LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SOFTWARE ENGINEERING LAB

(20IT55)

LABORATORY MANUAL

B.Tech IV Sem CSE

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)
Accredited by NAAC & NBA (Under Tier – I) ISO 9001:2015 Certified Institution
Approved by AICTE, New Delhi. and Affiliated to JNTUK, Kakinada
L.B. REDDY NAGAR, MYLAVARAM, KRISHNA DIST., A.P.-521 230.
http://www.lbrce.ac.in, cselbreddy@gmail.com, Phone: 08659-222933, Fax: 08659-222931

Vision of the Department

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department

- **DM1:** To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.
- **DM2:** To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.
- **DM3:** Expose the students to cutting edge technologies which enhance their employability and knowledge.
- **DM4:** Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs)

- **PEO1:** Pursue higher education, entrepreneurship and research to compete at global level.
- **PEO2:** Design and develop products innovatively in computer science and engineering and in other allied fields.
- **PEO3:** Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all the levels with ethics and necessary attitude.
- **PEO4:** Serve ever-changing needs of society with a pragmatic perception.

PROGRAMME OUTCOMES (POs):

PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success of organization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO 3	To inculcate an ability to analyze, design and implement database applications.

List of Experiments

UML:

Consider the following Case Studies:

1) Automated Teller Machine (ATM)
2) Library Management System
3) Railway Ticket Reservation System
4) Point-of-Sale Terminal
5) Customer Support Service Operations
6) Cab Booking Service

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

1) Automated Teller Machine(ATM)
2) Library Management System
3) Railway Ticket Reservation System

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

1) Point-of-Sale Terminal
2) Customer Support Service Operations
3) Cab Booking Service

Week-3: Basics of UML

1) Introduction to UML
2) Familiarization with any one of the Software such as Rational Rose or Umbrello or Gliffy Diagram etc.

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

1) Identify and Analyze the Actors
2) Identify the Actions
3) Analyze the Relationships between Actors and Actions
4) Sketch the Use Case Diagram

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 your problem
2) Analyze the Attributes and Operations
3) Analyze the Relationships between Classes
4) Sketch the Class Diagram

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

1) Identify the Objects participating in Communication
2) Identify the Messages between the objects
3) Give numbering to messages
4) Use Flat Sequencing or Procedural Sequencing for numbering.

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

1) Identify activities in your casestudy.
2) Identify relationships among activities.
3) Use Fork or Join, if necessary.
4) Sketch the diagram

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 case study.
2) Create a visual for each of the component.
3) Describe the organization and relationships between components using interfaces, ports etc.
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 the nodes.
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 any sort?
Facilitates construction of models that in turn can be used to:
Reason about system behavior.
Present proposed designs to others.
Document key elements of design for future understanding.
2) Which graphical notation should 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 BoochThe Booch Method,JamesRumbaugh,etal.ObjectModelingTechnique,IvarJacsobson—ObjectoryIn1994, BoochandRumbaugh,thenbothatRational,startedworkingonaunificationoftheirmethods.Afirst draftoftheirUnifiedMethodwasreleasedinOctober1995.In1996,(+/-)JacobsonjoinedBoochand Rumbaugh at Rational; the name UML was coined. In 1997 the Object Management Group (OMG) acceptedUMLasanopenandindustrystandardvisualmodelinglanguageforobjectorientedsystems. Current version of UMLis2.0.

UML DiagramTypes

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 DiagramsShow 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
- Objectsdiagrams
- Deploymentdiagrams
- Packagediagrams
- Componentdiagrams

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.Behavioralmodelingshowsthedynamicnatureofthesystem.Theyconsistofthe following:

- Activitydiagrams
- Interactiondiagrams
- Use casediagrams

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.

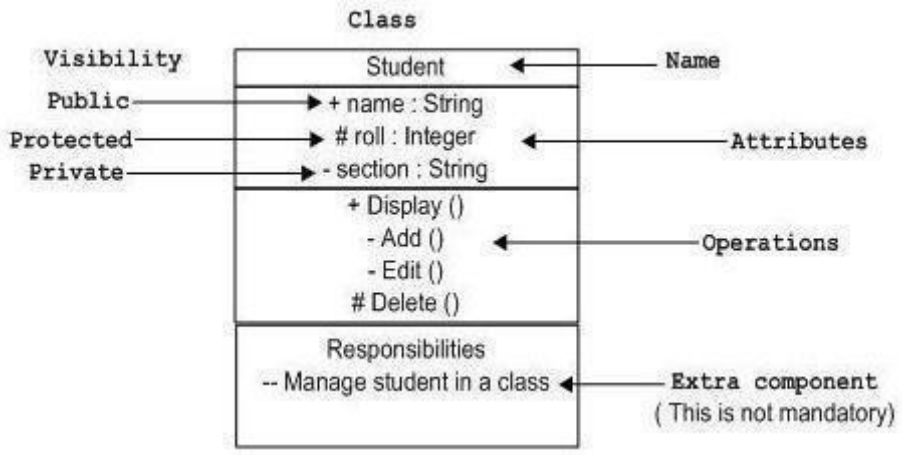
- Classes
- Interface

- Collaboration
- Usecase
- Activeclasses
- Components
- Nodes

Class Notation:

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

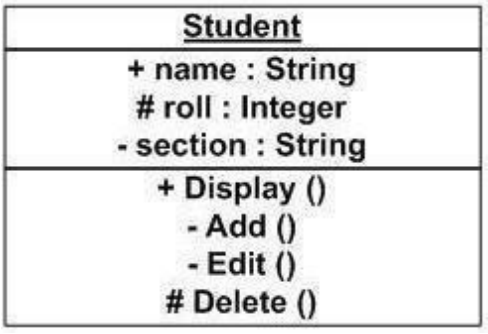
- The top section is used to name theclass.
- The second one is used to show the attributes of theclass.
- The third section is used to describe the operations performed bytheclass.
- The fourth section is optional to show anyadditionalcomponents.



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 shownbelow:

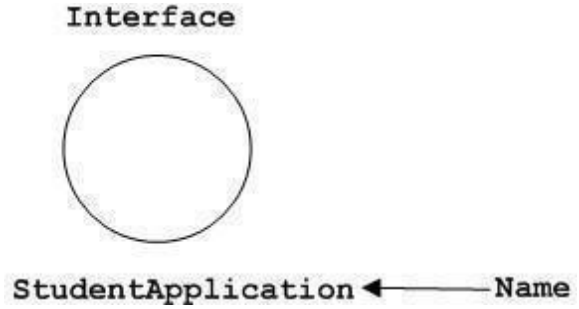


As object is the actual implementation of a class which is known as the instance of a class. So it has the same usage as theclass.

Interface Notation:

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

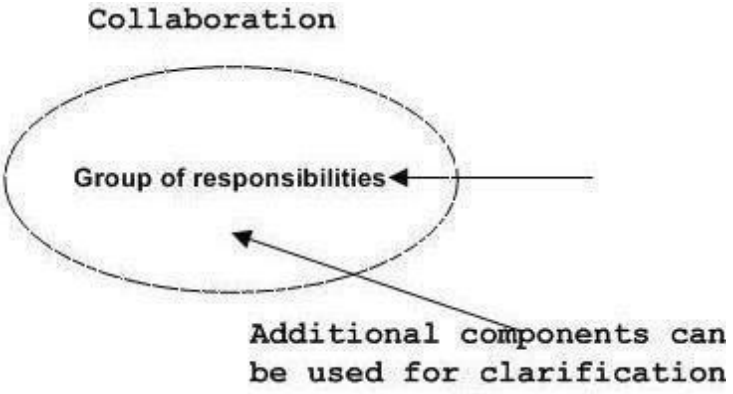
the circle.
Interfaceisusedtodescribefunctionalitywithoutimplementation.Interfaceisthejustlikeatemplatewhere



youdefinedifferentfunctionsnottheimplementation.Whenaclassimplementstheinterface it also implements the functionality as pertherequirement.

Collaboration Notation:

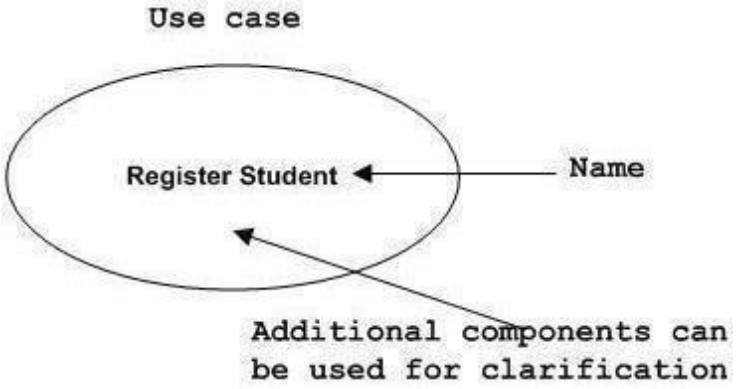
Collaboration is represented by a dotted eclipse as shown below. It has a name written inside the eclipse.



Collaboration represents responsibilities. Generally responsibilities are in a group.

Use case Notation:

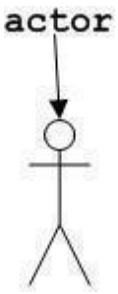
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.



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

Actor Notation:

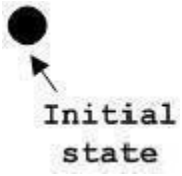
An actor can be defined as some internal or external entity that interacts with the system.



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

Initial State Notation:

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



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

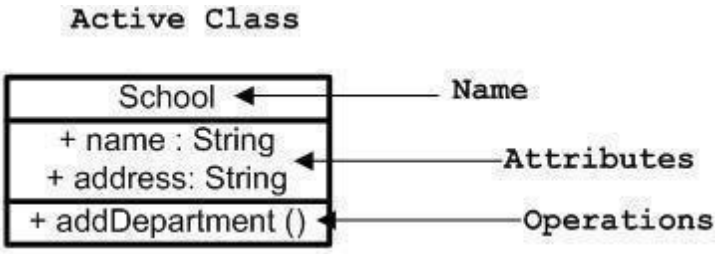
Final State Notation:

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

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

Active class Notation:

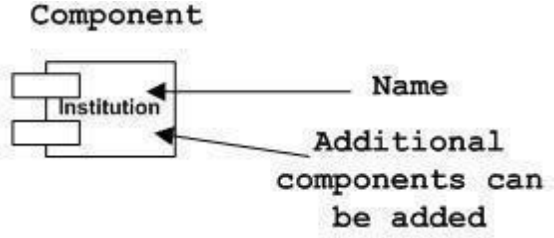
Active class looks similar to a class with a solid border. Active class is generally used to describe concurrent behavior of a system.



Active class is used to represent concurrency in a system.

Component Notation:

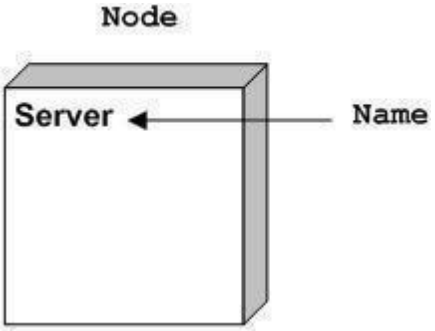
A component in UML is shown as below with a name inside. Additional elements can be added wherever required.



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

Node Notation:

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



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

Behavioural Things:

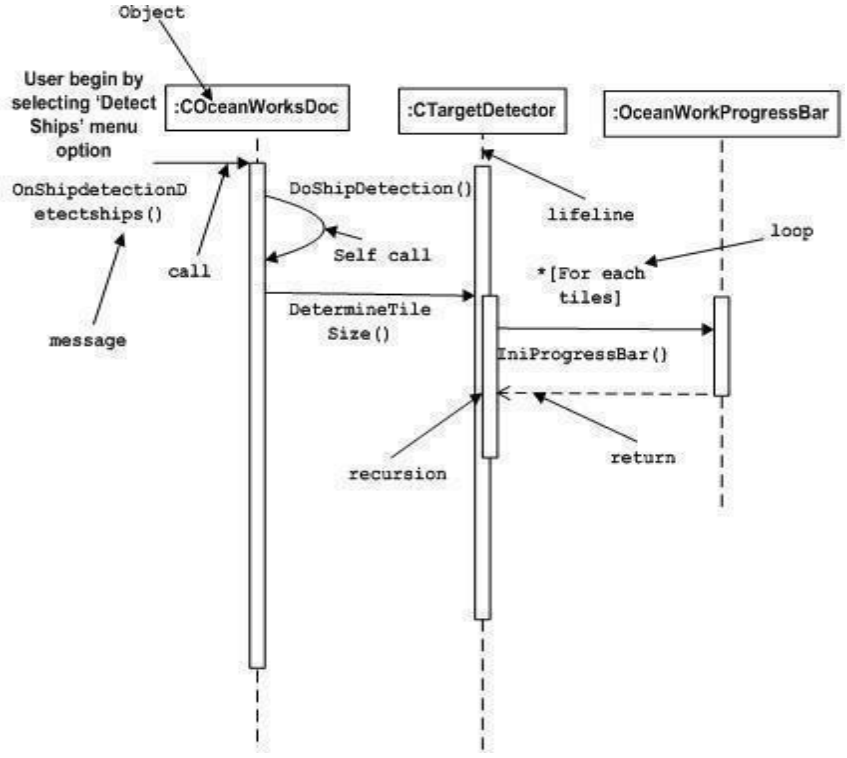
Dynamicpartsareoneofthemost important elements inUML.UMLhasasetofpowerfulfeaturesto representthedynamicpartofsoftwareandnon-softwaresystems.Thesefeaturesincludeinteractions and statemachines.

Interactions can be of two types:

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

Interaction Notation:

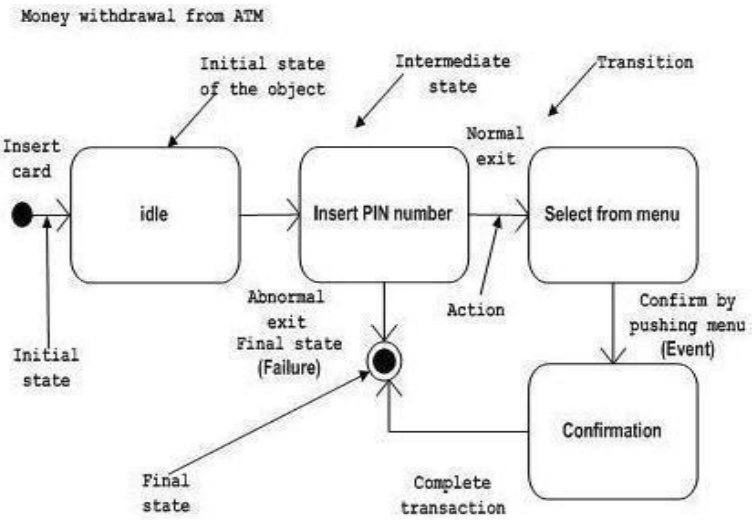
Interaction is basically message exchange between two UML components. The following diagram represents different notations used in an interaction.



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.

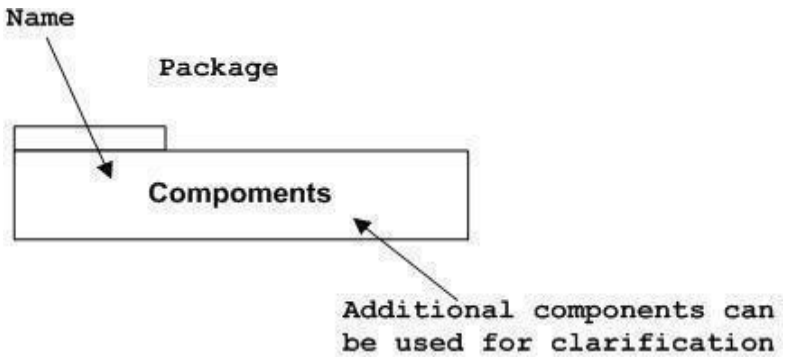


Grouping Things:

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.

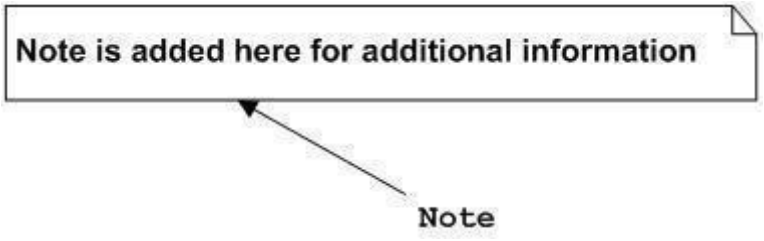


Annotational Things:

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.



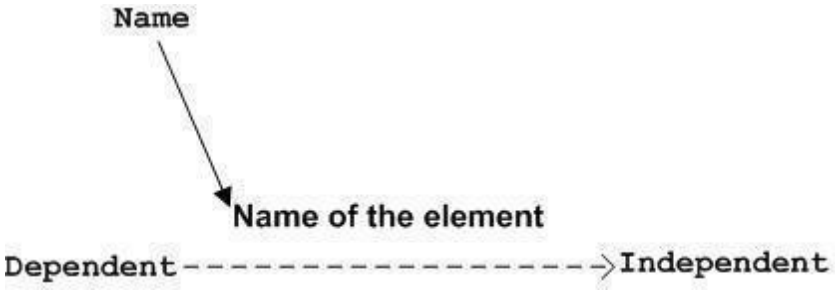
Relationships

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.

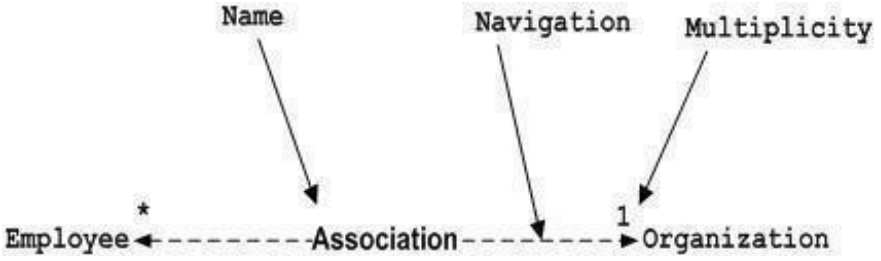


Dependency is used to represent dependency between two elements of a system.

Association Notation:

AssociationdescribeshowtheelementsinanUMLdiagramareassociated.Insimpleworditdescribes how many elements are taking part in aninteraction.

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 areassociated.



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

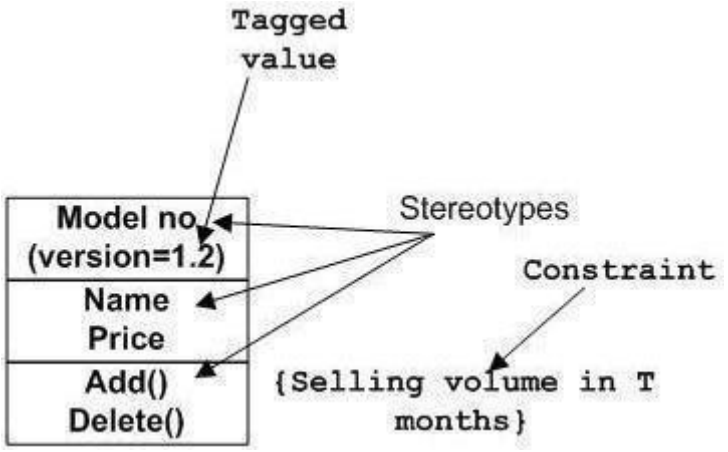
child relationship.
Generalization is represented by an arrow with hollow arrow head as shown below. One end represents the parent element and the other end child element.

