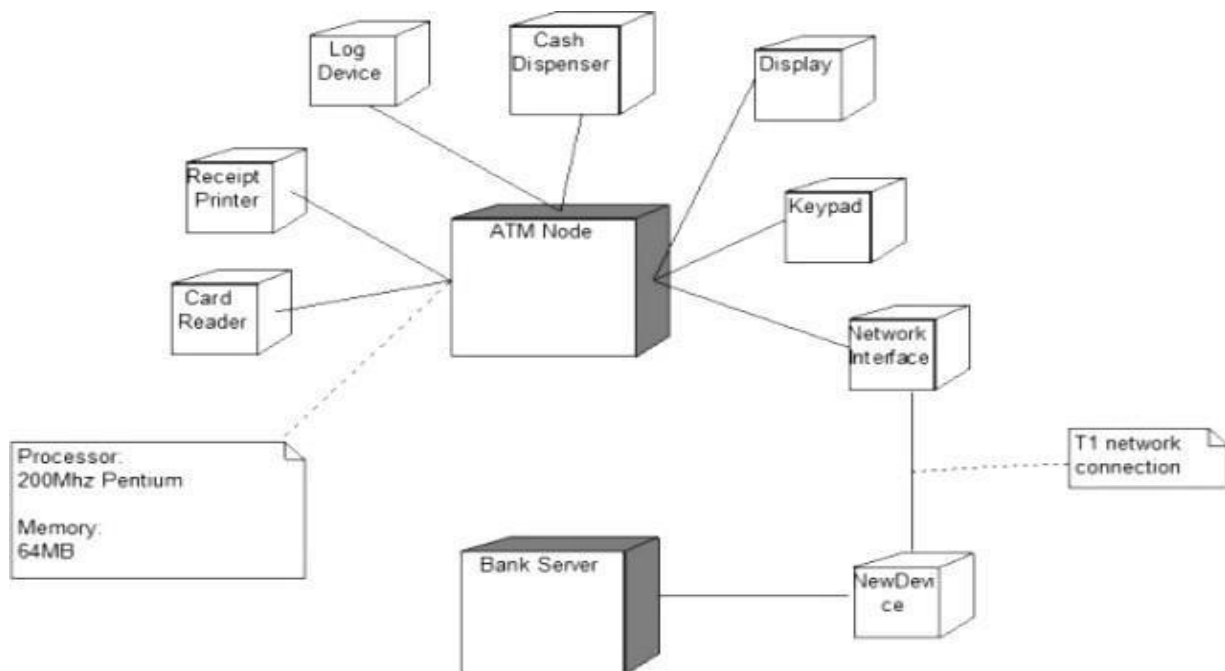
#### **COLLEGE MANAGEMENT SYSTEM:**



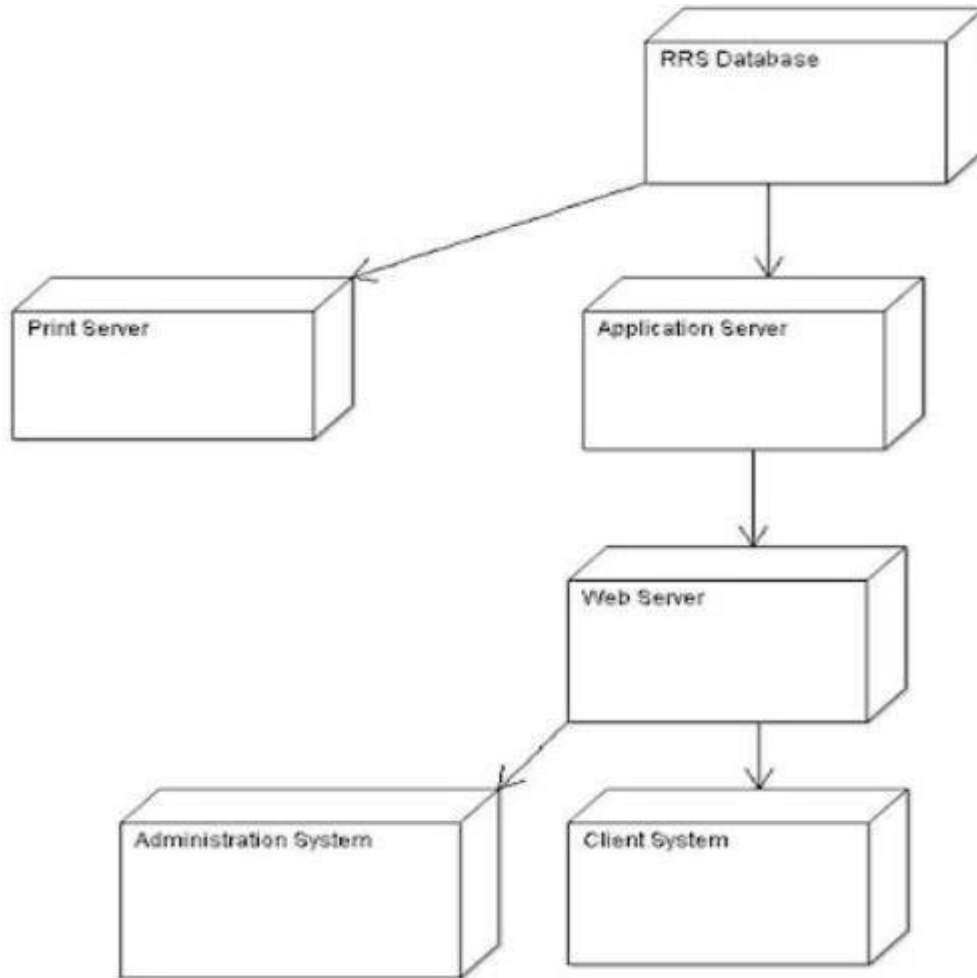
#### **LIBRARY MANAGEMENT SYSTEM:**



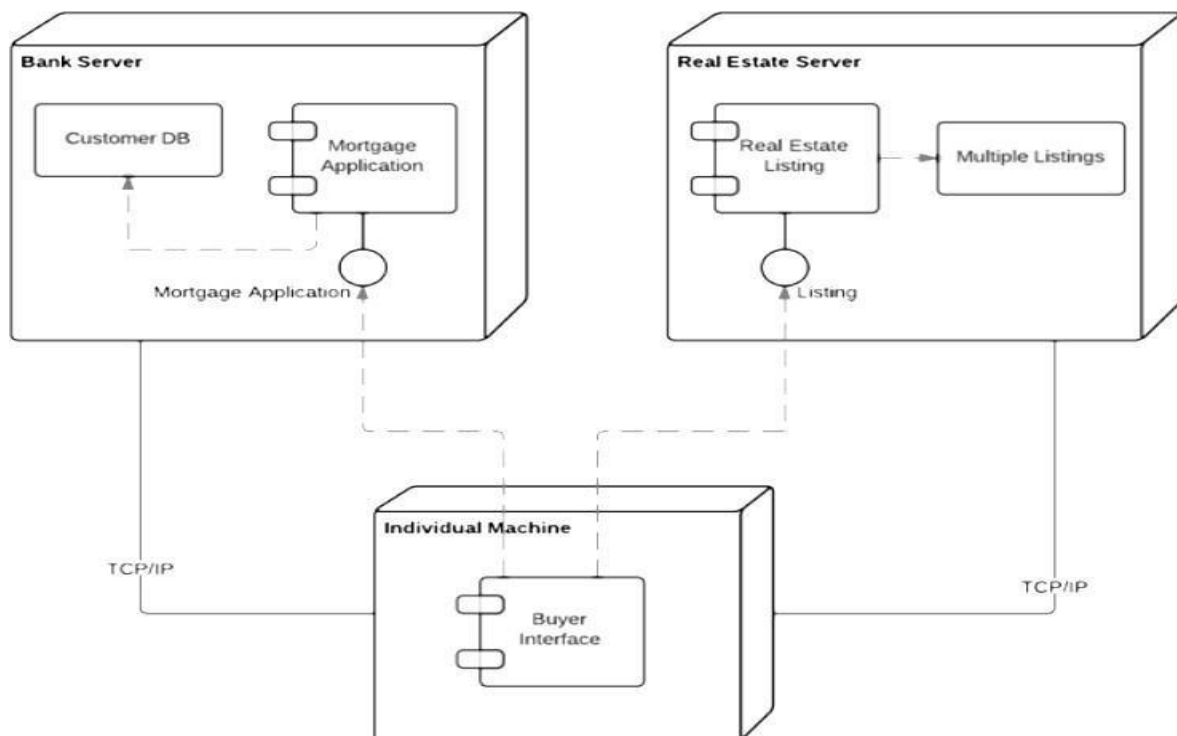
### ATM:



### RAILWAY RESERVATION SYSTEM:



### **CUSTOMER SUPPORT SYSTEM:**



### **POINT OF SALES:**