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 newelements)
- Tagged values (Represents newattributes)
- Constraints (Represents theboundaries)



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 diagramisnotonlyusedforvisualizing,describinganddocumentingdifferentaspectsofasystembut also for constructing executable code of the softwareapplication.

The class diagram describes the attributes and operations of a class andalso 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 diagramsaretheonlydiagramswhichcanbedirectlymappedwithobjectorientedlanguagesandthus widely used at the time ofconstruction.

TheUMLdiagramslikeactivitydiagram,sequencediagramcanonlygivethesequenceflowoftheapp licationbutclassdiagramisabitdifferent.Soitisthem Ost popularUMLdiagram inthecoder community.

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

- Analysisanddesignofthestaticview ofanapplication.
- Describe responsibilities of asystem.
- Base for component anddeploymentdiagrams.
- Forward and reverseengineering.

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:

- Thenameoftheclassdiagramshouldbemeaningfultodescribetheaspectofthesystem.
- Eachelementandtheirrelationshipsshouldbeidentifiedinadvance.
- Responsibility(attributesandmethods)ofeachclassshouldbeclearlyidentified.
- For each class minimum number of properties should be specified. Because unnecessary properties will make the diagramcomplicated.
- Use notes whenever required to describe some aspect of the diagram. Because at theend of the drawing it should be understandable tothedeveloper/coder.
- Finally,beforemakingthefinalversion,thediagramshouldbedrawnonplainpaperand rework as many times as possible to makeitcorrect.

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

- Firstof all Order and Customer are identified as the two elements of the system and they haveaonetomanyrelationshipbecauseacustomercanhavemultipleorders.
- We would keep Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder andNormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, theyhave additional functions like dispatch () andreceive().

UML Object Diagram

Objectdiagramsarederivedfromclassdiagramssoobjectdiagramsaredependentuponclass 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 reverseengineering.
- Object relationships of a system.
- Static view of aninteraction.
- Understandobjectbehaviourandtheirrelationshipfrompracticalperspective.

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 objectdiagramsthenwecanhaveunlimitednumberofinstanceswhichareuniqueinnature. So only those instances are considered which are having impact on thesystem.

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,analyzethesystemanddecidewhichinstancesarehavingimportantdataand association.
- Second, consider only those instances which will coverthefunctionality.
- Third,makesomeoptimizationasthenumbersofinstancesareunlimited.

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

- Object diagrams are consist ofobjects.
- Thelinkinobjectdiagram issued toconnectobjects.
- Objects and links are the two elements used to construct anobjectdiagram.

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

- Theobjectdiagramshouldhaveameaningfulnametoindicateitspurpose.
- The most important elements are tobeidentified.
- The association among objects shouldbeclarified.
- Valuesofdifferentelementsneedtobecapturedtoincludeintheobjectdiagram.
- Add proper notes at points where more clarity is required.

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

- Customer
- Order
- SpecialOrder
- NormalOrder

Now the customer object (C) is associated with three order objects (O1, O2 and O3). These order objects are associated with special order and normal order objects (S1, S2 and N1). The customer is having the following three orders with different numbers (12, 32 and 40) for the particular time considered.

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

Forordersthevaluesare12,32,and40whichimpliesthattheobjectsarehavingthesevalues for the particular moment (here the particular time when the purchase is made is considered as the moment) when the instance iscaptured.

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

UML Component Diagram

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

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

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

Purpose:

ComponentdiagramisaspecialkindofdiagraminUML.ThepurposeisalsodifferentfromallotherdiagramsdiscussedsofarItdoesnotdescribethefunctionalityofthesystembutitdescribesthecomponentsusedtomakethosefunctionalities.

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 asystem.
- Construct executables by using forward andreverseengineering.
- Describe the organization and relationships of thecomponents.

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 thesystem.
- Libraries and other artifacts relevant to theapplication.
- Relationships among theartifacts.

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

- Use a meaningful name to identify the component for which the diagram is tobedrawn.
- Prepare a mental layout before producingusingtools.
- Use notes for clarifying importantpoints.

The following is a component diagram for order management system. Here the artifacts are files.SothediagramshowsthefilesintheapplicationandtheirrelationshipsInactualthecomponentdiagramalsocontainsfiles,libraries,foldersetc.

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

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 diagramsare usedfordescribingthehardwarecomponentswheresoftwarecomponentsaredeployed.Component diagrams and deployment diagrams are closelyrelated.

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

UMLismainlydesignedtofocusonsoftwareartifactsofasystem.Butthesetwodiagramsare special diagrams used to focus on software components and hardwarecomponents.

So most of the UML diagrams are used to handle logical components but deployment diagramsaremadetofocusonhardwaretopologyofasystem.Deploymentdiagramsareusedbythe systemengineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of asystem.
- Describe the hardware components used to deploy softwarecomponents.
- Describe runtime processingnodes.

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 hardwareused 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 amongnodes

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
- Cachingserver
- Server

The application is assumed to be a webbased 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 andusecasediagramisoneofthem.Nowaswehavetodiscussthat theusecasediagramisdynamic in nature there should be some internal or external factors for makingtheinteraction.

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.

Becauseotherfourdiagrams(activity,sequence,collaborationandStatechart)arealsohaving thesamepurpose.Sowewilllookintosomespecificpurposewhichwilldistinguishitfromotherfour 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 asystem.
- Used to get an outside view ofasystem.
- Identify external and internal factors influencing thesystem.
- Show the interacting among the requirements areactors.

How to draw Use Case Diagram?

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

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

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

- Functionalities to be represented as an usecase
- Actors
- Relationships among the use cases andactors.

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.

- Thenameofausecaseisveryimportant.Sothenameshouldbechoseninsuchawayso that it can identify the functionperformed.
- Give a suitable name foractors.
- Show relationships and dependencies clearly inthediagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identifyrequirements.
- Use note when ever required to clarify someimportantpoints.

Thefollowingisasampleusecasediagramrepresentingtheordermanagementsystem.Soif we look into the diagram then we will find three use cases (Order, SpecialOrder and NormalOrder) and one actor which iscustomer.

TheSpecialOrderandNormalOrderusecasesareextendedfromOrderusecase.Sotheyhave extends relationship. Another important point is to identify the system boundary which is shown in thepicture. TheactorCustomerliesoutsidethesystemasitisanexternaluserofthesystem.

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.

ThisinteractivebehaviourisrepresentedinUMLbytwodiagramsknownasSequencediagram andCollaborationdiagram.Thebasicpurposesofboththediagramsaresimilar.

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:

Thepurposesofinteractiondiagramsaretovisualizethe interactivebehaviourofthesystem. Now visualizing interaction is a difficult task. So the solution is to use different types of models to capture the different aspects of theinteraction.

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

So the purposes of interaction diagram can be describes as:

- To capture dynamic behaviour of asystem.
- To describe the message flow inthesystem.
- To describe structural organization of theobjects.
- To describe interaction amongobjects.

How to draw Interaction Diagram?

As we have already discussed that the purpose of interaction diagrams are to capture the dynamicaspectofasystem.Sotocapturethedynamicaspectweneedtounderstandwhatadynamic aspect is and how it is visualized. Dynamic aspect can be defined as the snapshot of the running system at a particularmoment.

WehavetwotypesofinteractiondiagramsinUML. Oneissequencediagram andtheother is acollaborationdiagram.Thesequencediagramcapturesthetimesequenceofmessageflowfromone object to another and the collaboration diagram describes the organization of objects in a system taking part in the messageflow.

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

- Objects taking part in theinteraction.
- Message flows among theobjects.
- The sequence in which the messages areflowing.
- Objectorganization.

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).

ThefollowingdiagramhasshownthemessagesequenceforSpecialOrderobjectandthesame can be used in case of NormalOrder object. Now it is important to understand the time sequence of messageflows. Themessageflowisnothingbutamethodcall ofanobject.

ThefirstcallissendOrder()whichisamethodofOrderobject.Thenextcallisconfirm()which is a method of SpecialOrder object and the last call is Dispatch () which is a method of SpecialOrder object.Soherethediagramismainlydescribingthemethodcallsfrom oneobject toanotherandthis is also the actual scenario when the system isrunning.

The Collaboration Diagram:

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

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

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

UML Statechart Diagram

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

AStatechartdiagramdescribesastatemachine.Nowtoclarifyitstatemachinecanbedefined asamachinewhichdefinesdifferentstatesofanobjectandthesestatesarecontrolledbyexternalor internalevents.

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:

StatechartdiagramisoneofthefiveUMLdiagramsusedtomodeldynamicnatureofasystem. 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 internalevents.

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 asystem.
- To model lifetime of a reactivesystem.
- To describe different states of an object during itslifetime.
- Define a state machine to model states of anobject.

How to draw Statechart Diagram?

Statechart diagram is used to describe the states of different objects in its life cycle. So the emphasisisgivenonthestatechangesuponsomeinternalorexternalevents.Thesestatesofobjects are important to analyze and implement themaccurately.

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 beanalyzed.
- Identify thestates.
- Identify theevents.

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

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

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

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

UML Activity Diagram

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

Activity diagram is basically a flow chart to represent the flow form one activity to another

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

Purpose:

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

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

Itdoesnotshowanymessageflowfromoneactivitytoanother.Activitydiagramissometime considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows differentflowlikeparallel,branched,concurrentandsingle.

So the purposes can be described as:

- Draw the activity flow of asystem.
- Describe the sequence from one activity toanother.
- Describe the parallel, branched and concurrent flow ofthesystem.

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

Oncetheabovementionedparametersareidentifiedweneedtomakea mentally outofthe entireflow. This mentally outisthetransformedintoanactivitydiagram.

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:

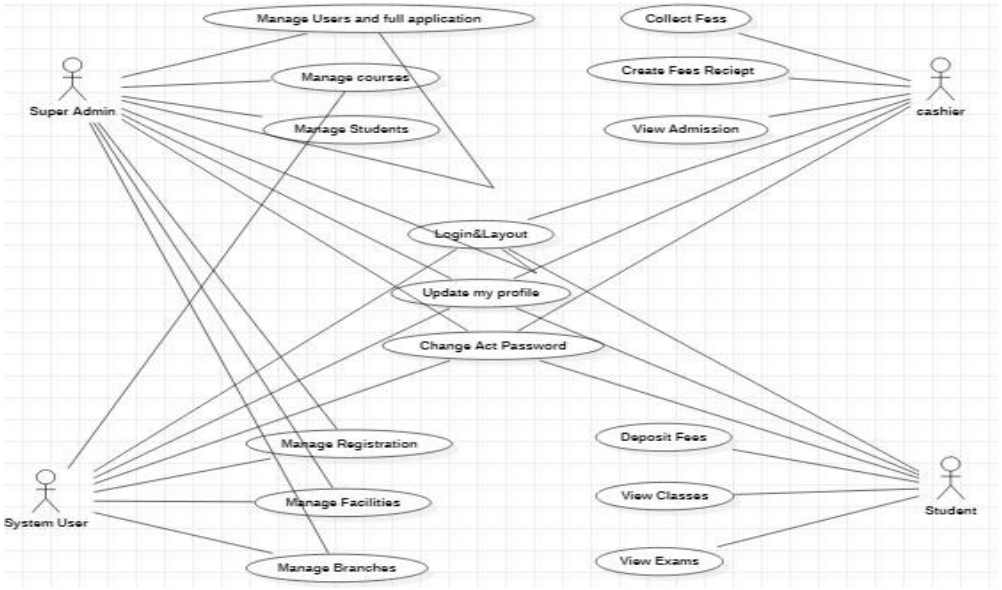
- Send order by thecustomer
- Receipt of theorder
- Confirmorder
- Dispatchorder

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:

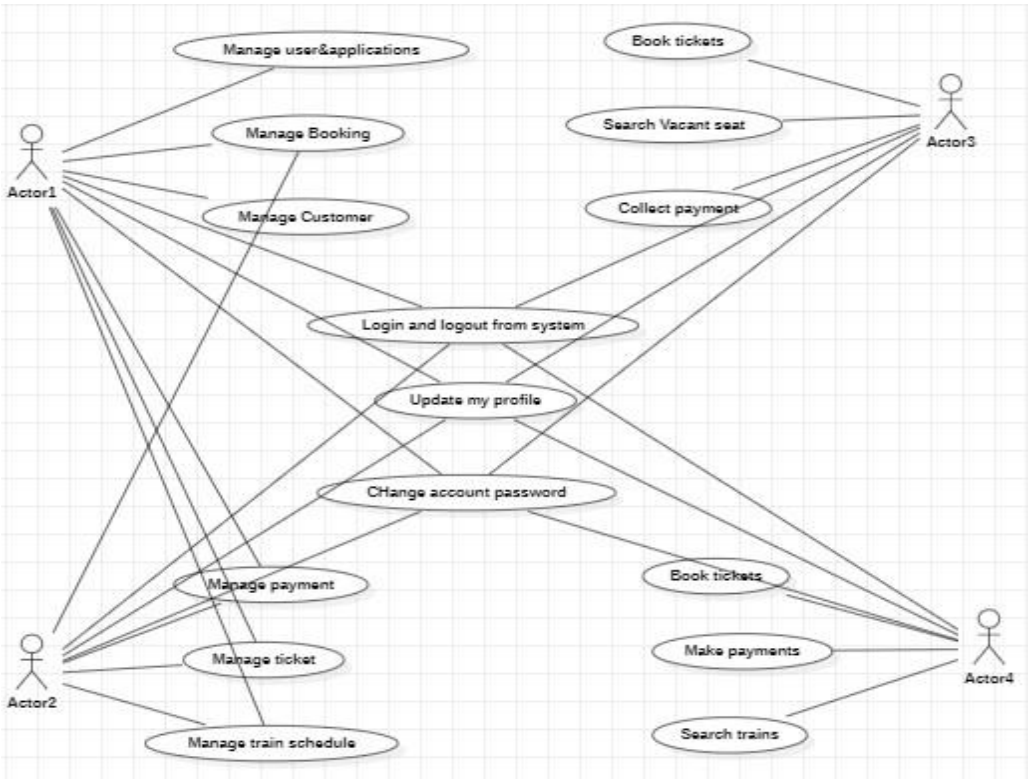
USECASE DIAGRAMS:

COLLEGE MANAGEMENT SYSTEM:

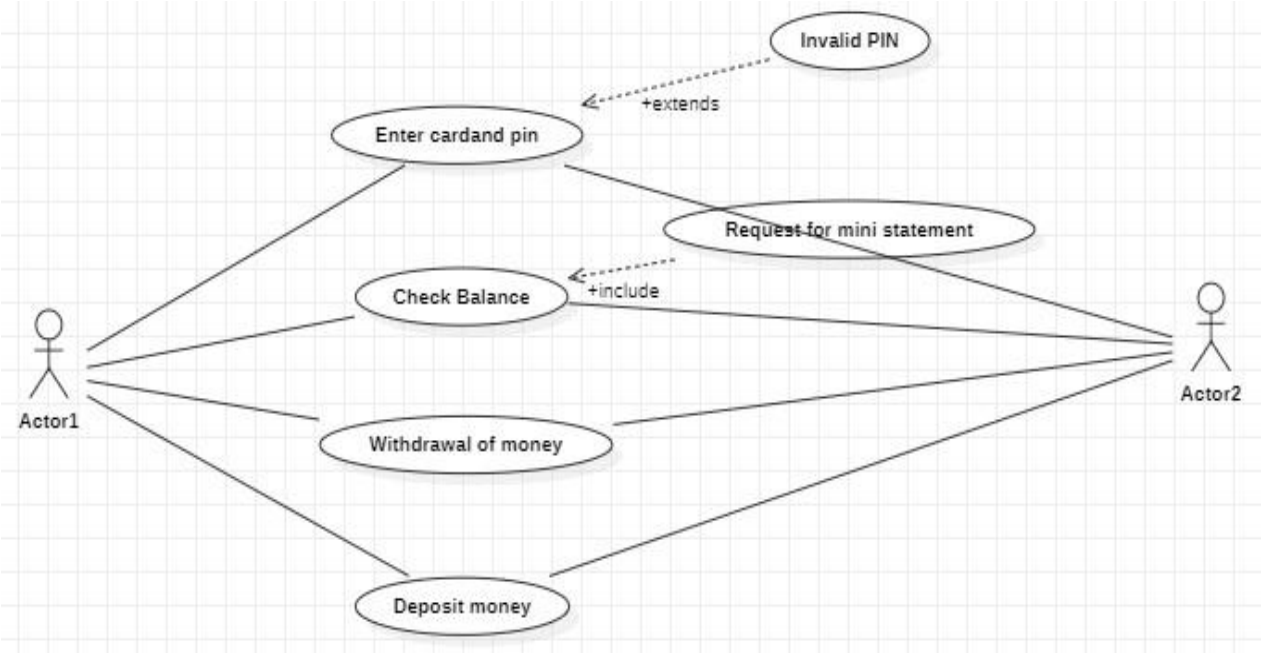


LIBRARY MANAGEMENT SYSTEM

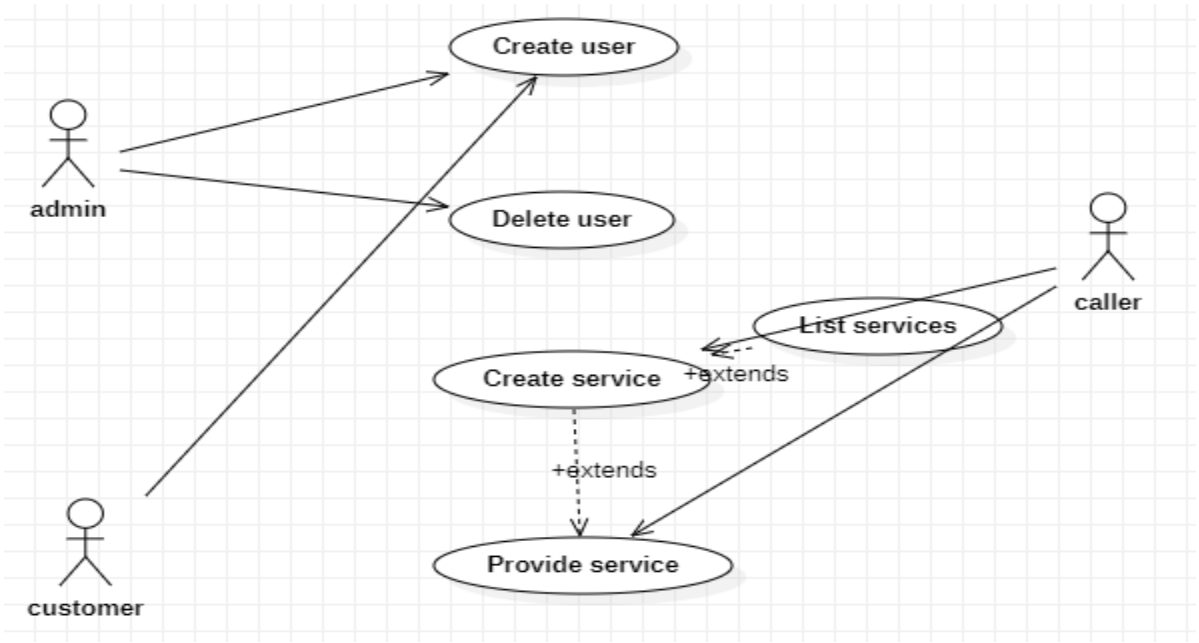
RAILWAY RESERVATION SYSTEM:



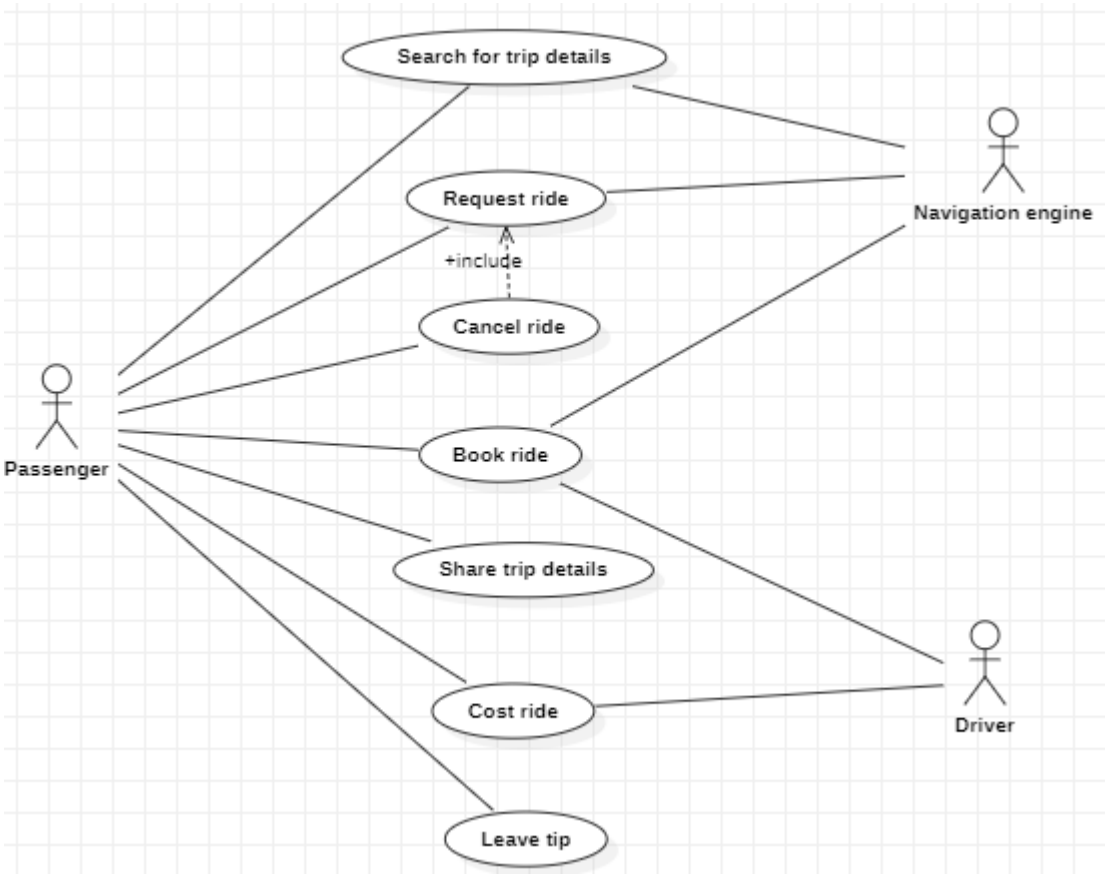
ATM:



CUSTOMER SUPPORT SYSTEM:



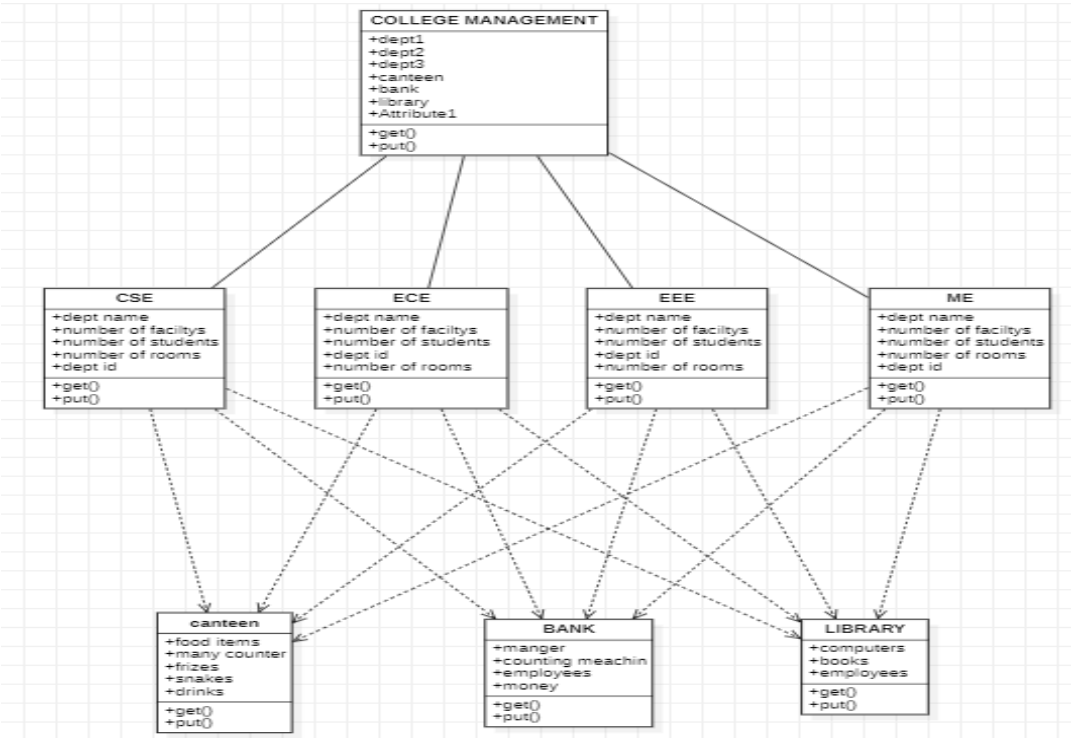
UBER CAB SYSTEM:



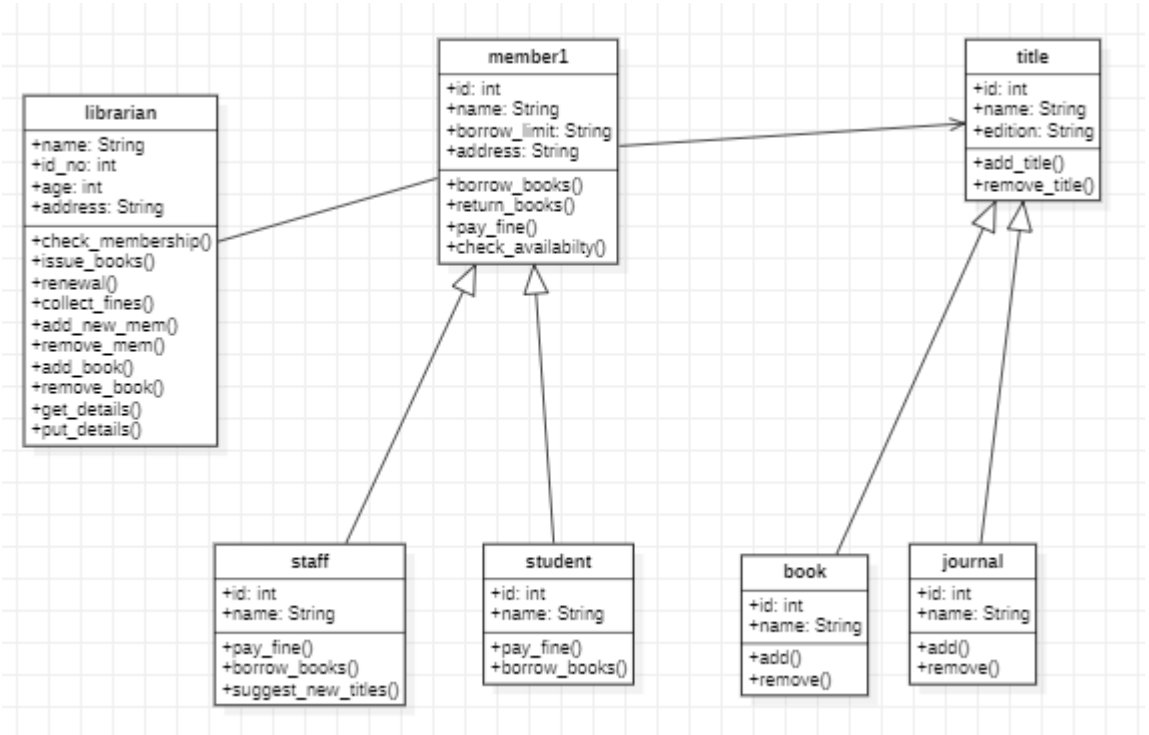
Week-5 and Week-6: For each case study given earlier, Construct CLASS Diagram in the following manner:

CLASS DIAGRAMS:

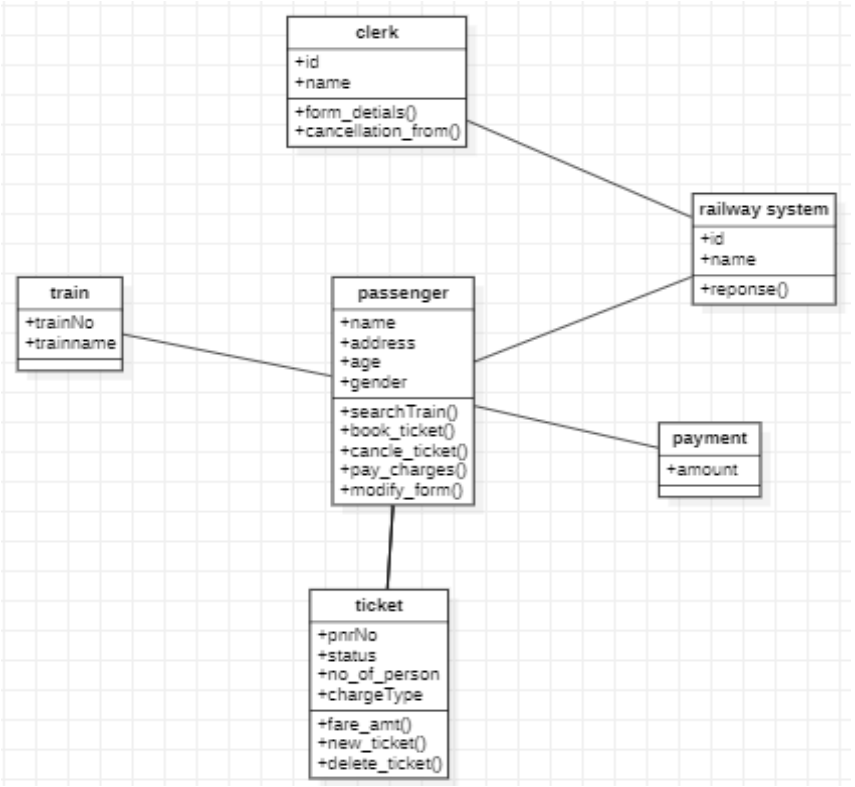
COLLEGE MANAGEMENT SYSTEM:



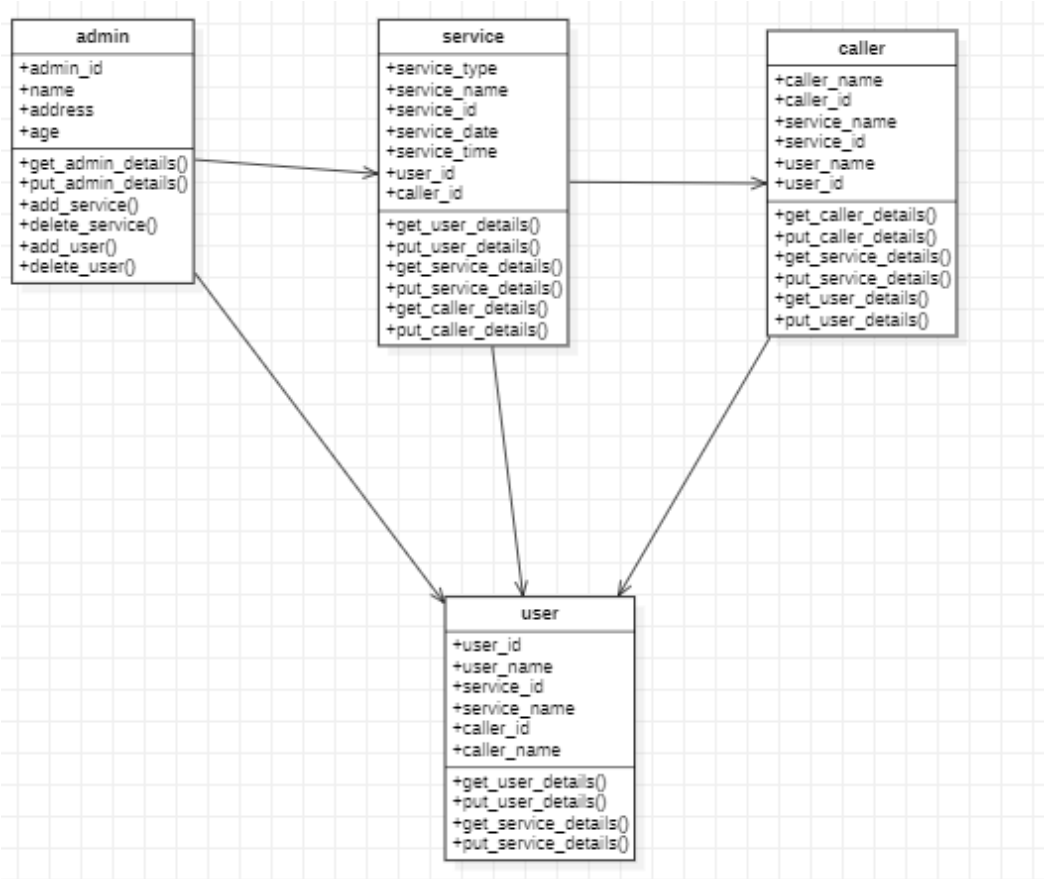
LIBRARY MANAGEMENT SYSTEM:



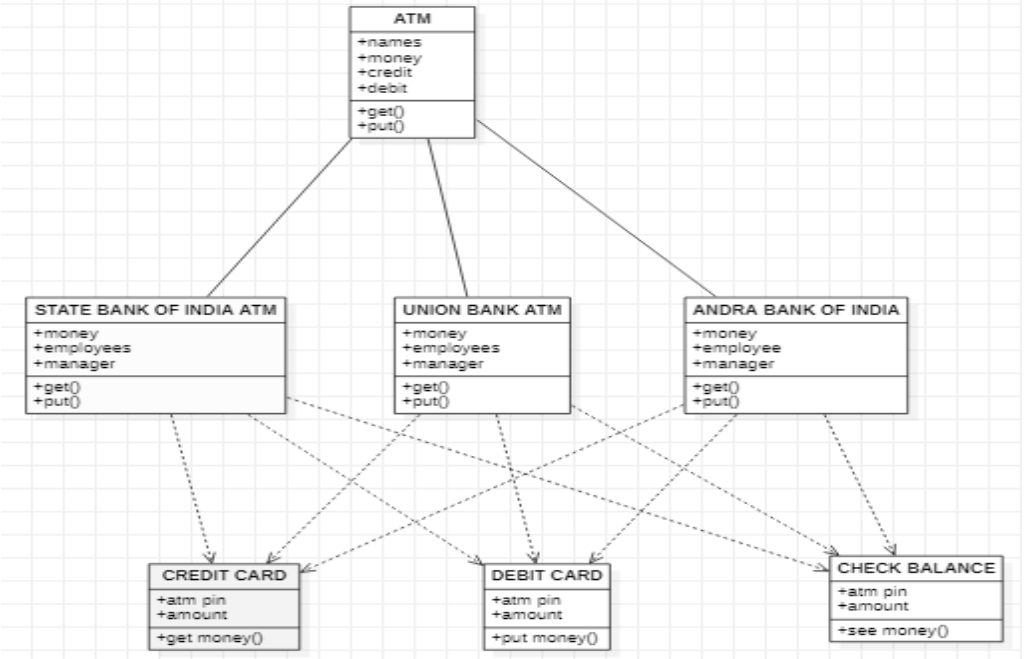
RAILWAY RESERVATION SYSTEM:



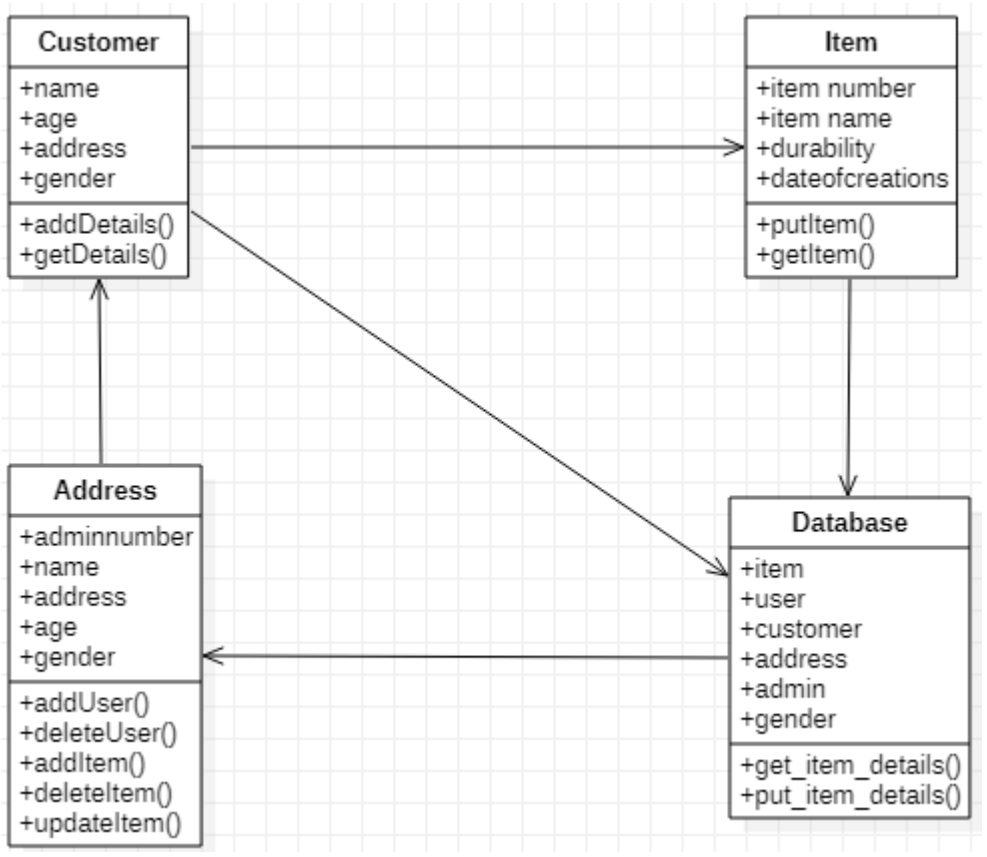
CUSTOMER SUPPORT SYSTEM:



ATM:



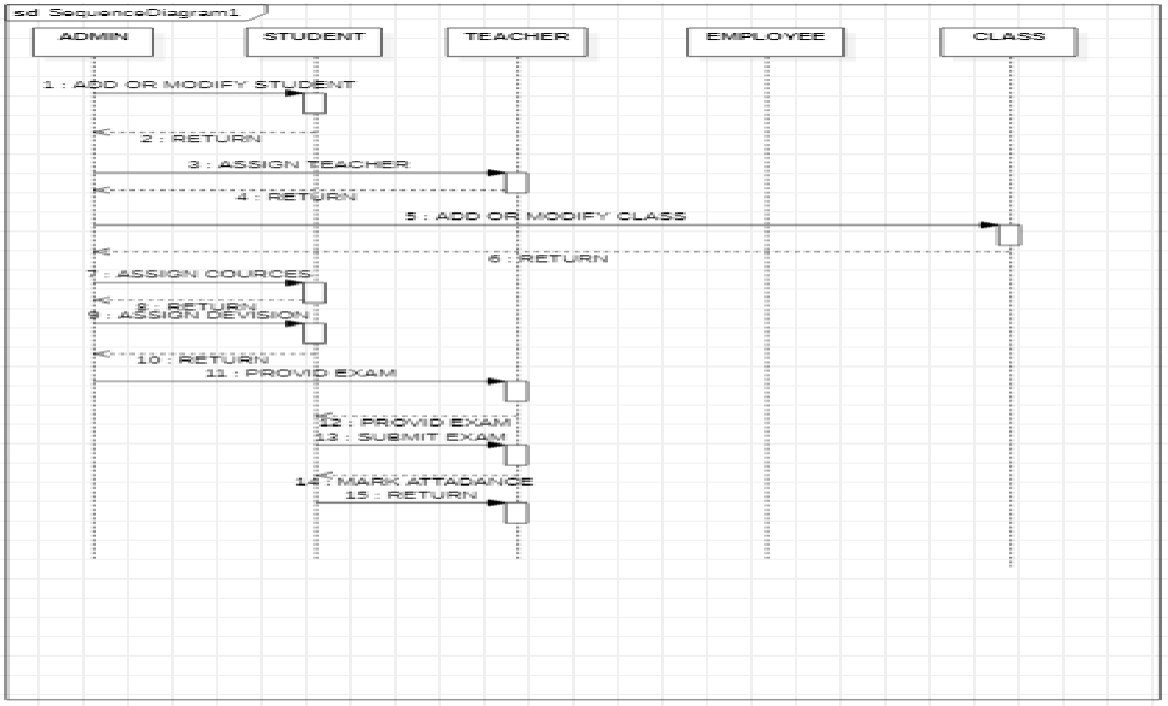
POINTOFSALES:



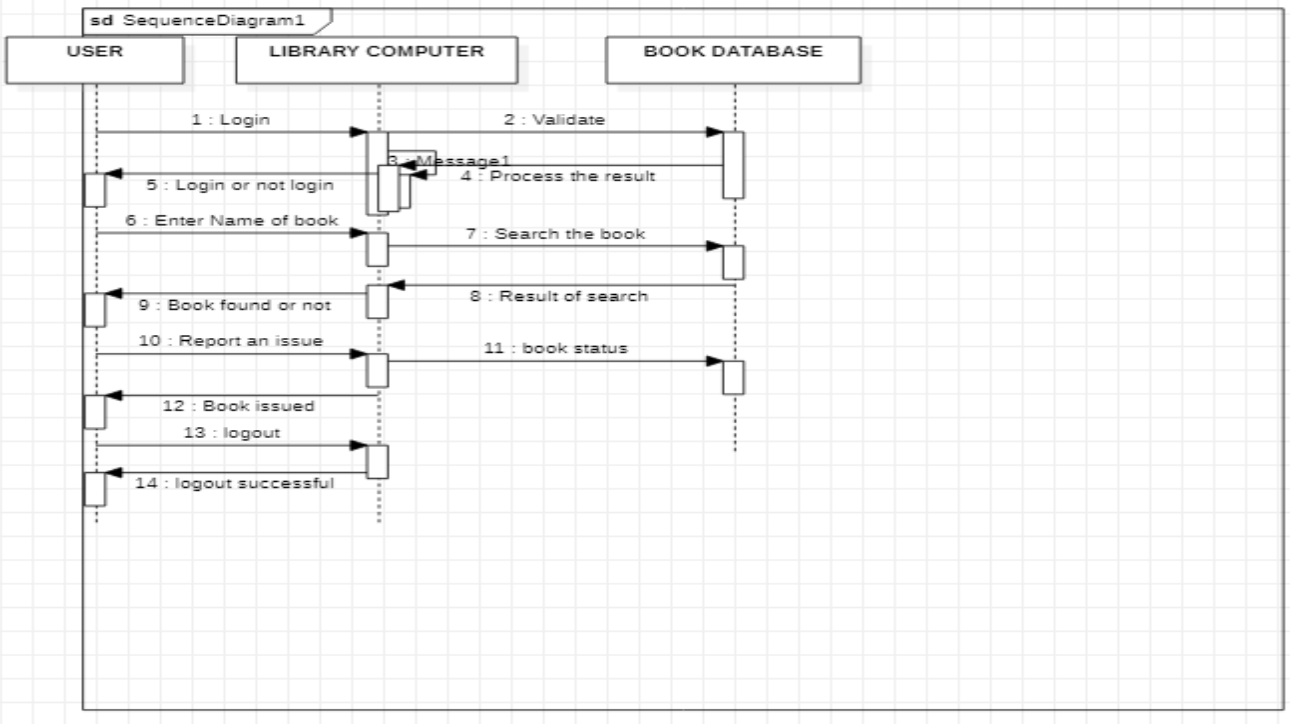
Week-7: For each case study given earlier, Construct Interaction Diagrams in the following manner:

SEQUENCE DIAGRAMS:

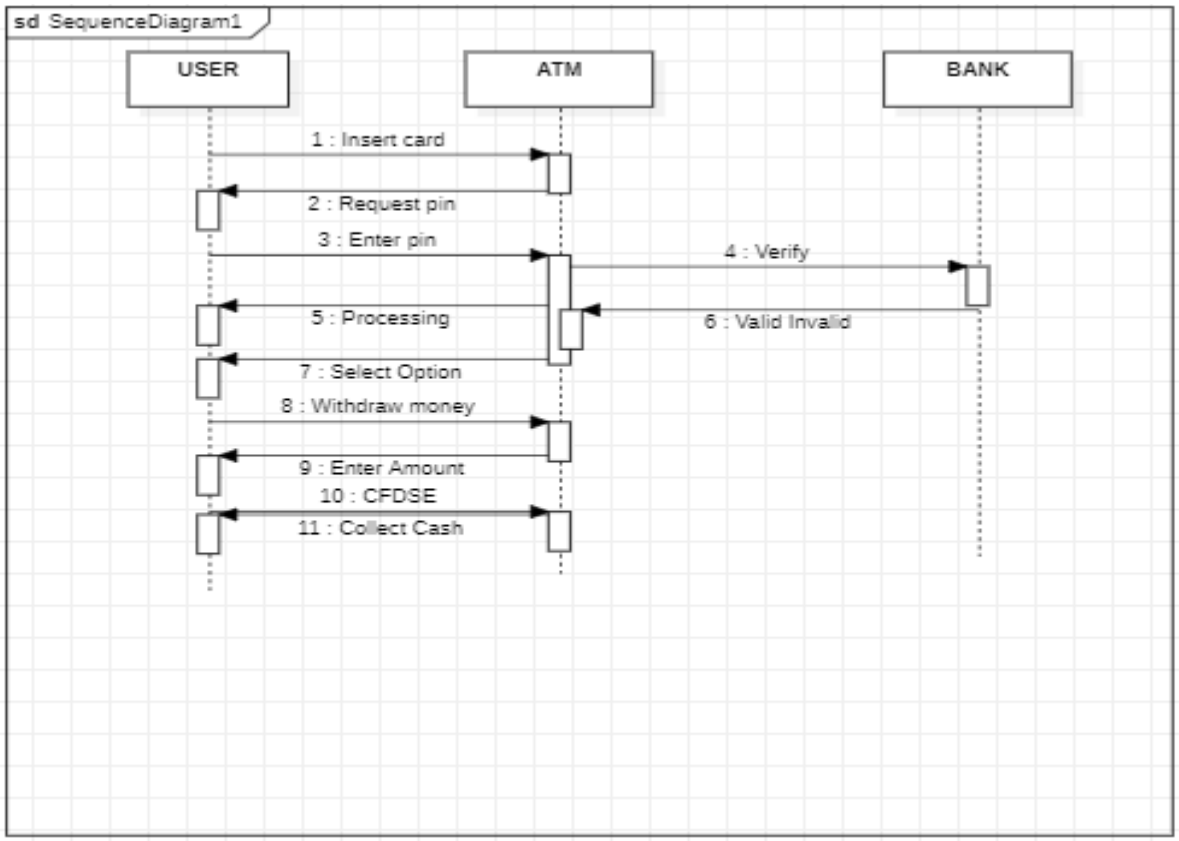
COLLEGE MANAGEMENT SYSTEM:



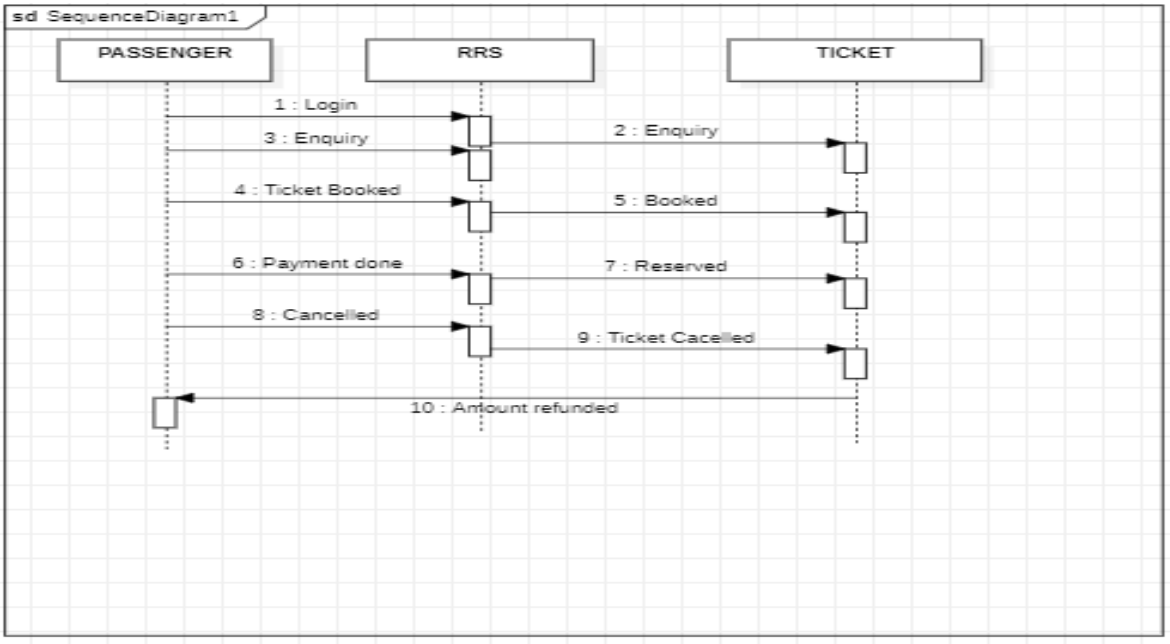
LIBRARY MANAGEMENT SYSTEM:



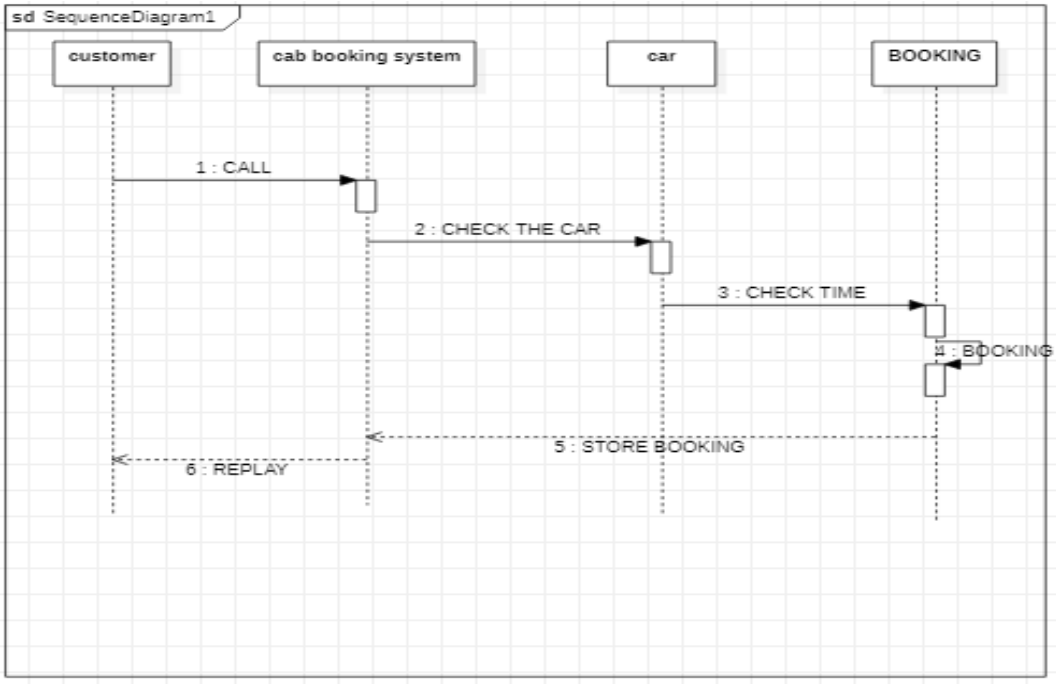
ATM:



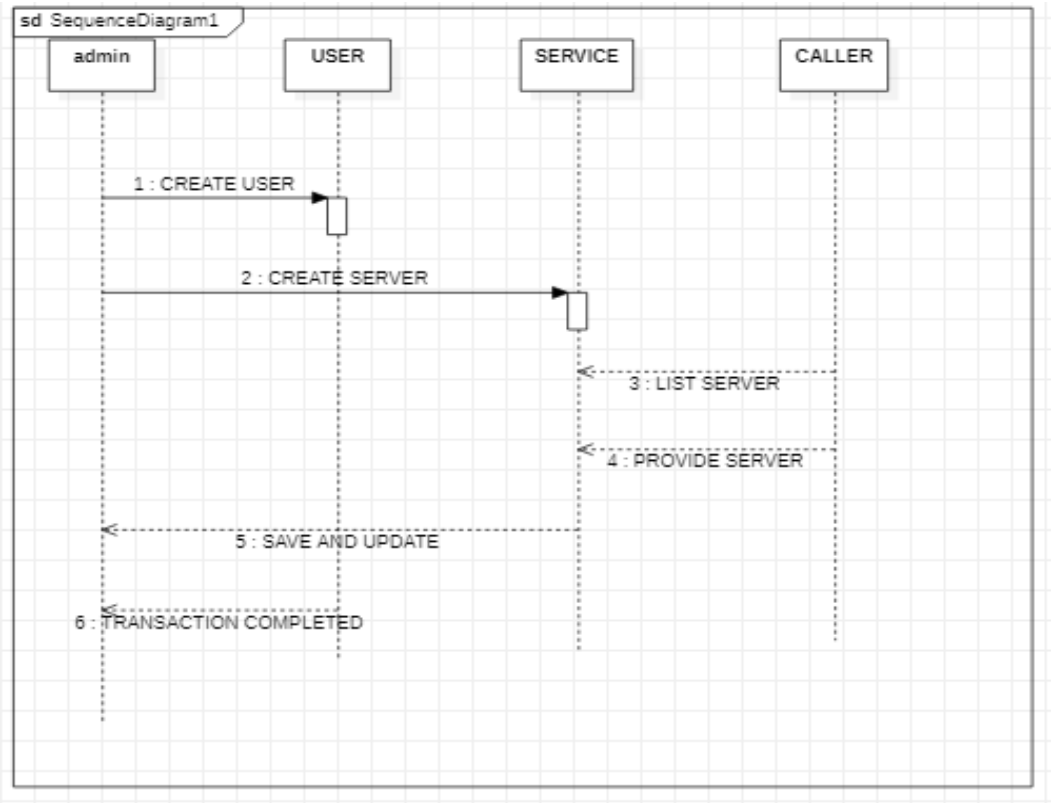
RAILWAY RESERVATION SYSTEM:



CAB BOOKING SYSTEM:

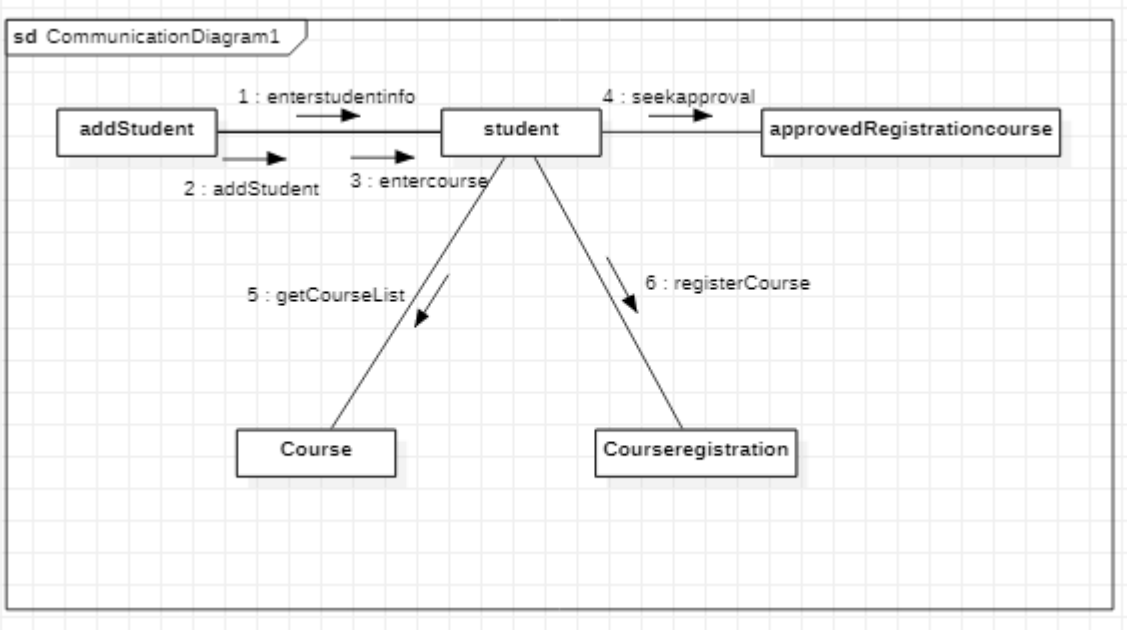


CUSTOMER SUPPORT SYSTEM:

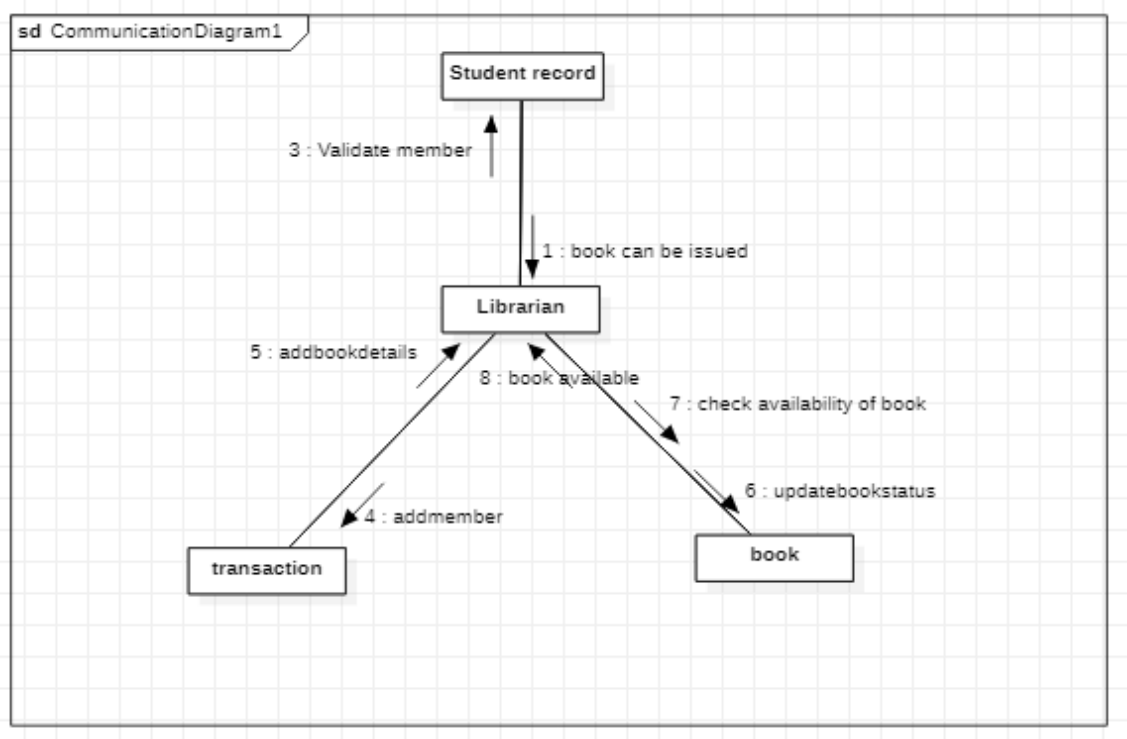


COLLOBORATION DIAGRAMS:

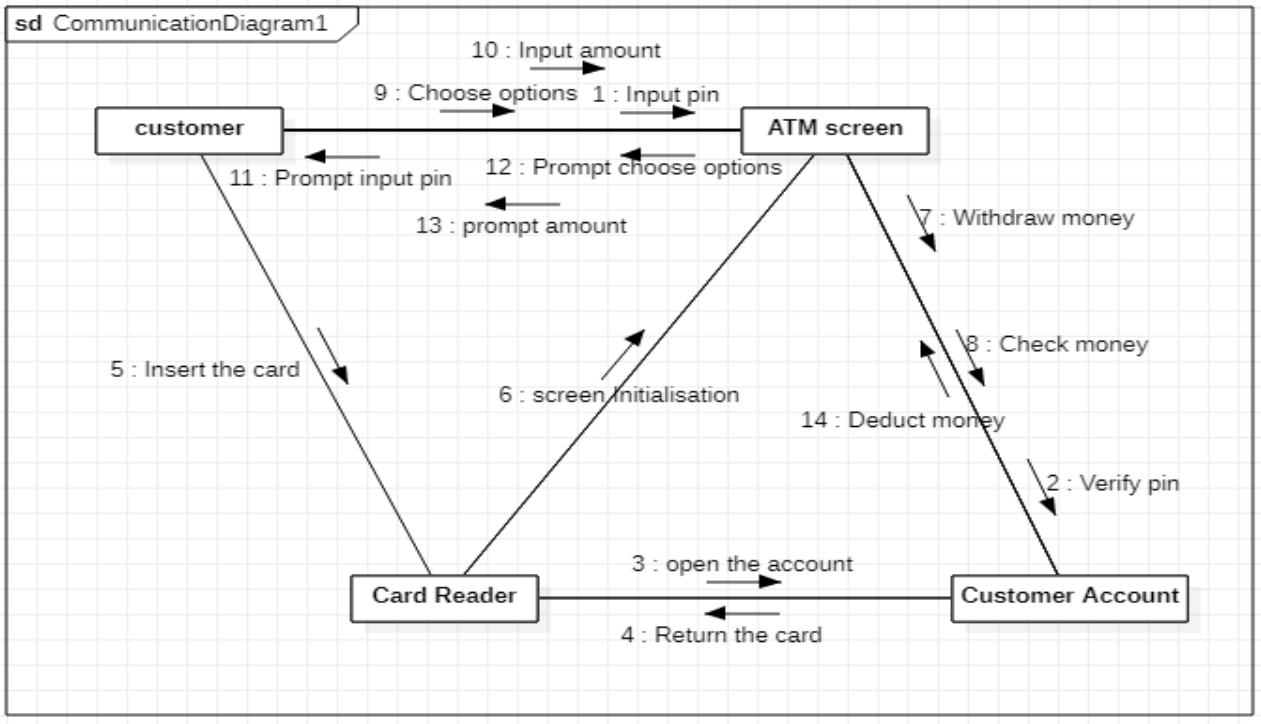
COLLEGE MANAGEMENT SYSTEM:



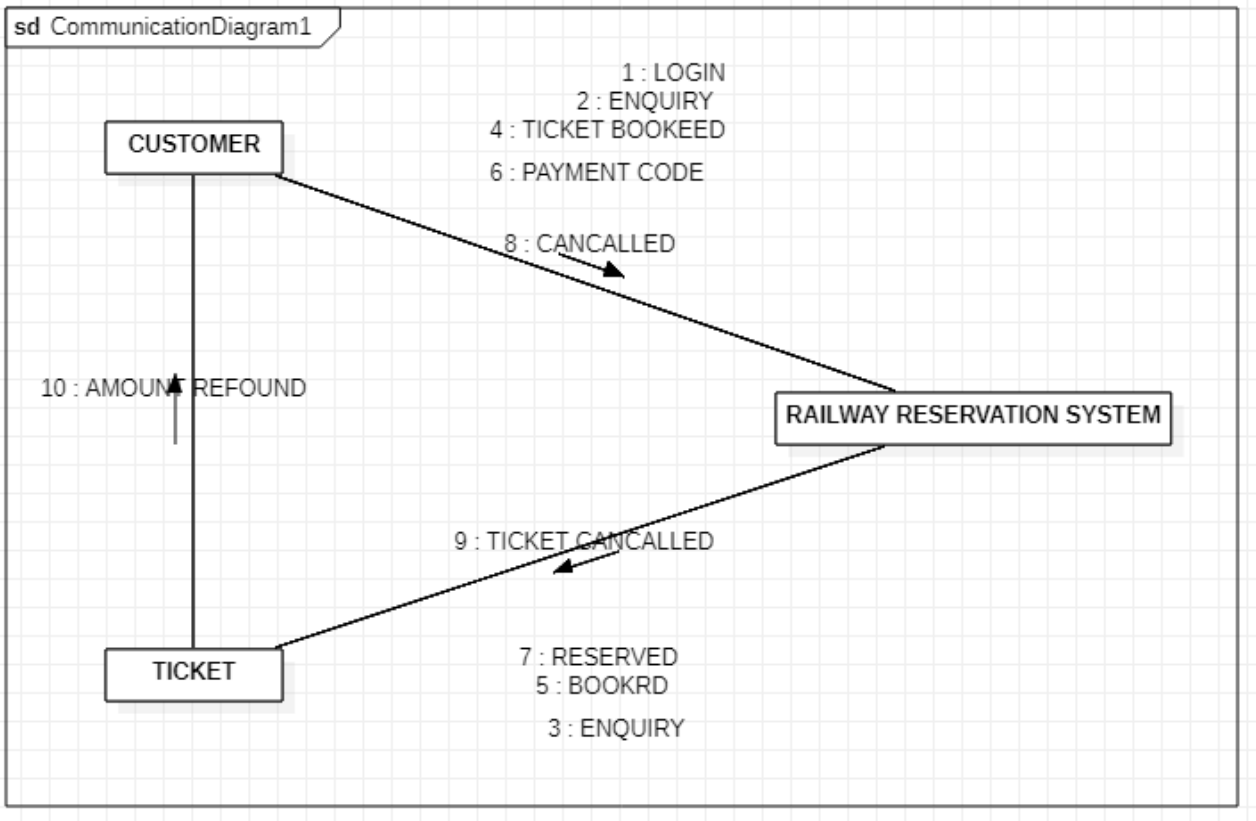
LIBRARY MANAGEMENT SYSTEM:



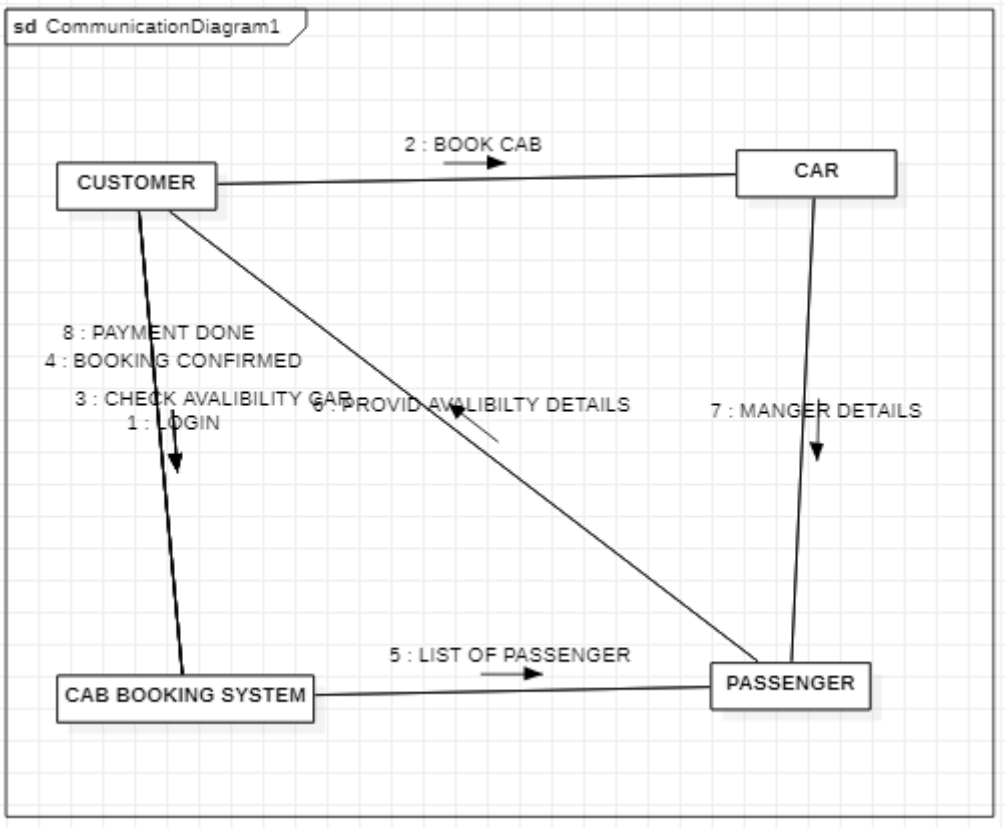
ATM:



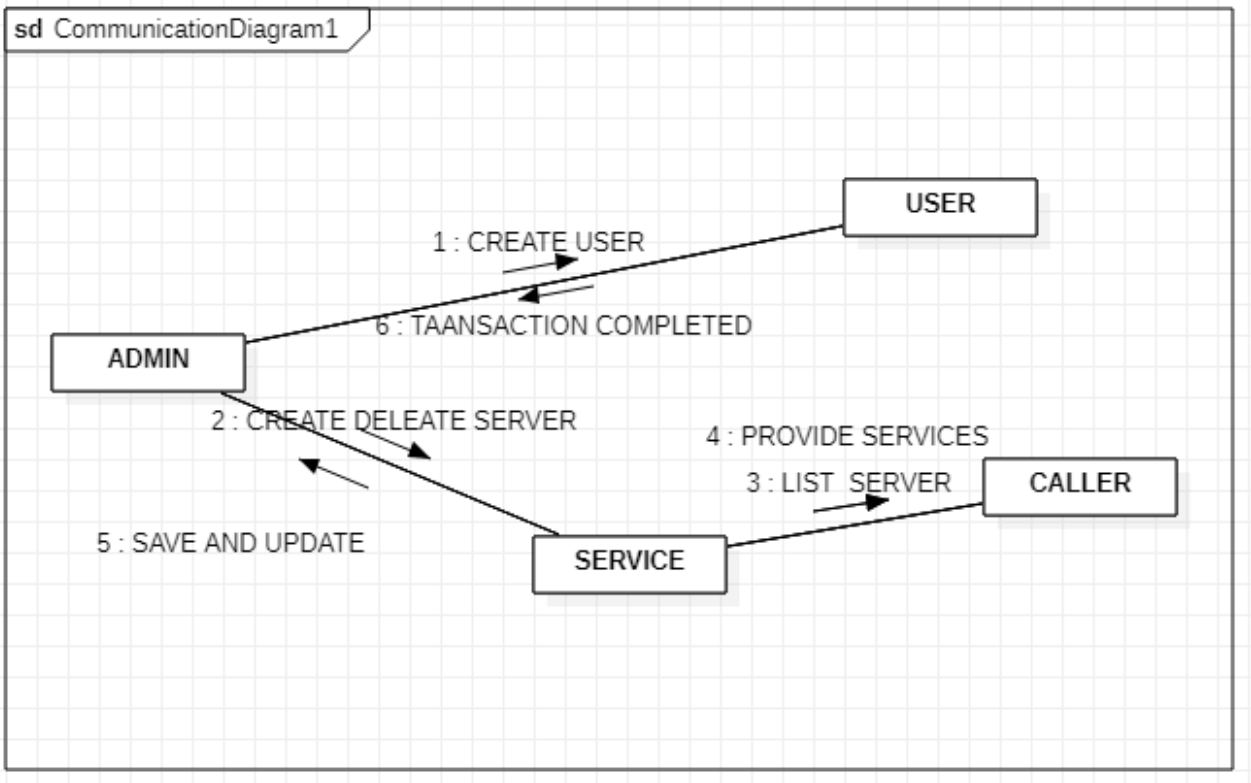
RAILWAY RESERVATION SYSTEM:



CAB BOOKING SYSTEM:



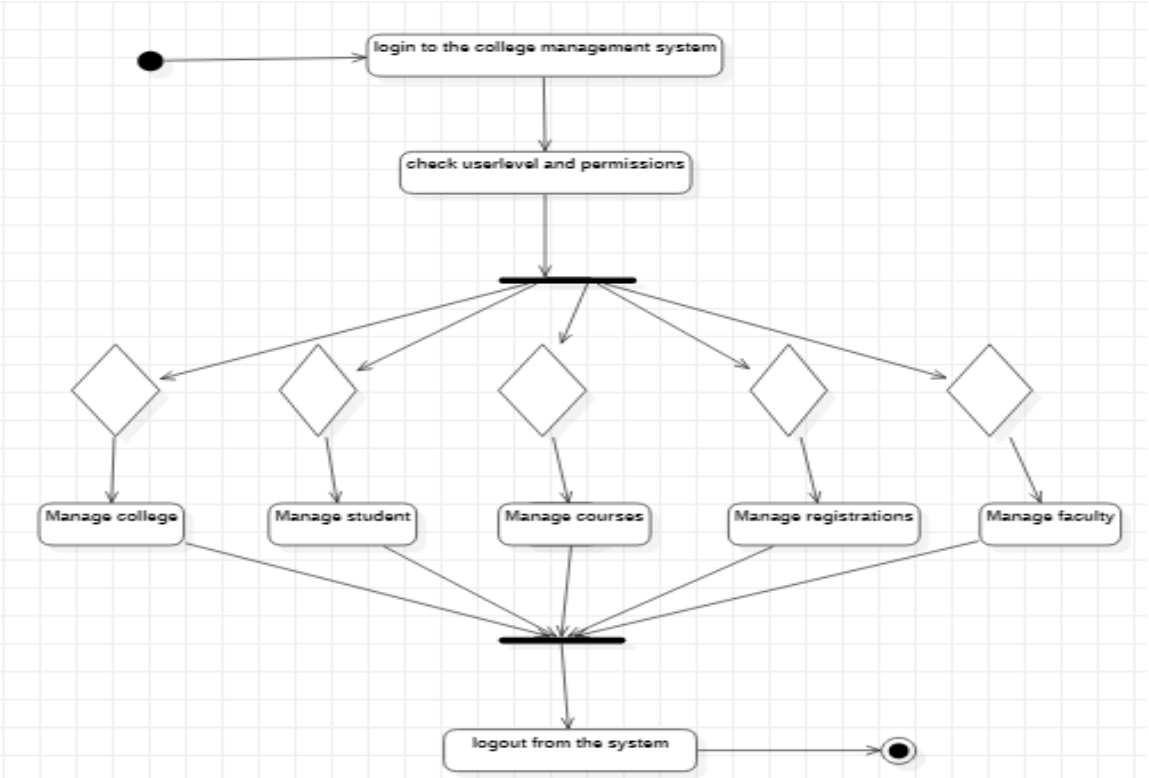
CUSTOMER SUPPORT SYSTEM:



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

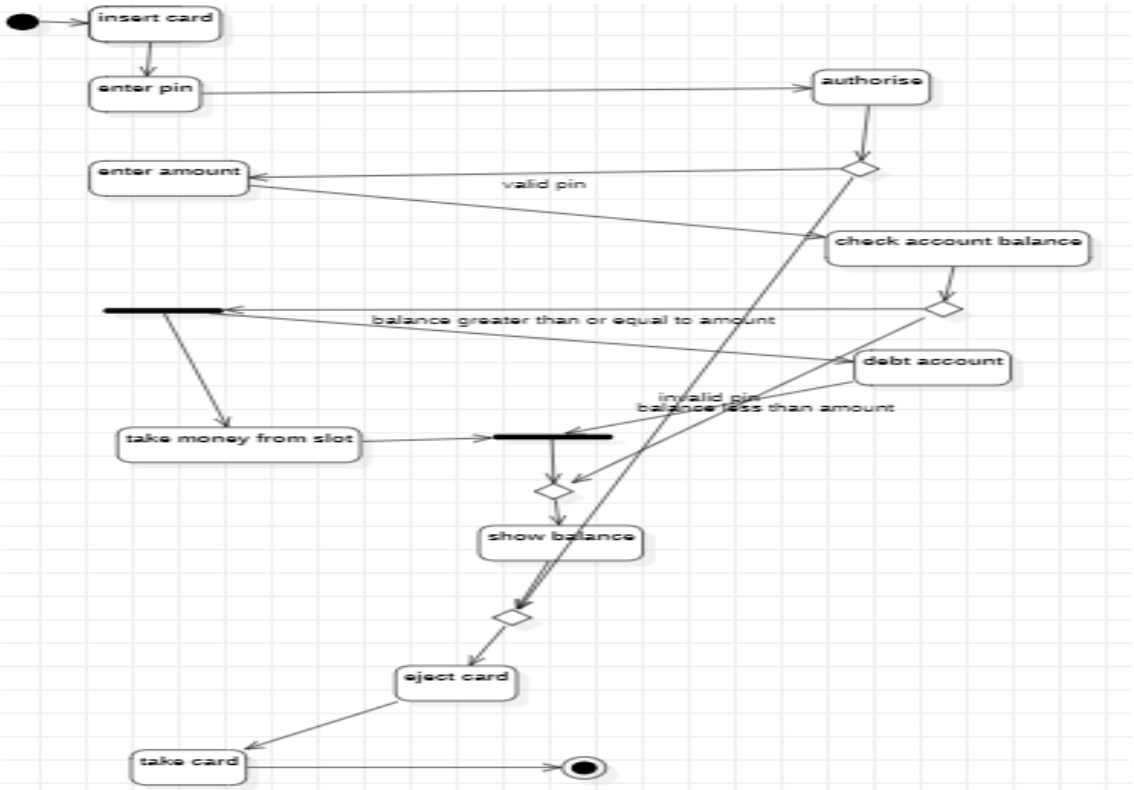
ACTIVITY DIAGRAMS:

COLLEGE MANAGEMENT SYSTEM:

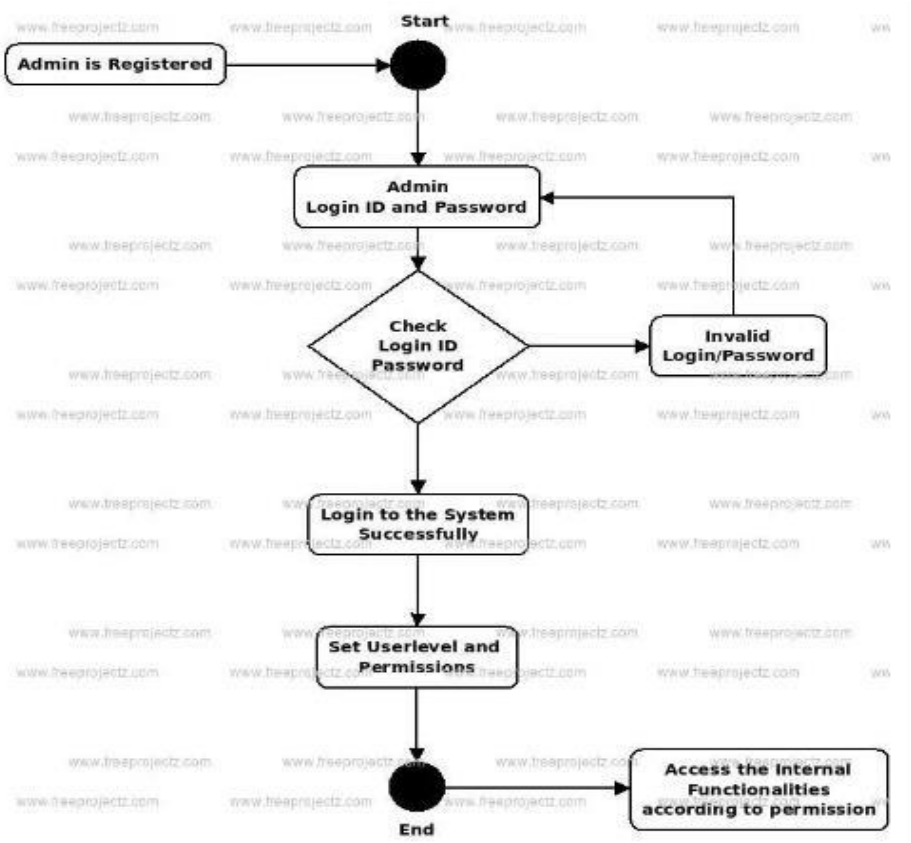


LIBRARY MANAGEMENT SYSTEM:

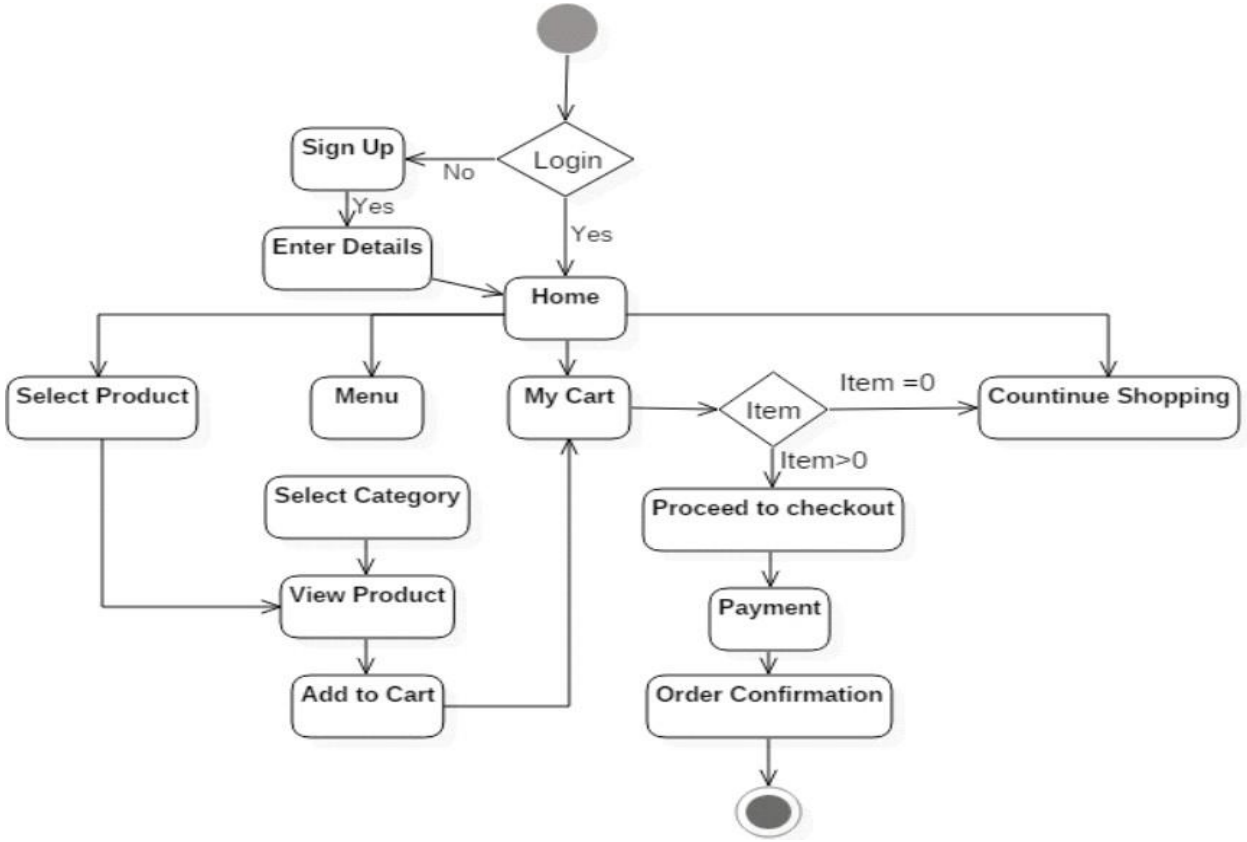
ATM:



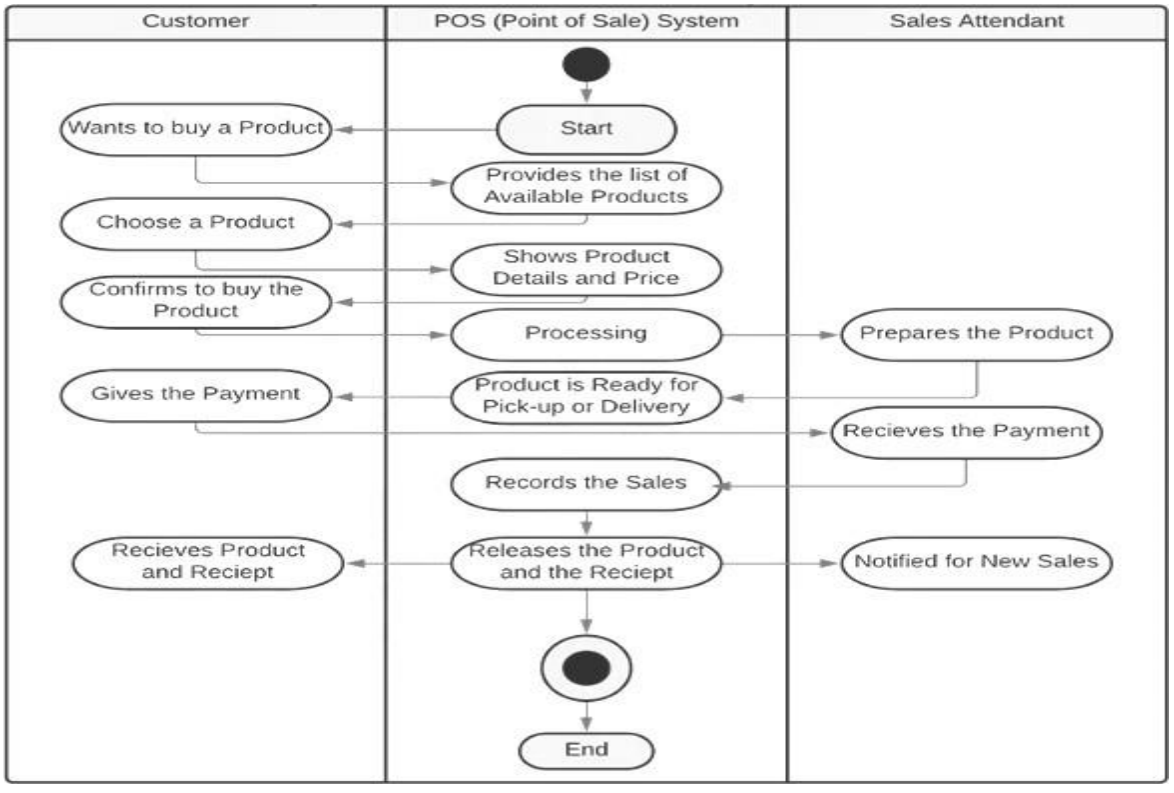
RAILWAY RESERVATION SYSTEM:



CUSTOMER SUPPORT SYSTEM:



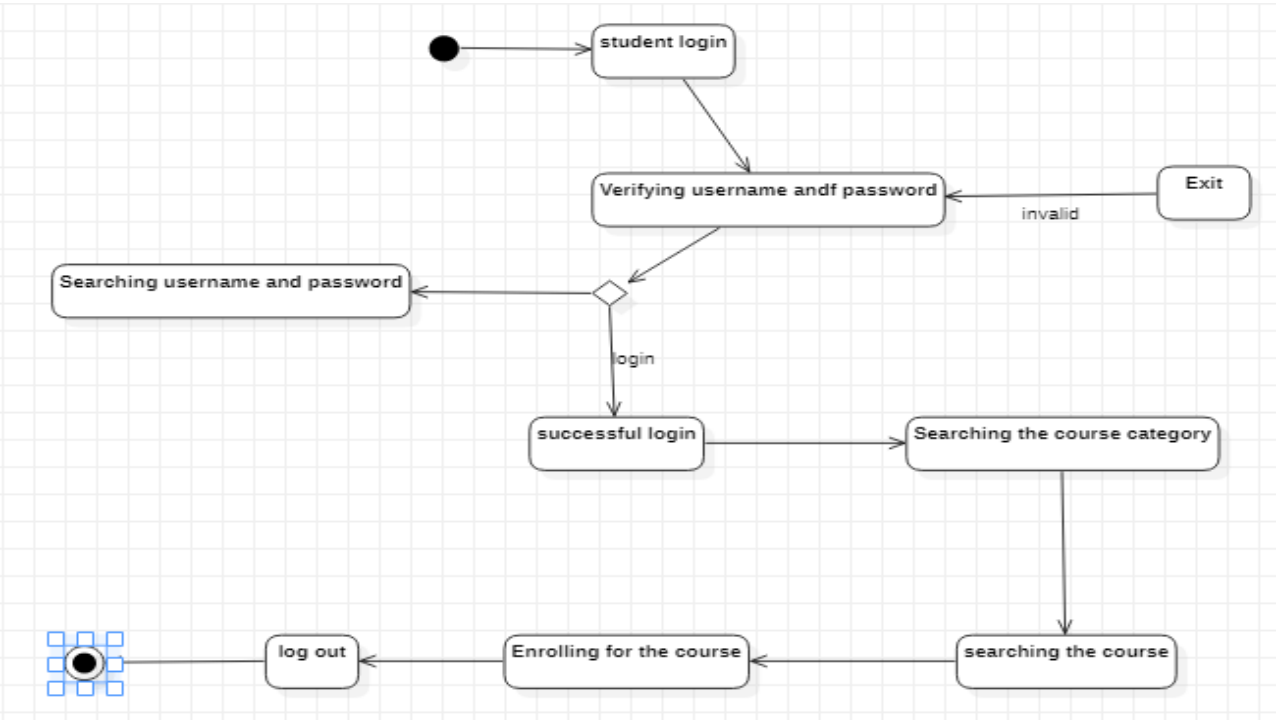
POINT OF SALES:



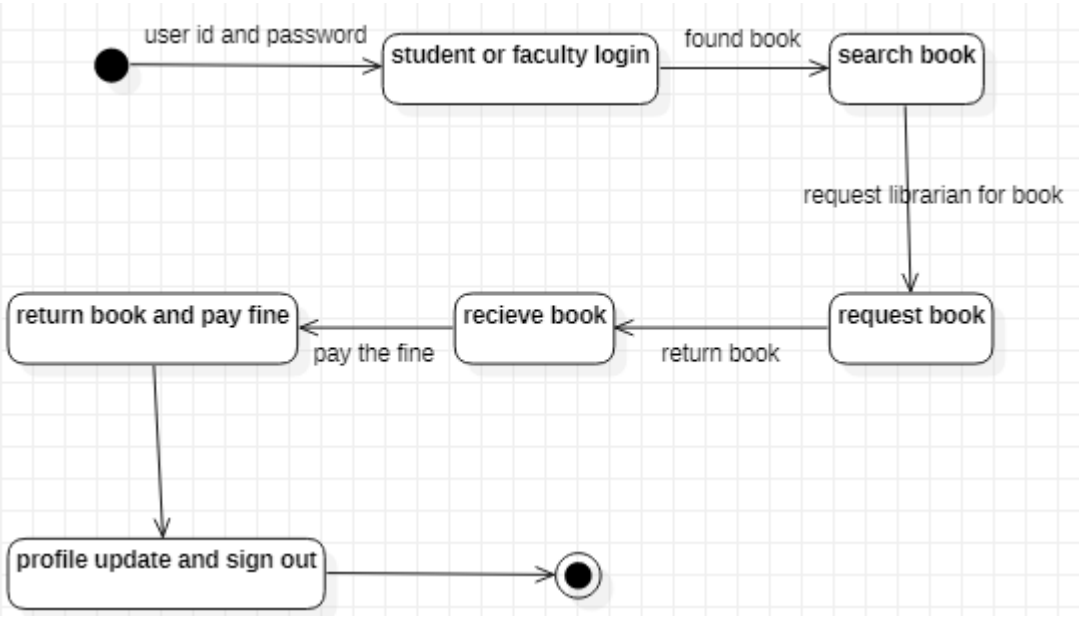
Week-9: For each case study given earlier, Construct State Chart Diagram in the following manner:

STATE CHART DIAGRAMS:

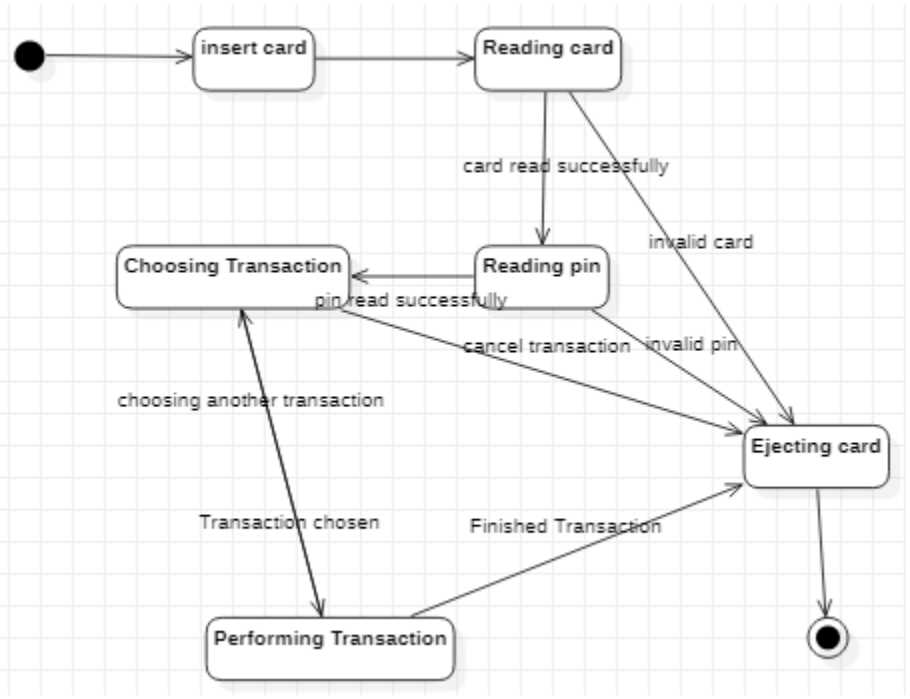
COLLEGE MANAGEMENT SYSTEM:



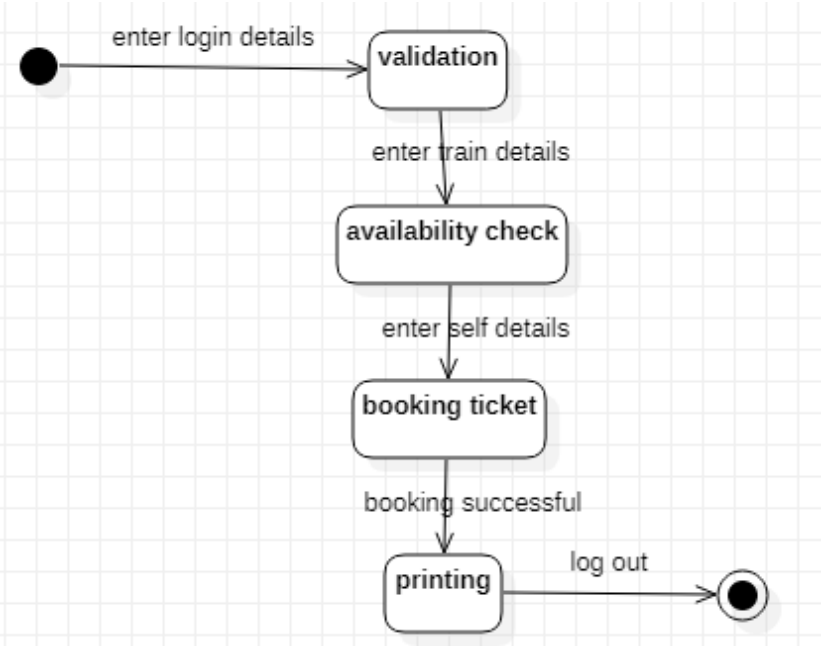
LIBRARY MANAGEMENT SYSTEM:



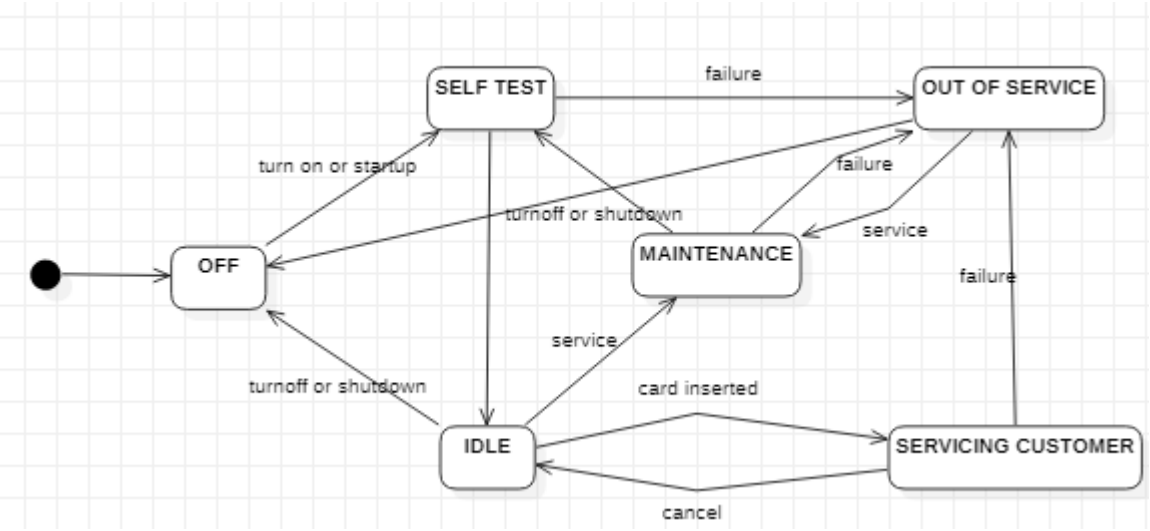
ATM:



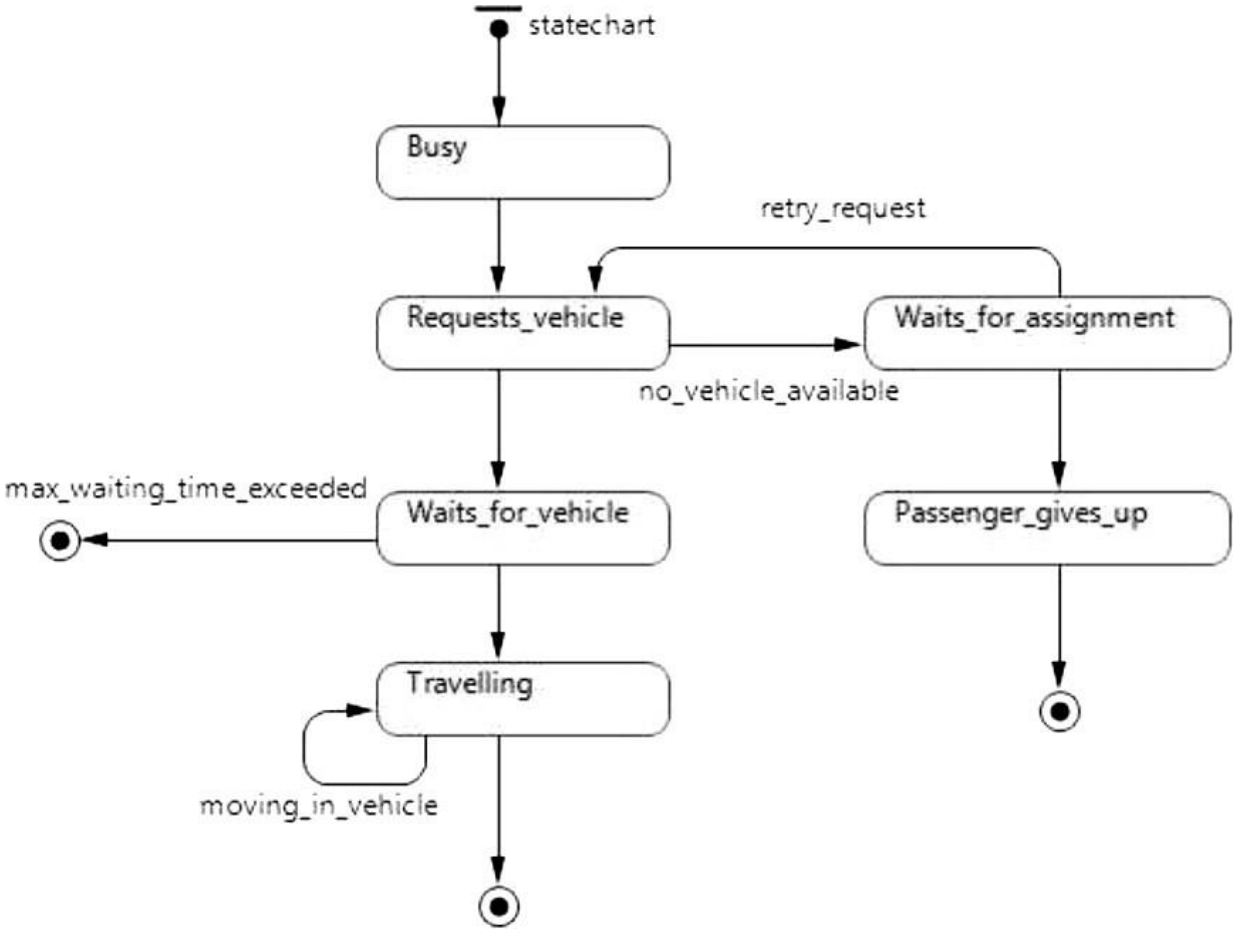
RAILWAY RESERVATION SYSTEM:



CUSTOMER SUPPORT SYSTEM:



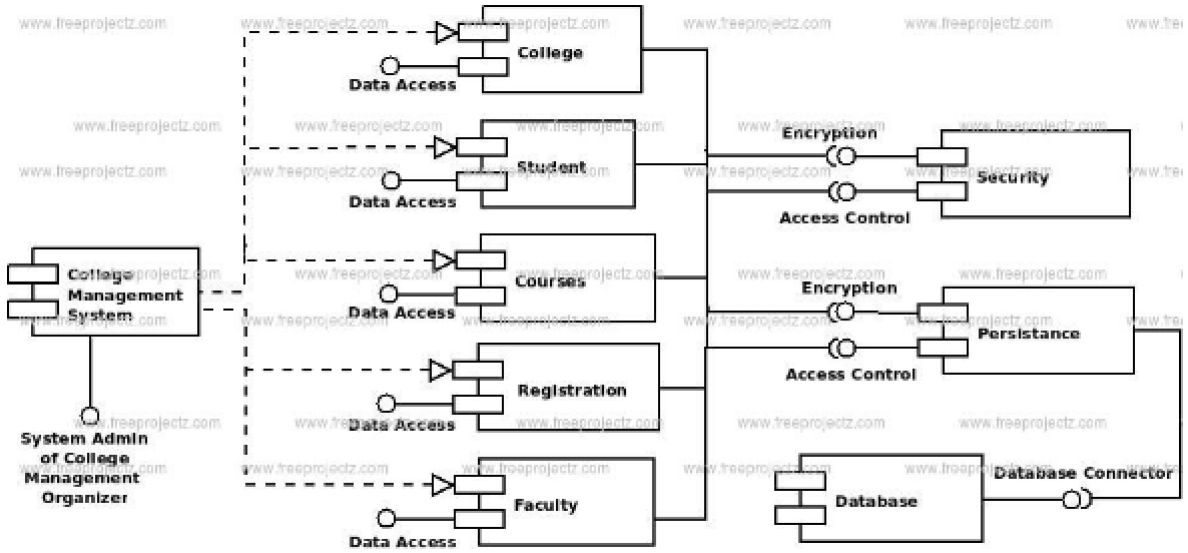
UBER CAB SYSTEM:



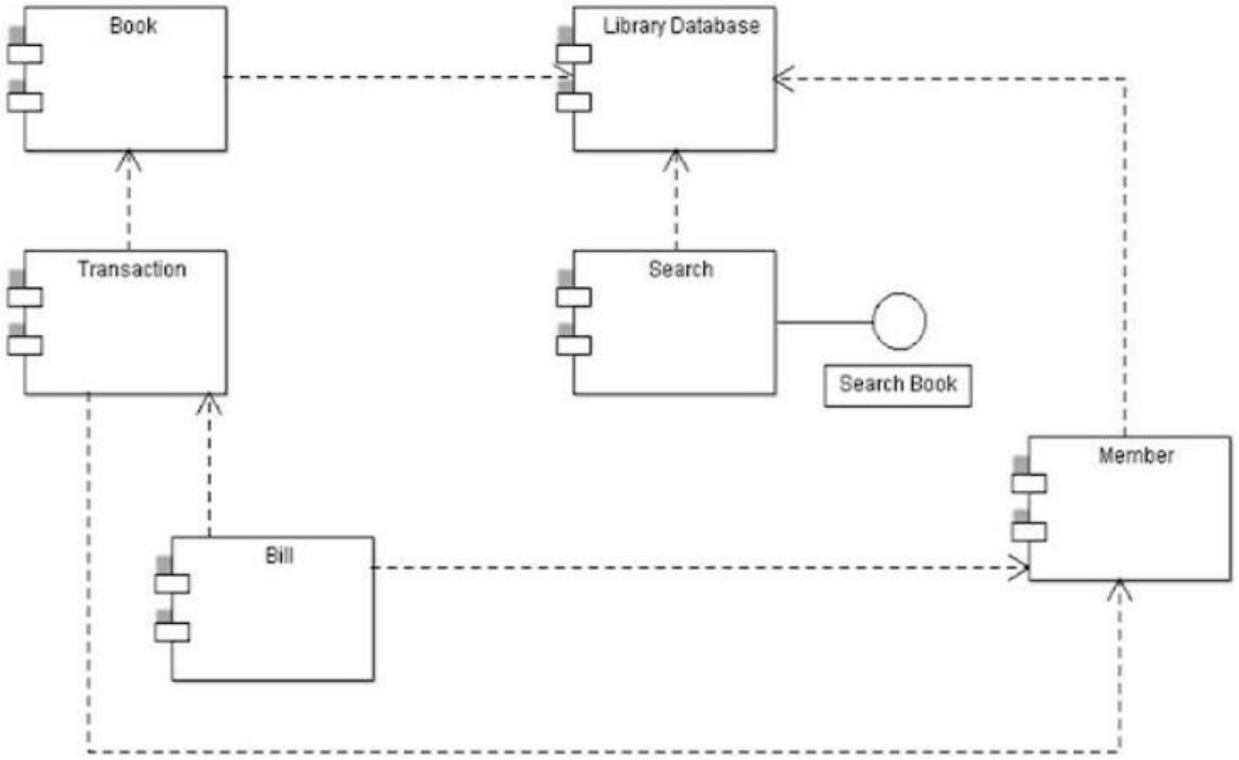
Week-10: For each case study given earlier, Construct Component Diagram in the following manner:

COMPONENT DIAGRAMS:

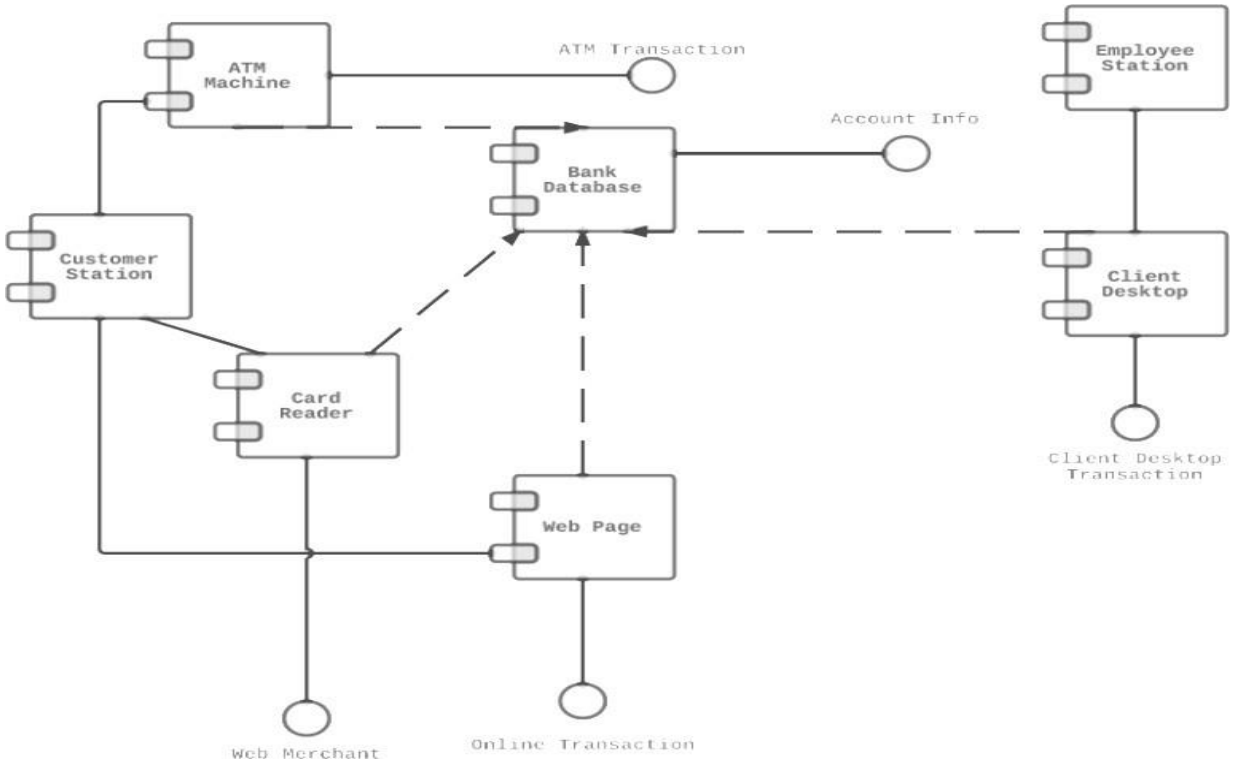
COLLEGE MANAGEMENT SYSTEM:



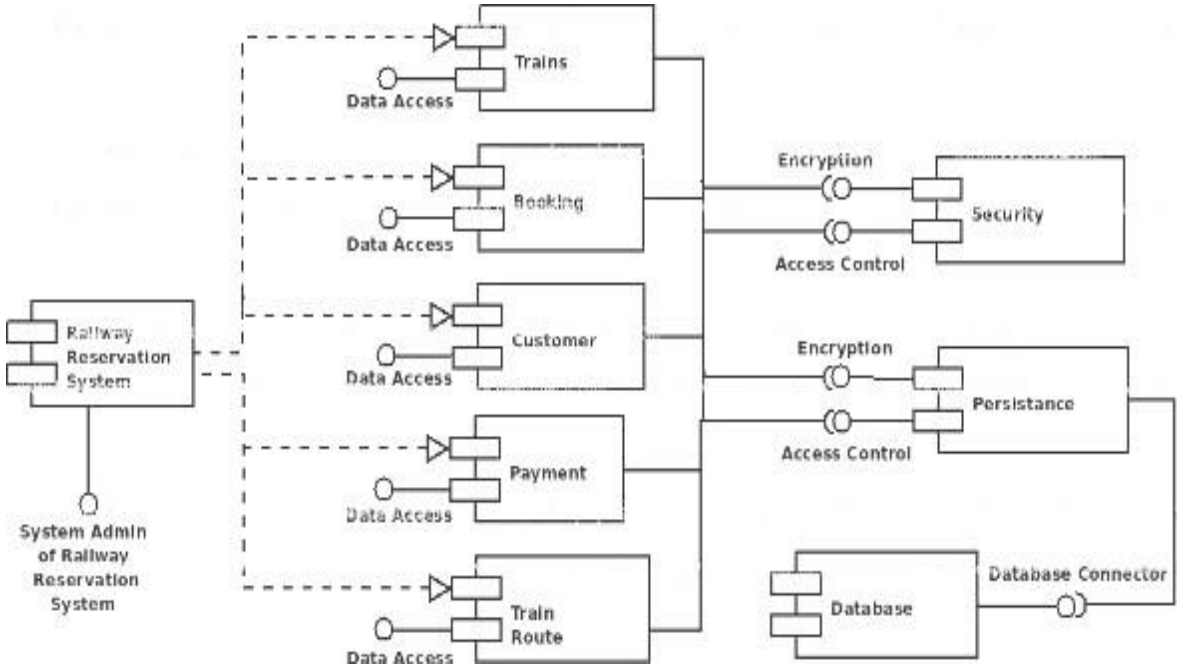
LIBRARY MANAGEMENT SYSTEM:



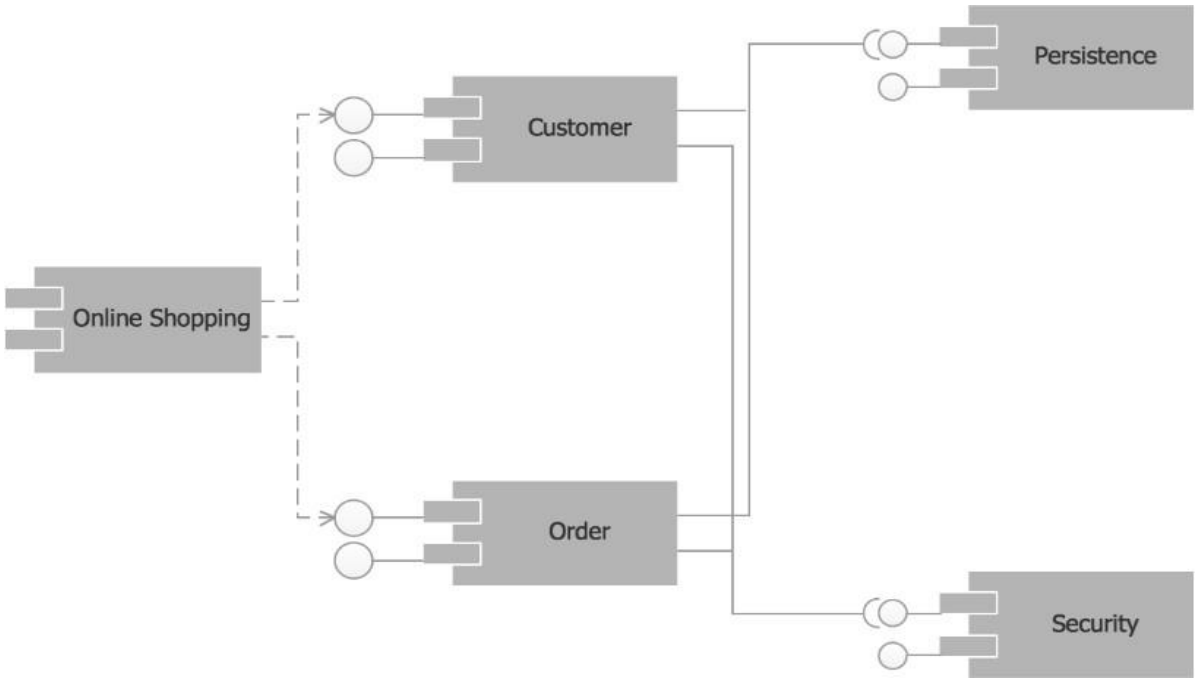
ATM:



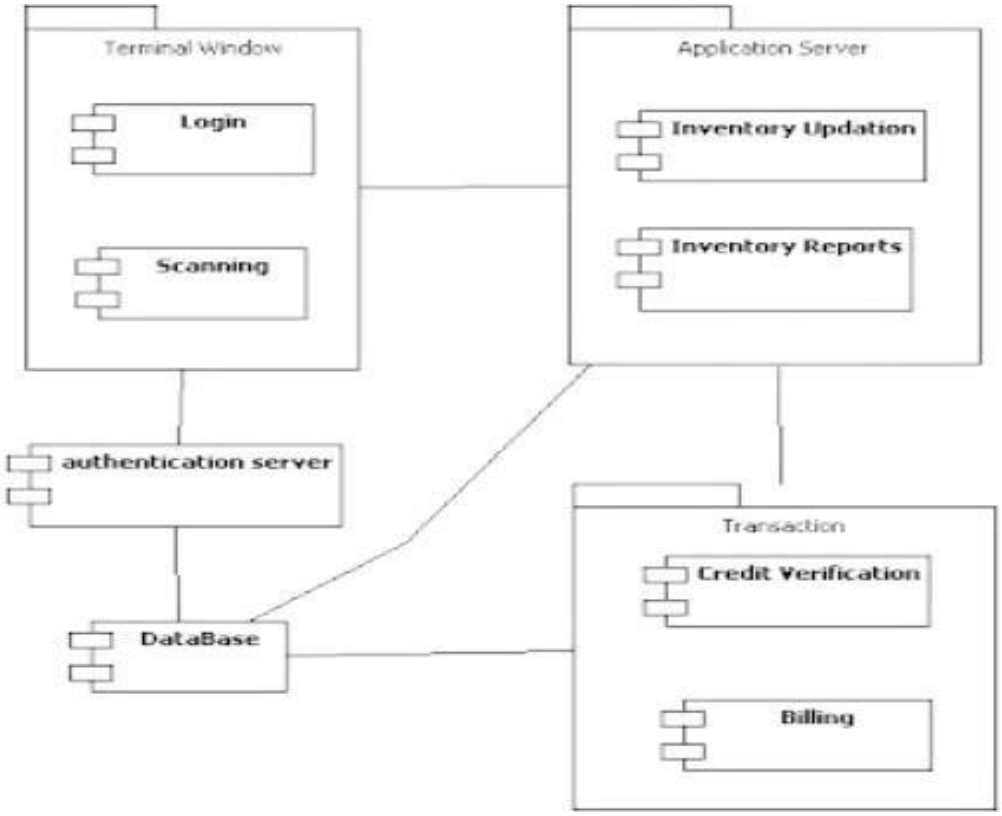
RAILWAY RESERVATION SYSTEM:



CUSTOMER SUPPORT SYSTEM:



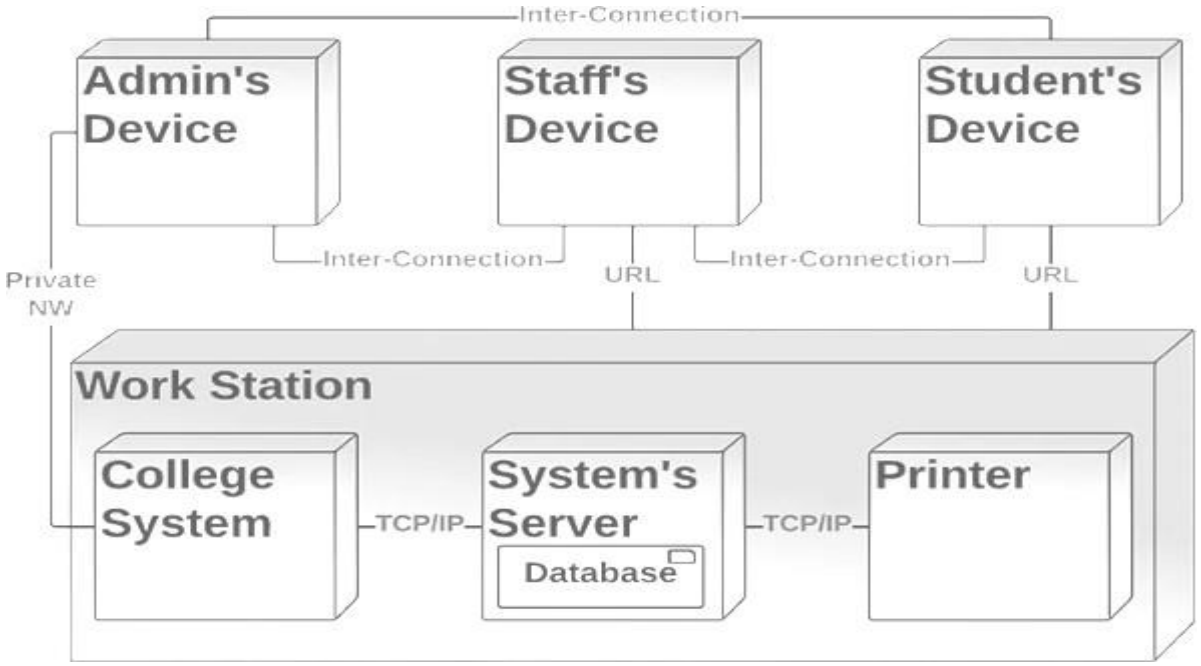
POINT OF SALES:



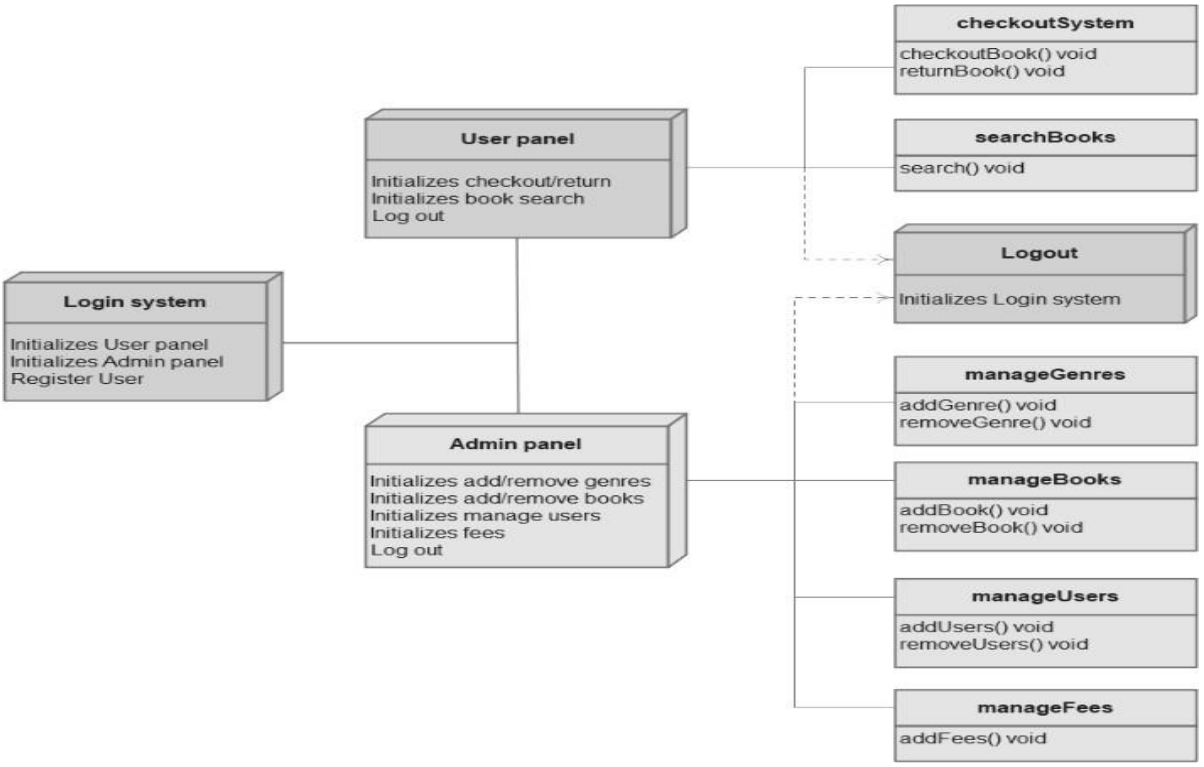
Week-11: For each case study given earlier, construct Deployment Diagram in the following manner:

DEPLOYMENT DIAGRAMS:

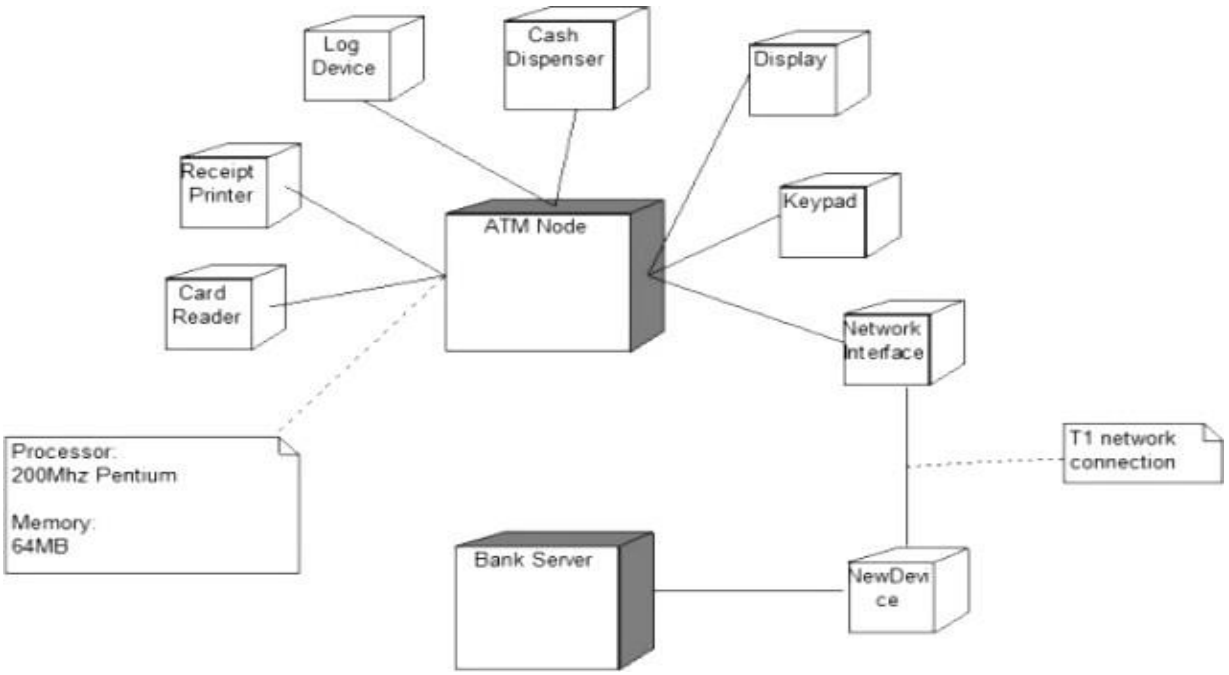
COLLEGE MANAGEMENT SYSTEM:



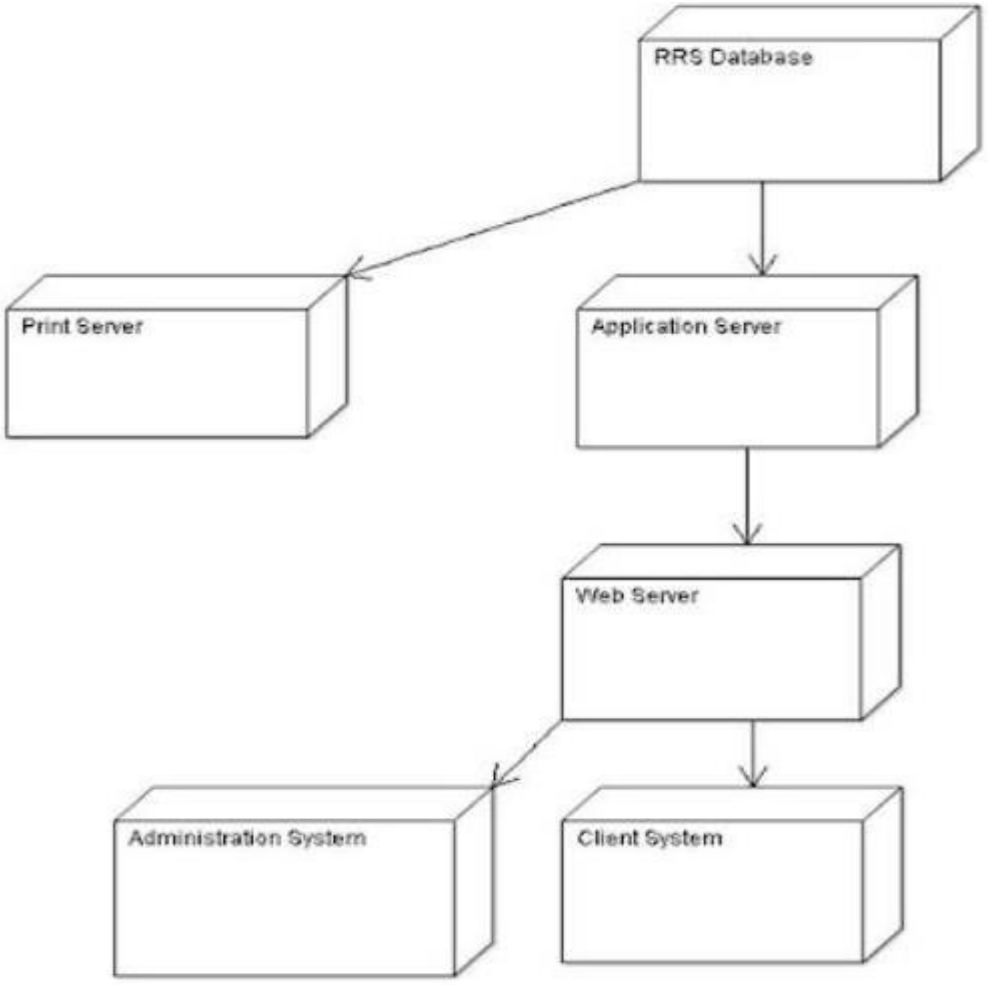
LIBRARY MANAGEMENT SYSTEM:



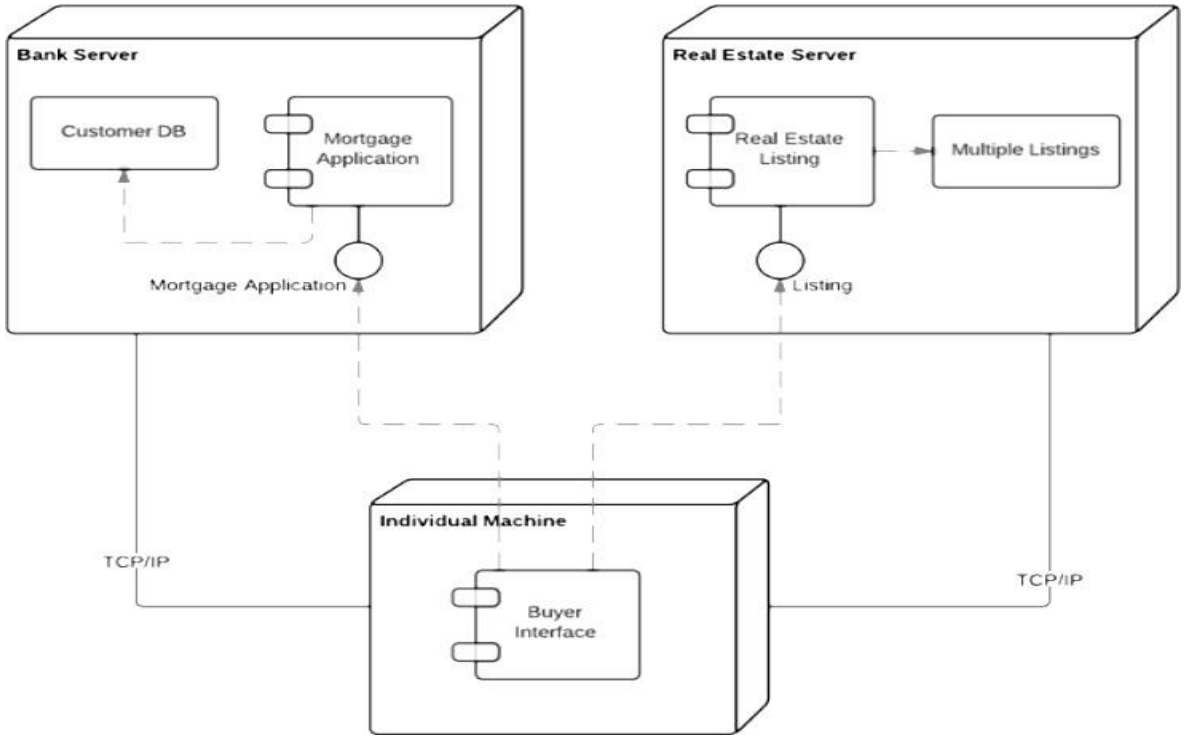
ATM:



RAILWAY RESERVATION SYSTEM:



CUSTOMER SUPPORT SYSTEM:



POINT OF SALES: