LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)
L.B. REDDY NAGAR, MYLAVARAM, KRISHNA DIST., A.P.-521 230.

DEPARTMENT
OF
INFORMATION TECHNOLOGY

Software Engineering Lab
20IT55
B.TECH IV SEMESTER

Mrs.S.Jyothi

Cycle-1

Analyze the Requirements for the following Case Studies.

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

AUTOMATED TELLER MACHINE (ATM)

Documentation:

Objective: In the above use case diagram we have two actors, customer and system performing different operations, represented by the use cases for the **ATM** system.

The use case diagram is initiatedwith the customer inserting the card

Flow of messages: in response to which the system displays the pin no screen. The customer enters the pin no. The system verifies the pin no and displays an options menu. The customer selects the option he wants (withdraw money in this case). Further the customer enters the amount he would like to withdraw. The system issues cash.

The use case diagram terminateswith the user collecting the cash and the card.

Alternative flows:In case the pin no entered by the customer is invalid, he is directed to re-entering the pin no.

In case the amount entered by the user is insufficient or out of limit, then the user is directed to re-enter the amount**.**

LIBRARY MANAGEMENT SYSTEM CASE STUDY

The Library Management System is an application for assisting a librarian in managing a book library in a university. The system would provide a basic set of features to add/update members. Add/update books and manage check in specifications for the systems based on the client's statement of need. Library management system is a typical management Information system (MIS)

Operations: – check out a book; – return a book; – add a book to library; – remove book from library; – get list of books by author or subject area; – get list of books checked out by particular borrower;

Actors of the system are:

1) Librarian: librarian can check availability of book, verify member issue book, calculate fine and return book
2) Member: Member can check availability of book, issue book and return books to the system

RAILWAY TICKET RESERVATION SYSTEM

A railway reservation system is a useful service which enables people book railway tickets online saving their time, energy and money. Today, nearly every railway company in the world offers a special reservation system in order to make the life of their passengers easier. Railways are still considered to be the most popular means of transportation, so more and more people take advantage of it every year. It is obvious that it is difficult and unpleasant to spend long hours at the booking office in a long queue in order to purchase a ticket.

The railway reservation system is a complicated service with a broad and strict structure available on the Internet and can be observed on the example of the website of the railway company. A client is required to log in the system and there he can find all the necessary information about the tickets, their price, terms, discounts and all possible routes and types of trains.

The system provides clients with information and enables them book tickets online and pay with the help of a credit card, so that there is no need to go to a booking office any more. This service saves time, money (it is possible to find a cheap route or class of a train) and the nerves of a customer. The topic of railway reservation system can be called an interesting one, because more and more companies offer the opportunity to book tickets online.

Actors of the System are:

Passenger,

Admin.

Cycle-2

Analyze the Requirements for the following Case Studies.

1) Point-of-Sale Terminal

2) Customer Support Service Operations

3) Cab Booking Service

POINT-OF-SALE TERMINAL

Point of Sale (POS) Terminal or Checkout. A retail POS system typically includes a computer, monitor, keyboard, barcode scanners, weight scale, receipt printer, credit card processing system, etc. and POS terminal software.

Actors of the system are:

1) Customer

2) Clerk

Checkout use case involves Customer, Clerk and Credit Payment Service actors and includes scanning items, calculating total and taxes, payment use cases.

Checkout use case requires a Customer actor, hence the 1 multiplicity of Customer.

Clerks can only participate in a single Checkout use case.

Credit Payment Service can participate with many Checkout use cases at the same time.

Checkout use cases may not need Credit Payment Service (for example, if payment is in cash)

CUSTOMER SUPPORT SERVICE OPERATIONS

The Specific objectives for customer support system are the functions associated with the customer the system should include all the functions associated with customer, technician and administrator. Products for the customer from order entry to arrival of shipment.

Customer Service support service operations has use cases

- Customer
- Technician
- Administrator

CAB BOOKING SERVICE

Online Cab Booking Service deals with an online system designed for booking cabs as per the requirements of the customers at their convenience. options to book a cab by entering details like their journey date and time, origin, pick-up point, destination and the drop-off point they need to reach.

The proposed Online Cab Booking system ensures that the users can book the cab as per their requirements by logging on to the website. It allows users to book their cabs online, manage their bookings and cancel their bookings at any point of time. So as to communicate with him. Regular updates are provided to the customer so that they are aware of their bookings, driver details, and booking status. The user can also drop in their suggestions or queries in the feedback form. Advantages: It enhances business processes since it makes use of internet technology to increase their profits. The software acts as a 24/7 office due to its all-time availability. It increases the efficiency of the system in offering quality services to its customers

Actors of the System are:

1) Passenger
2) Service provider
3) Drive

CYCLE-3:
BASICS OF UML

1.Introduction to Unified Modeling Language (UML)

What is UML?

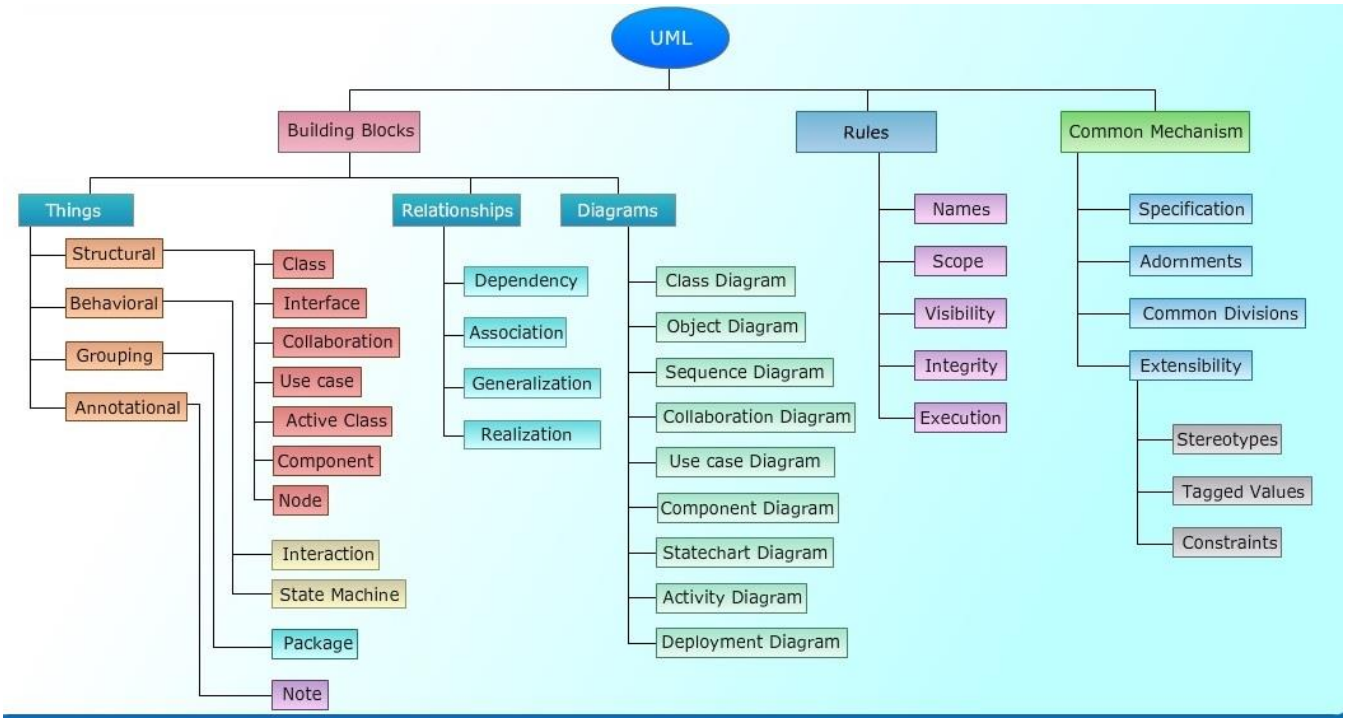
"The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the software systems.

The UML is a visual modeling language that

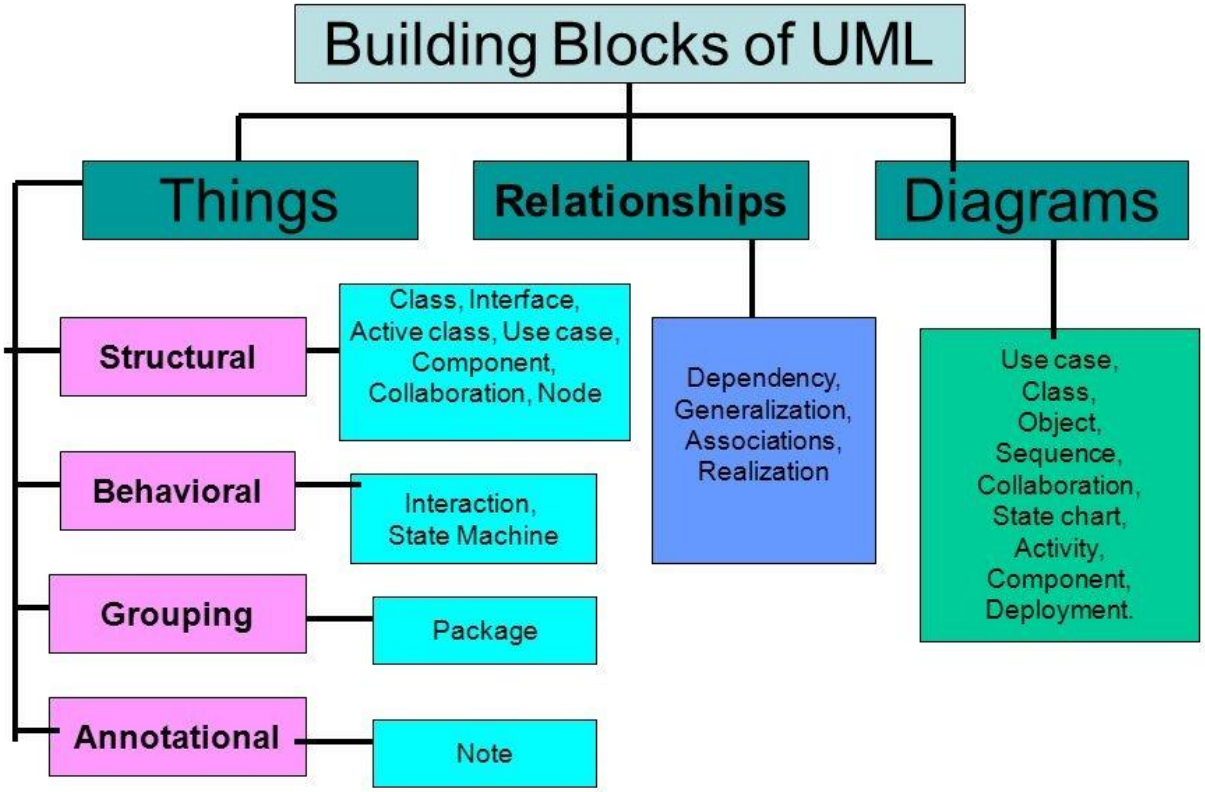
- create blueprints
- easy-to-understand
- provides a mechanism to effectively
- communicate these visions with others.

A Conceptual Model of the UML:

To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements: the UML basic building blocks, the rules that dictate how those building blocks may be put together, and some common mechanisms that apply throughout the UML.



Building Blocks of the UML:



The vocabulary of the UML encompasses three kinds of building blocks:

1. Things
2. Relationships
3. Diagrams

Things: Things are the abstractions that are first-class citizens in models. There are four kinds of things in the UML:

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

1. Structural Things:

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. In all, there are seven kinds of structural things.

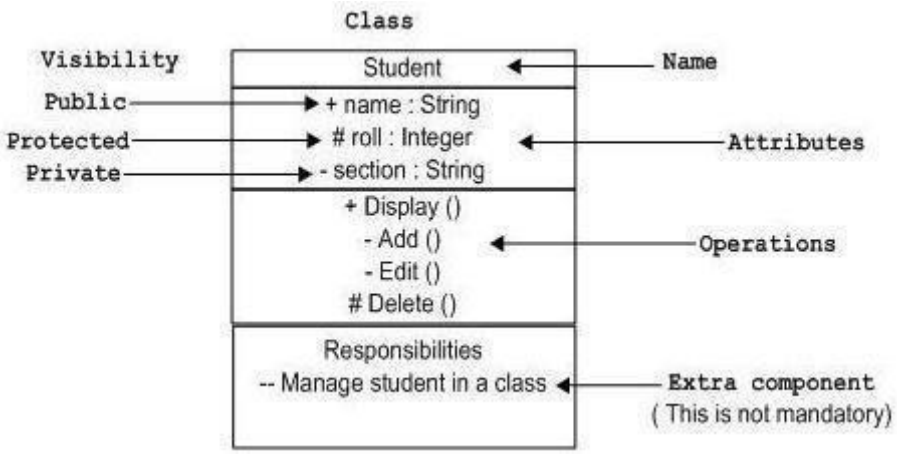
Graphical notations used in structural things are the most widely used in UML. These are considered as the nouns of UML models. Following are the list of structural things.

- Class
- Interface
- Collaboration
- Use case
- Components
- Nodes

Class:

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.



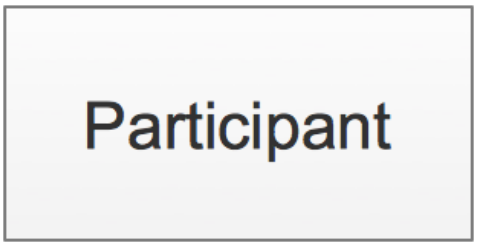
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.

Interface:

Interface is an abstract class that defines a set of operations that the object of the class associated with this interface provides to other objects.

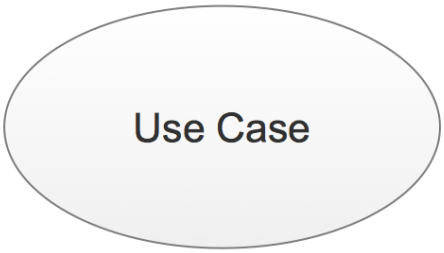
Collaboration:

Collaboration determines interactions between the elements.



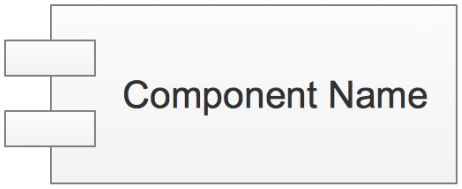
Use case:

Use case is a description of the system behavior on the request from the outside of this system.



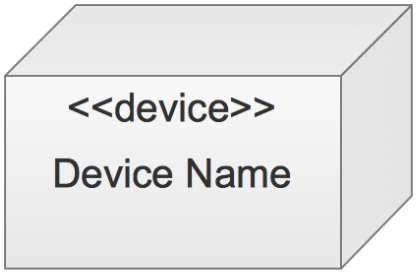
Component:

Component describes the physical part of the system.



Node:

Node is a resource available during the run time.



Behavioral things (dynamic part of the model):

Interaction:

Interaction is a set of messages that the elements exchange for execution the tasks.

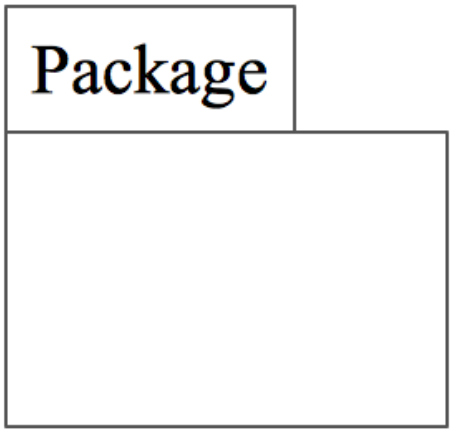
State machine:

State machine defines the states of the object that go in response to the events.

Grouping things

Package:

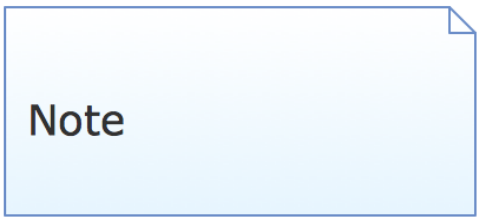
Package groups the classes and other packages.



Annotational things

Note:

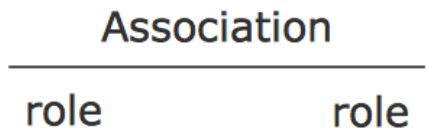
Note is a textual explication.



Relationship :

Association:

Association is a relationship that connect two classes.



Dependency:

Dependency is a relationship when some changes of one element of the model can need the change of another dependent element.

Generalization:

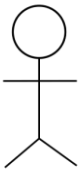
Generalization is an association between the more general classifier and the more special classifier.

Realization:

Realization is a relationship between interfaces and classes or components that realize them.

Actor Notation:

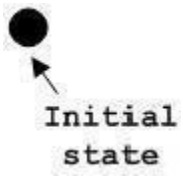
An actor can be defined as some internal or external entity that interacts with the system.



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

Initial State Notation:

Initial state is defined to 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.

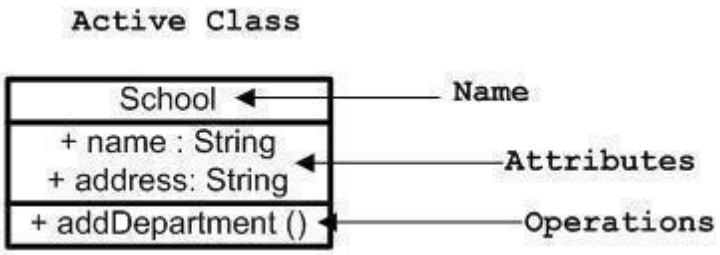
Final State Notation:

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.

Diagrams:

- **Class diagram**
- **Object diagram**
- **Use case diagram**
- **Sequence diagram**
- **Collaboration diagram**
- **Activity diagram**
- **Statechart diagram**
- **Deployment diagram**
- **Component diagram**

Diagrams in the UML

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). In theory, a diagram may contain any combination of things and relationships. In practice, however, a small number of common combinations arise, which are consistent with the five most useful views that comprise the architecture of software – intensive systems. For this reason, the UML includes nine such diagrams:

1. Class Diagram: A class diagram shows a set of classes, interfaces, and collaborations and their relationships. These diagrams are the most common diagrams found in modeling object-oriented systems. Class diagrams address the static design view of a system.

2. Object Diagram: An object diagram shows a set of objects and their relationship. Object diagrams represent static snapshots of instances of the things found in class diagrams. These diagrams address the static design view or static process view of a system as do class diagrams, but from the perspective of real or prototypical cases.

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

4. Sequence Diagram: A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. It consists of a set of objects and their relationships, including the messages that may be dispatched among them. Sequence diagrams address the dynamic view of a system.

5. Collaboration Diagram: A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. It consists of a set of objects and their relationships, including the messages that may be dispatched among them. Collaboration diagrams address the dynamic view of a system. Sequence diagrams and collaboration diagrams are isomorphic, meaning that you can take one and transform it into the other.

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

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

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

9. Deployment Diagram: A deployment diagram shows the configuration of run-time processing nodes and the components that live to them. Deployment diagrams address the static deployment view of architecture. They are related to component diagrams in that a node typically encloses one or more components.

Install Software

- Staruml or
- Rational Rose or
- Umbrello or
- Gliffy Diagram

How to draw a 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 diagrams.

Class diagrams have a 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 represents the whole system.

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

- Thenameoftheclassdiagramshouldbemeaningfultodescribetheaspectofthesystem.
- Each element and the relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified.
- For each class a 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 is drawn on plain paper and reworked 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 a customer can have multiple orders.
- The Order class is an abstract class and it has two concrete classes (inheritance relationship) Special Order and Normal Order.
- The two inherited classes have all the properties as the Order class. In addition they have additional functions like dispatch () and receive ().

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 consisting 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 closer 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.
- Understandobjectbehaviourandtheirrelationshipfrompracticalperspective.

How to draw an 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 the same basic elements but in different forms. In class diagram elements are in abstract form to represent the blueprint and in object diagrams 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 considerobject diagrams then we can have an unlimited number of instances which are unique in nature. So only those instances are considered which are having an 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 have important data and association.
- Second, consider only those instances which will cover the functionality.
- Third, make some optimization as the numbers are unlimited.

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

- Object diagrams consistof objects.
- The link in the 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.
- 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 the Class Diagram. The following diagram is an instance of the system at a particular time of purchase. It has the following objects

- Customer
- Order
- Special Order
- Normal Order

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

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 the 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 have some values.

Forordersthevaluesare12,32, 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 have a 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 behavior. 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 systemsPurpose:

ComponentdiagramisaspecialkindofdiagraminUML.Thepurposeisalsodifferentfromall 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 is 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 a Component Diagram?

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

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

implementation details.

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

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

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

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

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

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

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

In the following diagram four files are identified and their relationships are produced. Component diagrams 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.

UMLismainlydesignedtofocusonsoftwareartifactsofasystem.Butthesetwodiagramsare 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 a 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 systems. 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 the 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 behavior. To clarify a bit in details, dynamic behavior means the behavior of the system when it is running /operating.

So only static behavior is not sufficient to model a system; rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagrams are one of them. Now as we have discussed 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 consist 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 a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration and State chart) alsohave the same purpose. So we will look into some specific purpose which will distinguish it from the 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 modeled 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.
- Showing the interacting among the requirements are actors.

How to draw a 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 system functionalities written in an organized manner. Now the second thing which is relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

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

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

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

- Thenameofausecaseisveryimportant.Sothenameshouldbechoseninsuchawayso that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use noteswhenever required to clarify some important points.

Thefollowingisasampleusecasediagramrepresentingtheordermanagementsystem.Soif we look into the diagram then we will find three use cases (Order, Special-order and Normal Order) and one actor which is customer.

TheSpecialOrderandNormalOrderusecasesareextendedfromOrderusecase.Sotheyhave 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 behavior 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:

The purposes of interaction diagrams are to visualize the interactive behavior 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 the interaction diagram can be described as:

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

How to draw an Interaction Diagram?

As we have already discussed, the purpose of interaction diagrams is 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 a sequence diagram and the other is a collaboration diagram. The sequence diagram captures the time sequence of message flow from one object to another and the collaboration diagram describes the organization of objects in a system taking part in the message flow.

So the following things are to 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 modeling an order management system. The first diagram is a sequence diagram and the second is a collaboration diagram.

The Sequence Diagram:

The sequence diagram has four objects (Customer, Order, Special-order and Normal Order).

ThefollowingdiagramhasshownthemessagesequenceforSpecialOrderobjectandthesame can be used in case of a Normal Order 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 sends Order () which is a method of Order object. The next call is confirming () which is a method of a Special-order object and the last call is Dispatch () which is a method of a Special-order 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 a collaboration diagram. It shows the object organization as shown below. Here in the 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 a sequence diagram is used and if organization is required then a collaboration diagram is use

UML State chart 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 State chart 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 the next chapter, is a special kind of a State chart diagram. A State chart diagram defines which states it is used to model the lifetime of an object.

Purpose:

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

State chart 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 a State chart diagram is to model the time of an object from creation to termination.

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

Following are the main purposes of using State chart diagrams:

- To model the dynamic aspect of a system.
- To model the time 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 a State chart Diagram?

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

State chart 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 State chart 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 State chart diagram where the state of an Order object is analyzed.

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

During the life cycle of an object (here, order object) it goes through the following states and there may be some abnormalities 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 flowchart 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 types of flow control by using different elements like fork, join etc.

Purpose:

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

It does not show any message flow from one activity to another. Activity diagram is sometimes considered a flow chart. Although the diagrams look like a flow chart, they are not. It shows different flows 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 an Activity Diagram?

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

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

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

- Activities
- Association
- Conditions
- Constraints

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

The following is an example of an activity diagram for an 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 a 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.

Cycle-4

For each case study given earlier, Construct a Use Case Diagram for the following:

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

1. **Use case diagram for Automated Teller Machine (ATM)**

Documentation:

Objective: In the above use case diagram we have two actors, customer and system performing different operations, represented by the use cases for the atm system.

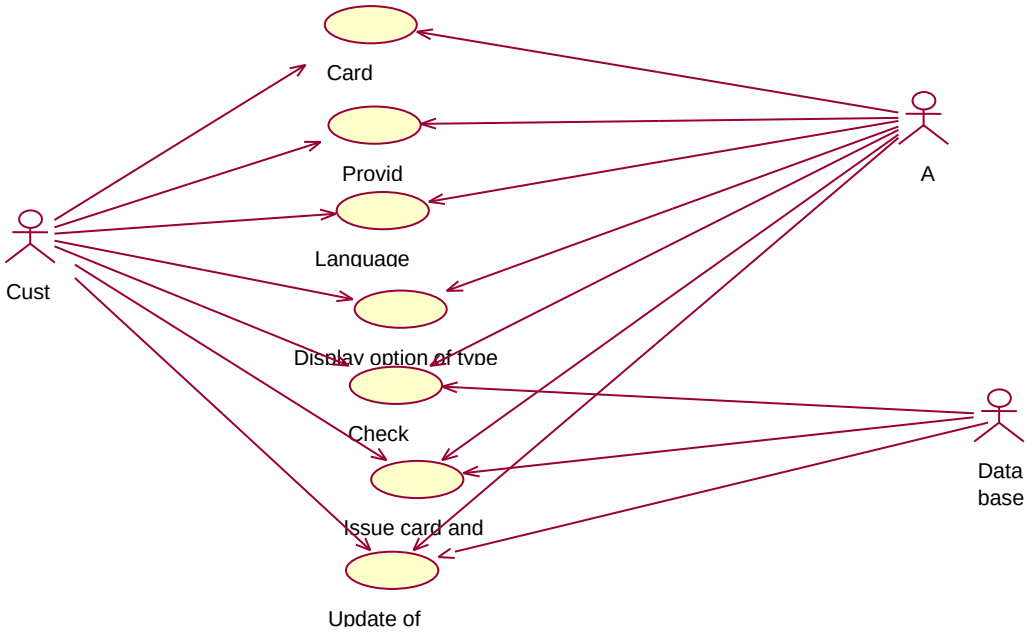
The use case diagram is initiatedwith the customer inserting the card

Flow of messages: in response to which the system displays the pin on screen. The customer enters the pin no. The system verifies the pin no and displays an options menu. The customer selects the option he wants (withdraw money in this case). Further the customer enters the amount he would like to withdraw. The system issues cash.

The use case diagram terminateswith the user collecting the cash and the card.

Alternative flows:In case the pin no entered by the customer is invalid, he is directed to re-entering the pin no.

In case the amount entered by the user is insufficient or out of limit, then the user is directed to re-enter the amount**.**



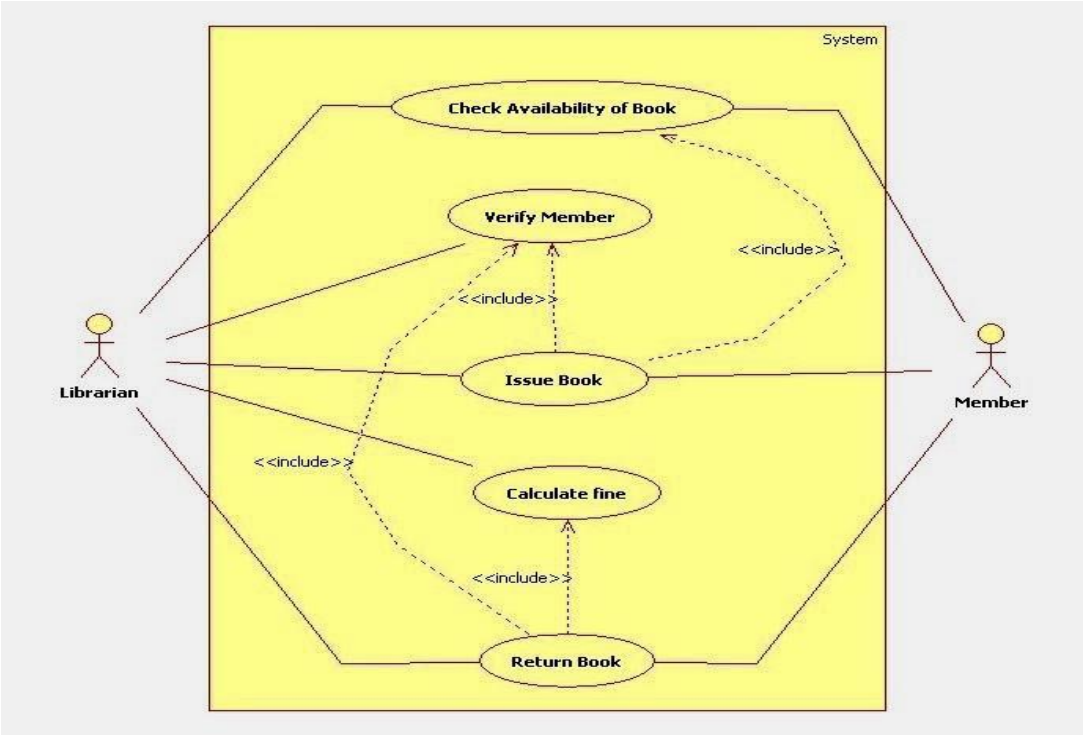
USE CASE DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

Use cases represent a typical set of scenarios that help to structure, relate and understand the essential requirement. A use case diagram is a diagram that helps a system analyst to discover the requirement of a target system from the user's perspective.

Use case diagrams can be used to describe the functionality of a systemin a horizontal way. Use case diagrams are used to represent functionality of the system from a top-down perspective.

Actors of the system are:

1) Librarian: librarian can check availability of book, verify member issue book, calculate fine and return book
2) Member: Members can check availability of books, issue books and return booksto the system.

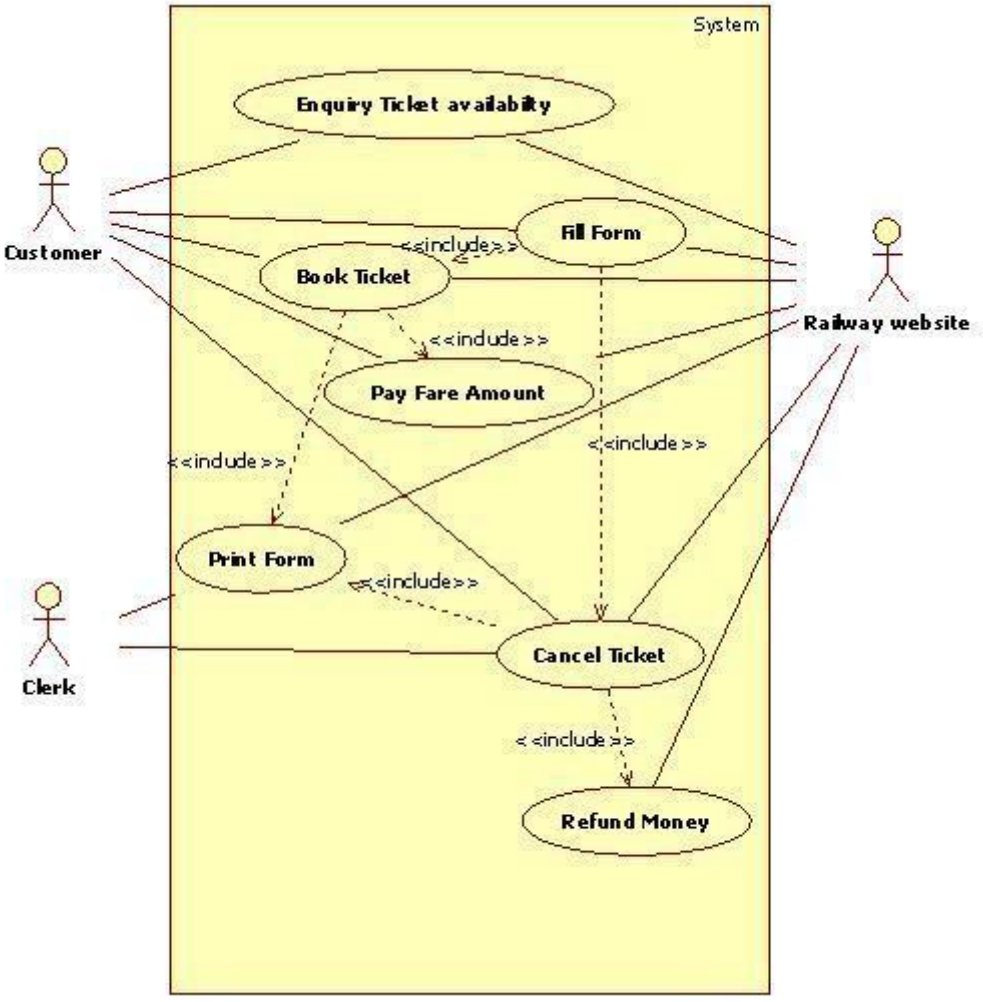


USE CASE DIAGRAM FOR RAILWAY TICKET RESERVATION SYSTEM

The system provides clients with information and enables them to book tickets online and pay with the help of a credit card, so that there is no need to go to a booking office any more. This service saves time, money (it is possible to find a cheap route or class of a train) and the nerves of a customer. The topic of the railway reservation system can be called an interesting one, because more and more companies offer the opportunity to book tickets online.

Actors of the System are:

- Customer
- Clerk
- Railway website.



USE CASE DIAGRAM FOR POINT-OF-SALE TERMINAL

Point of Sale (POS) Terminal or Checkout. A retail POS system typically includes a computer, monitor, keyboard, barcode scanners, weight scale, receipt printer, credit card processing system, etc. and POS terminal software.

Actors of the system are:

1) Customer

2) Retailer

3) Supplier

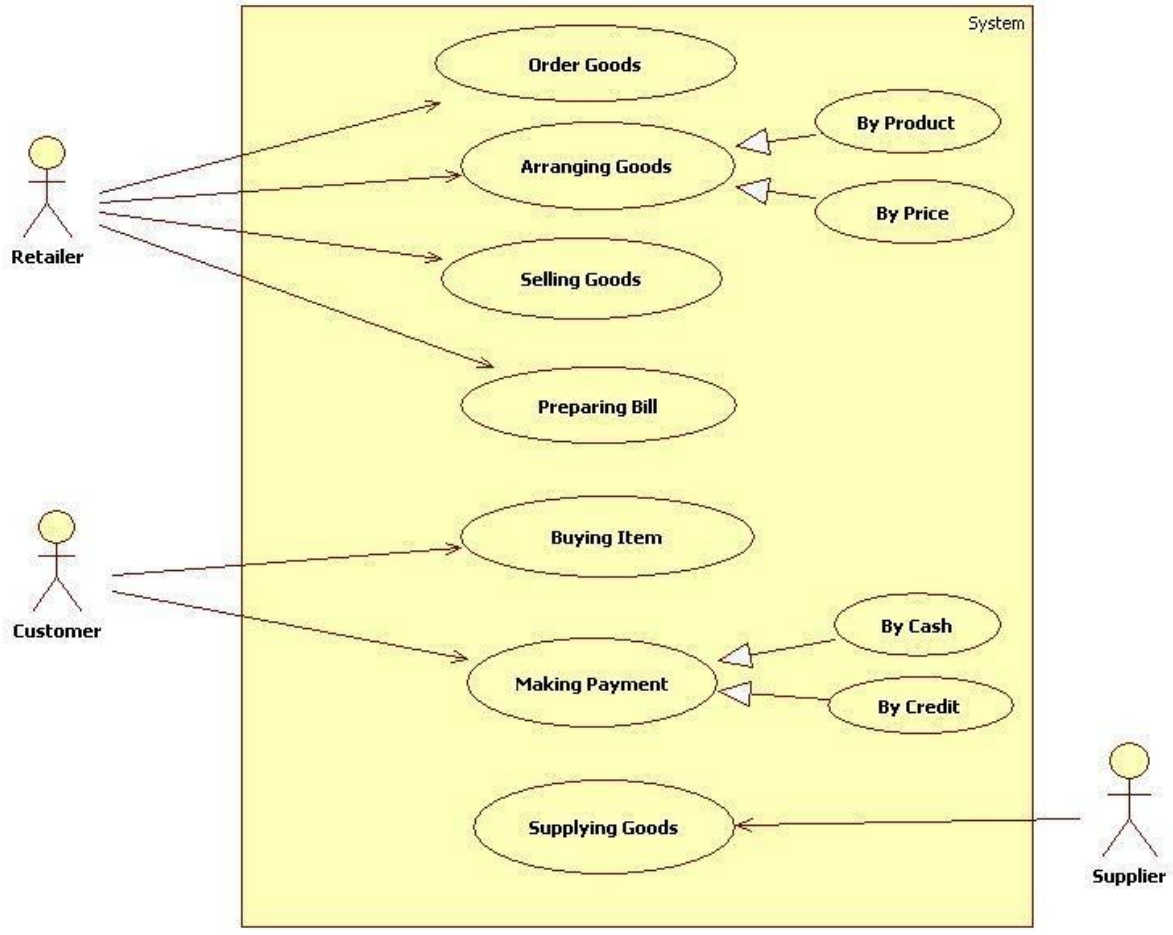
Checkout use case involves Customer, Clerk and Credit Payment
Service actors and includes scanning items, calculating total and taxes, payment use cases.

Checkout use case requires a Customer actor, hence the 1 multiplicity of Customer.

Suppliers can only participate in a single Checkout use case.

Credit Payment Service can participate with many Checkout use cases at the same time.

Checkout use cases may not need Credit Payment Service (for example, if payment is in cash)

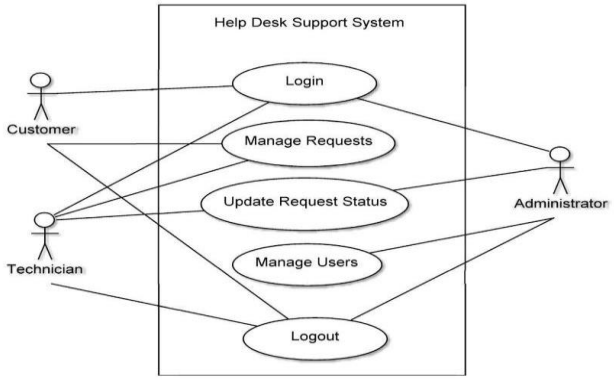


USE CASE DIAGRAM FOR CUSTOMER SUPPORT SERVICE OPERATIONS

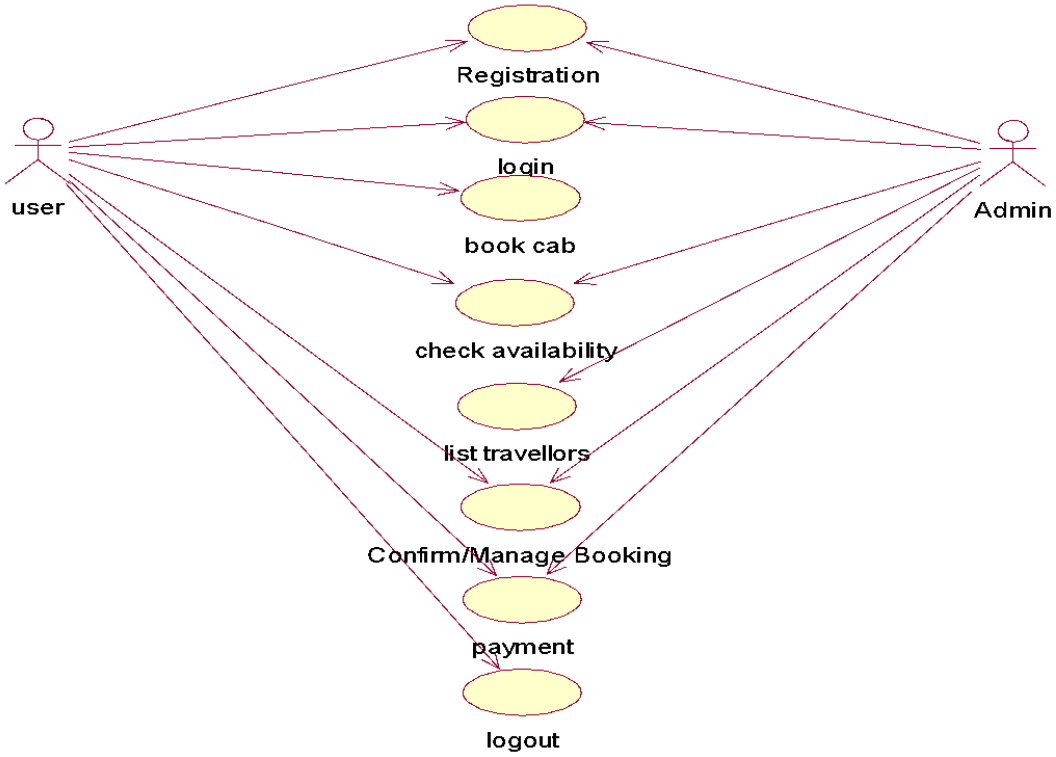
The Specific objectives for customer support system are the functions associated with the customer the system should include all the functions associated with customer, technician and administrator. Products for the customer from order entry to arrival of shipment.

Customer Service support service operations has use cases

- Customer
- Technician
- Administrator



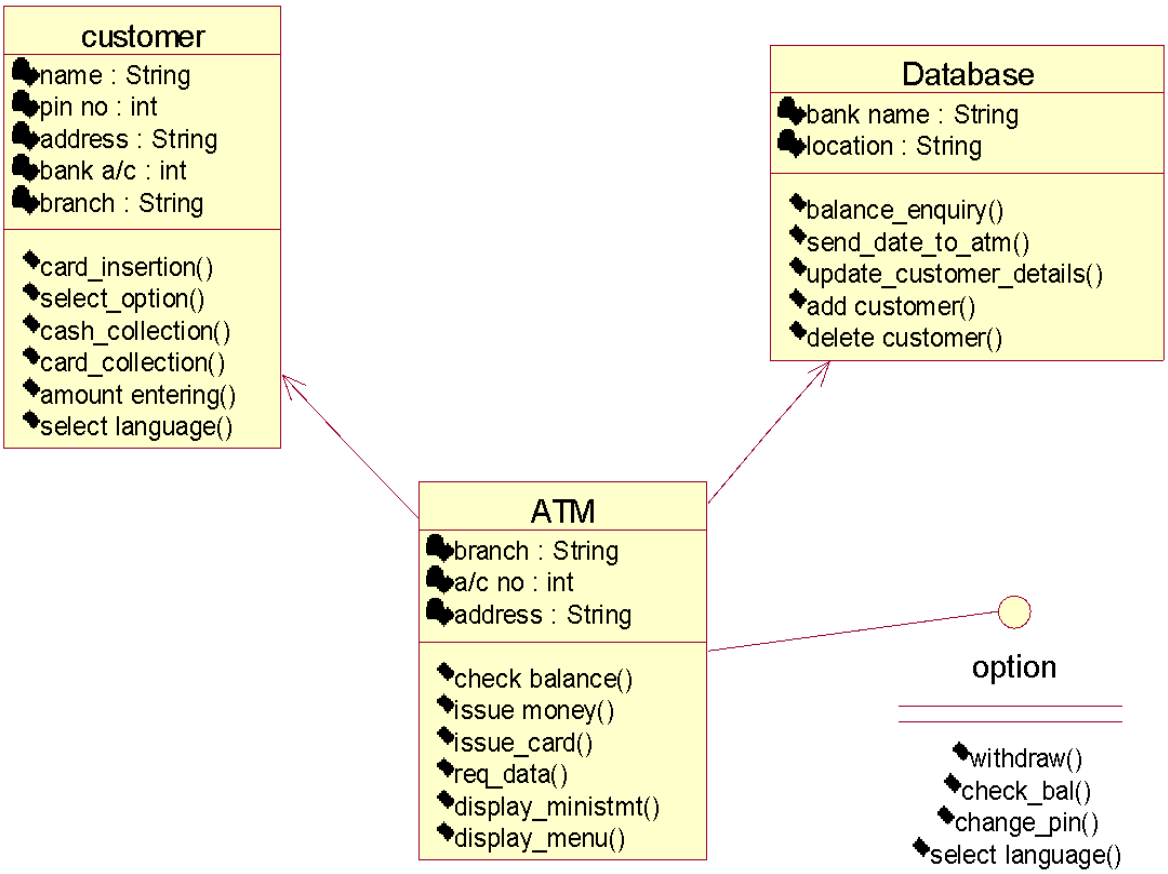
USE CASE DIAGRAM FOR CAB BOOKING SERVICE



Cycle-5 and Cycle-6

For each case study given earlier, Construct a Class Diagram in the following manner:

CLASS DIAGRAM FOR ATM



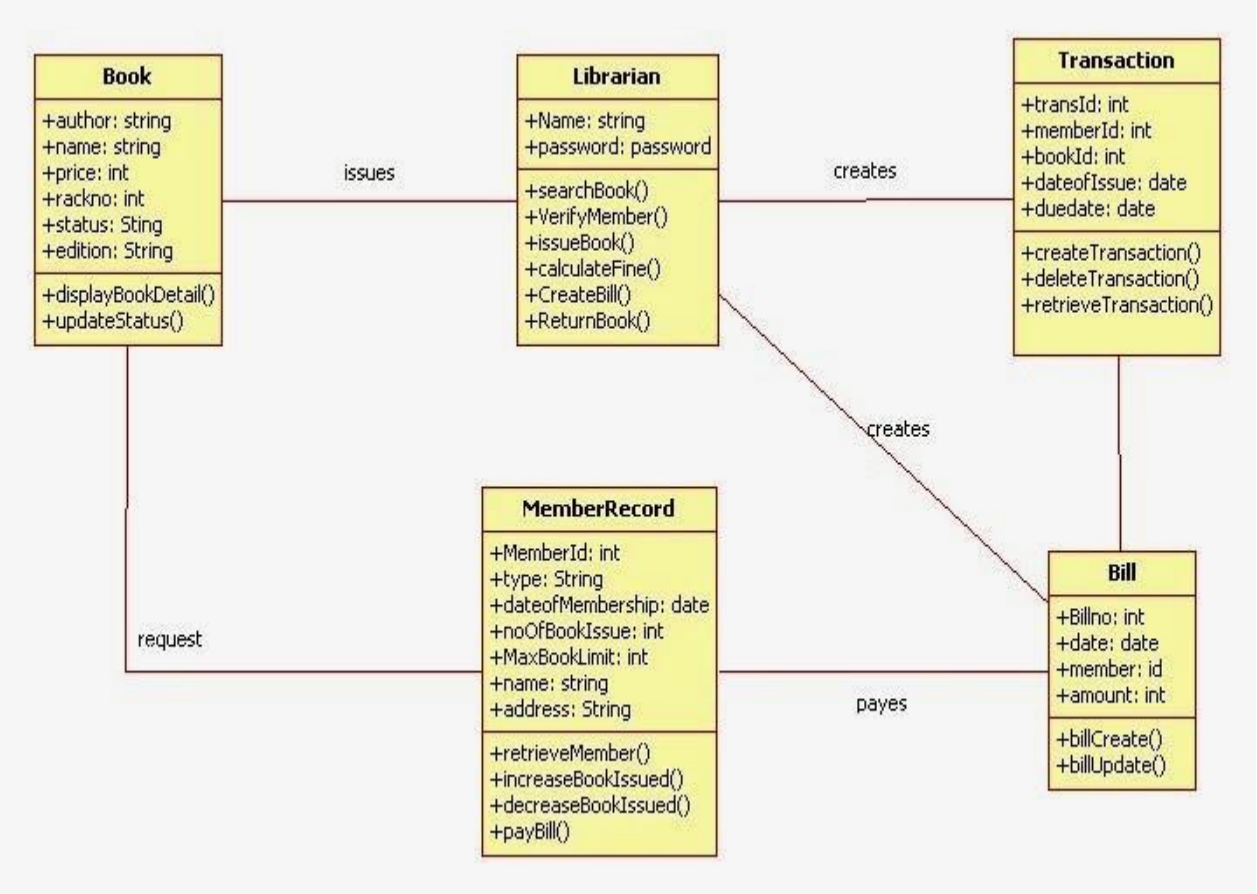
CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

The uml class diagram describes the structure of the system by showing the system classes, attributes, methods, and associations between classes.

UML Class diagram for library management system contains classes such as
1.Book class
2.librarian class
3. Transaction class
4.Member record class
5.Bill class

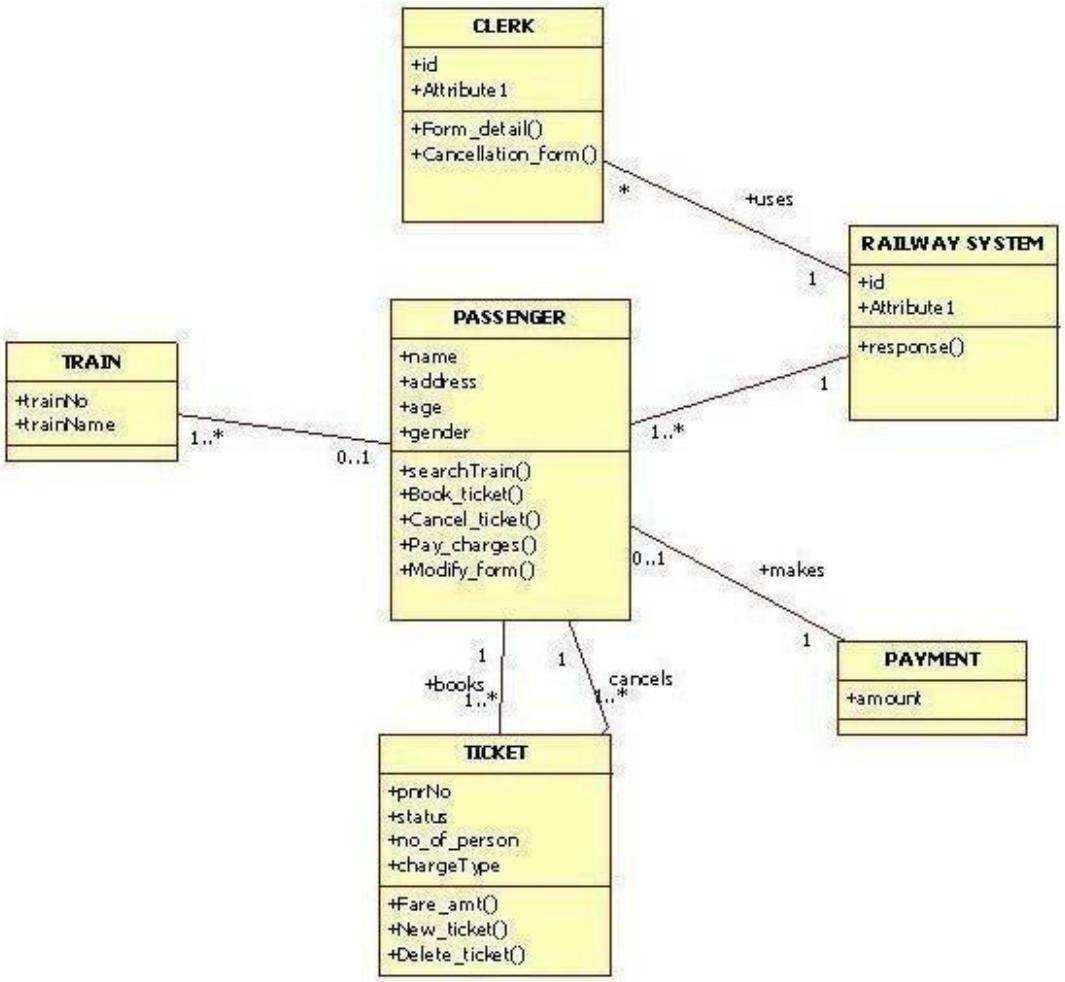
Each class contains various attributes and methods (Functions) which call other class attributes to share data.

1. Book Class contains attributes such as author, book name, price status, rack no, edition and functions such as display book details, update status.
2. Librarian class contains attributes such as name, password and functions such as search book, issue book, calculate fine, calculate bill.
3.transaction class contains attributes such as transaction id, member id bill id date of issue etc. and functions such as create transaction delete transaction
4.Member record class contains attributes such as member id, type, date of issue, number of books, etc. and functions such as increase book issue, decrease book issue, pay bill.
5. Bill class contains attributes such as bill no, date, member, amount and functions such as create bill, update bill.

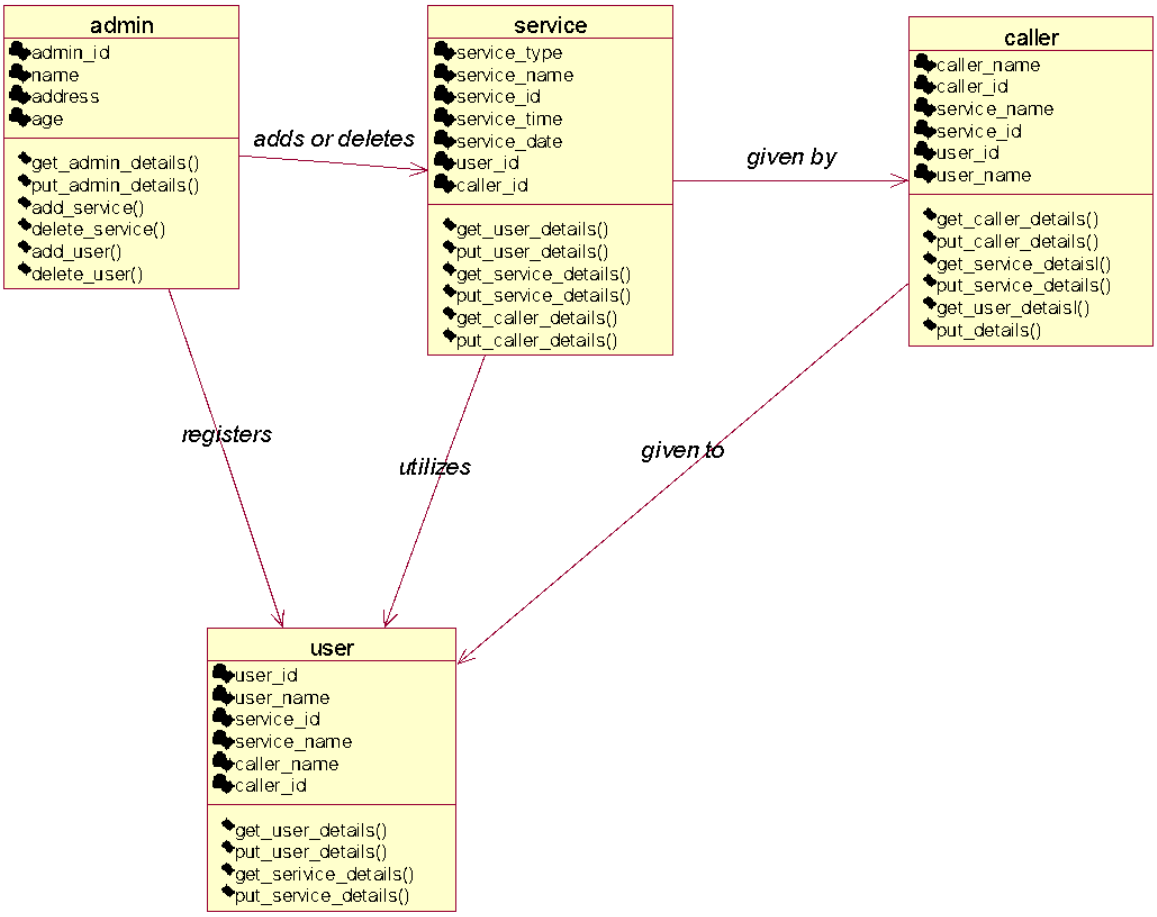


uml class diagram for library management system

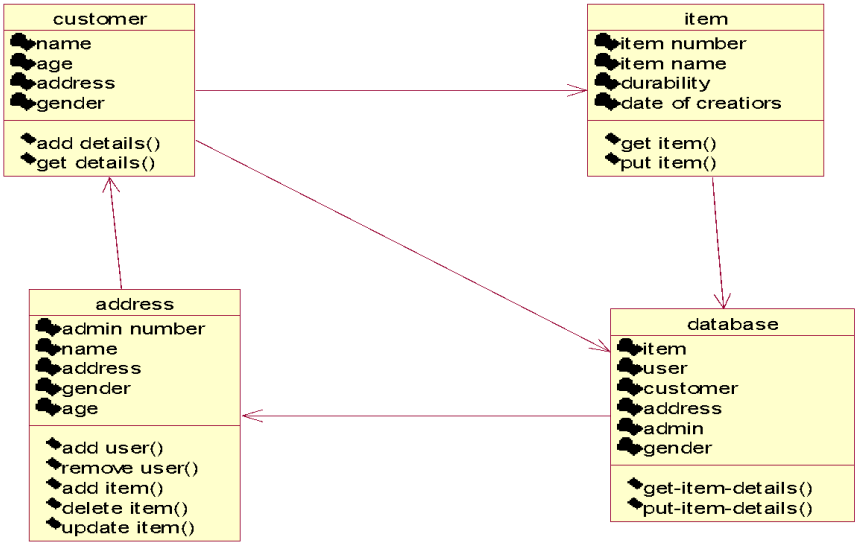
RAILWAY RESERVATION SYSTEM



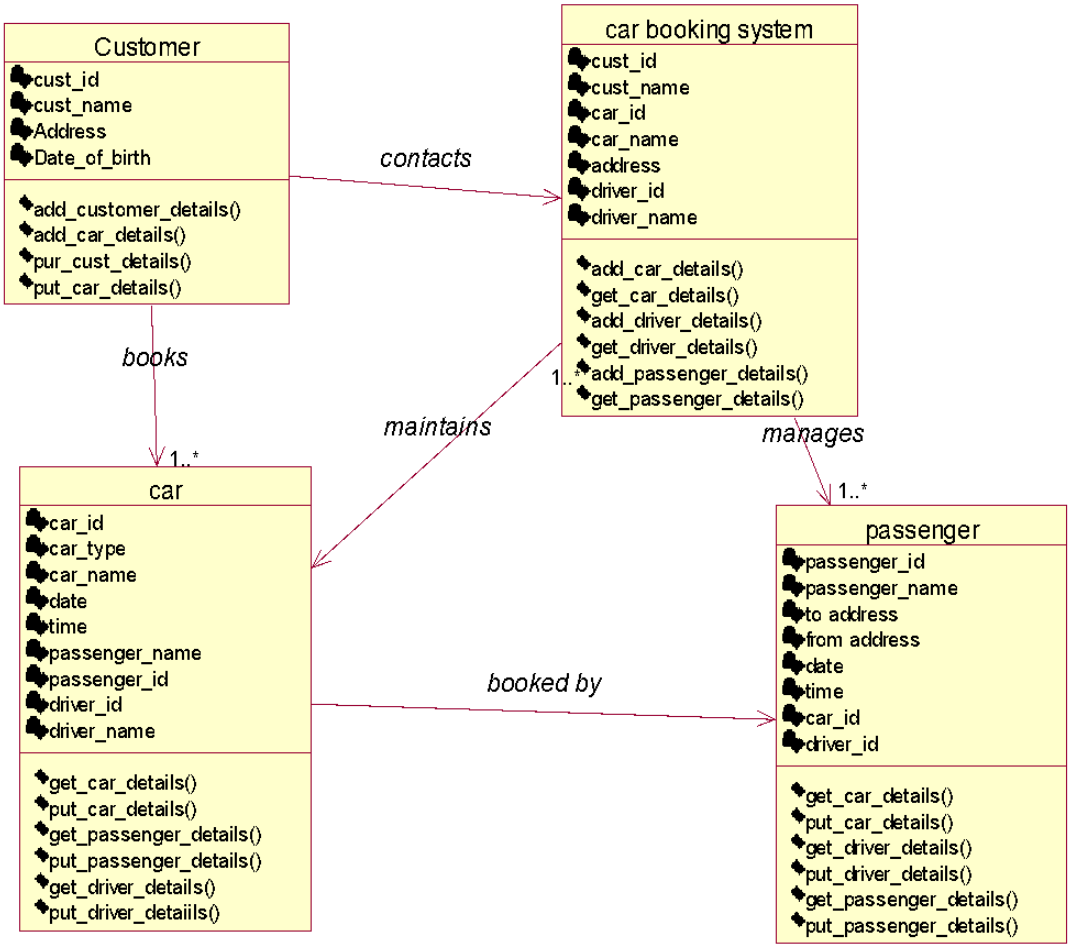
CUSTOMER SUPPORT SERVICE OPERATIONS



POINT OF SALE



CLASS DIAGRAM FOR CAB BOOKING SERVICE

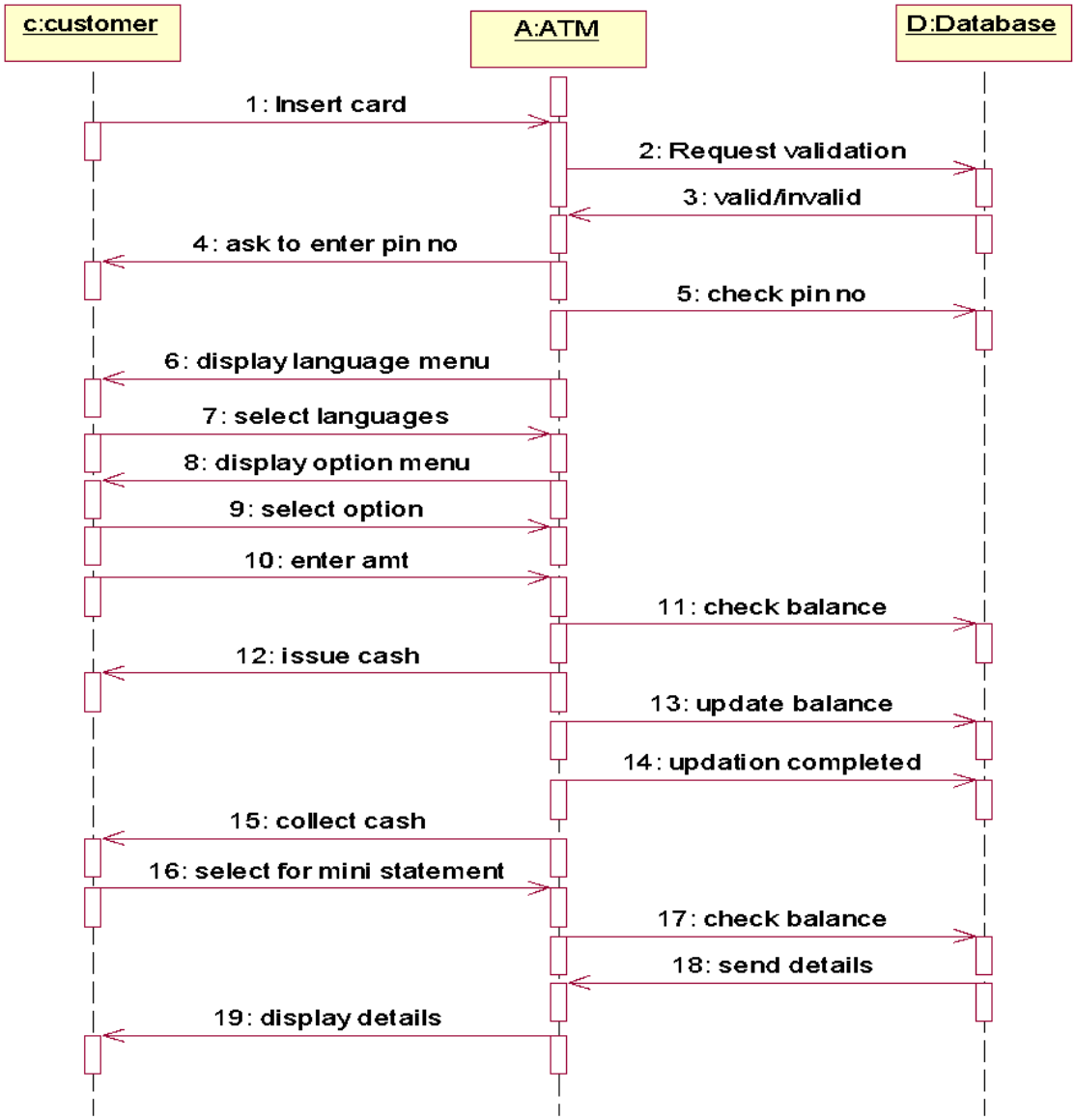


Cycle-7

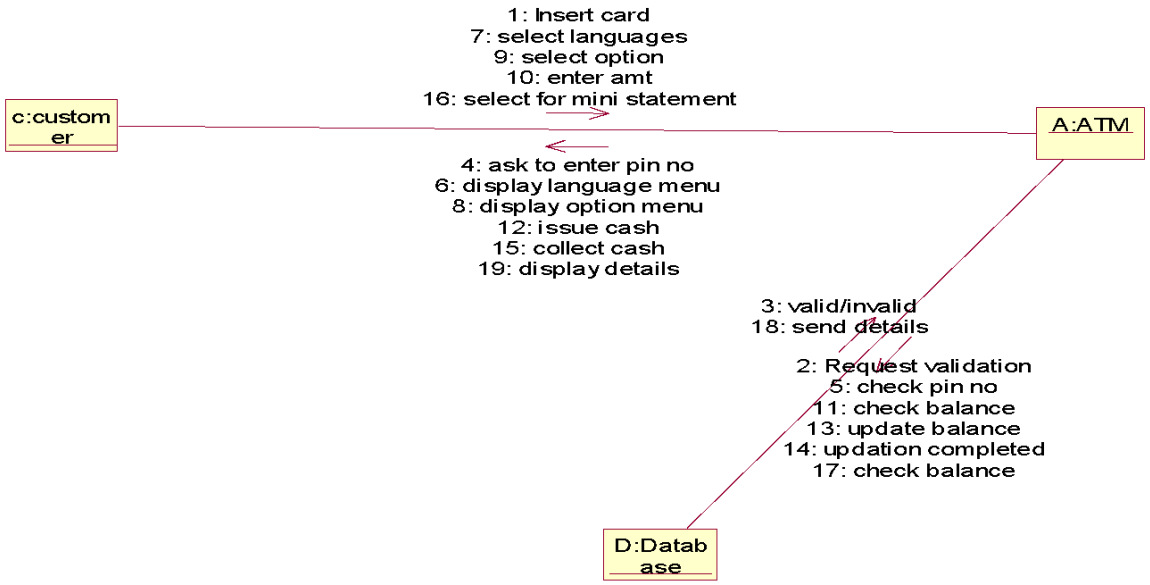
For each case study given earlier, Construct Interaction Diagrams in the following manner:

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

SEQUENCE DIAGRAM FOR ATM



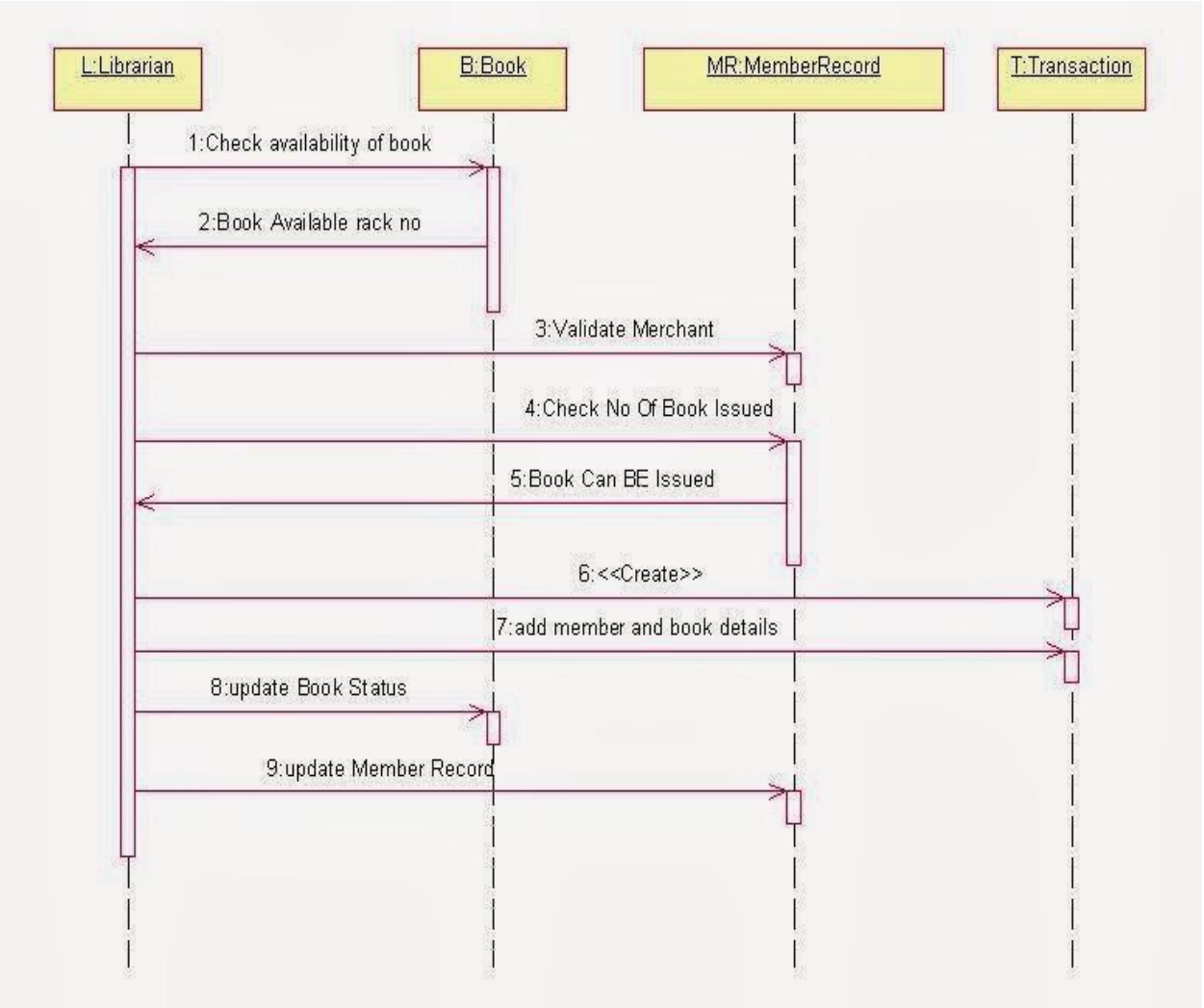
COLLABORATION DIAGRAM FOR ATM



SEQUENCE DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

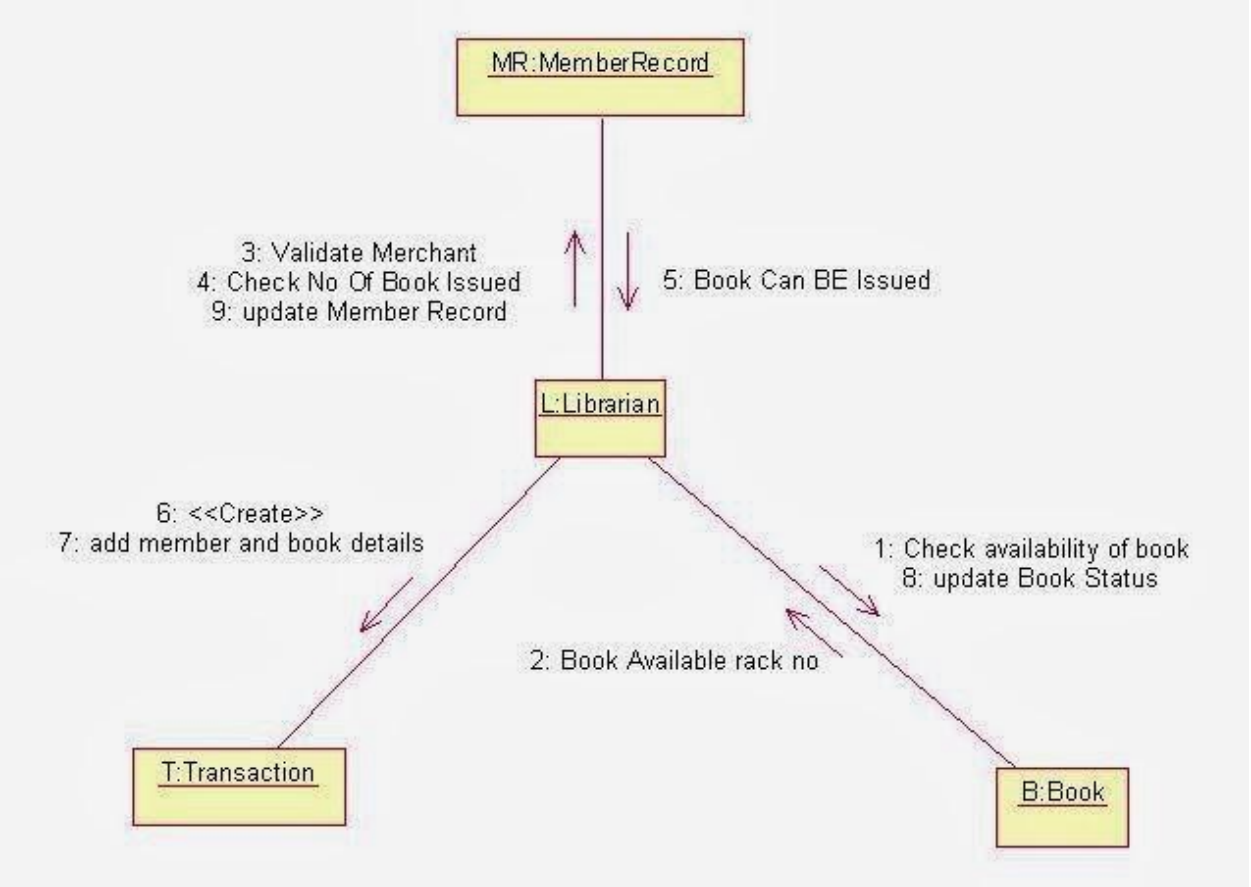
A Sequence diagram describes interaction among classes in terms of an exchange of messages over time. Sequence diagrams demonstrate the behavior of objects in a use case by describing the object and messages they pass. A sequence diagram depicts the sequence of actions that occurs in a system. The invocation of methods in each object and the order in which they are captured in a sequence diagram. This makes sequence diagrams very useful. Sequence diagram in two dimensional in nature

Librarian Issues a book to member: Whenever a member asks for a book, the librarian checks the availability of the book, if the book is available then the rack number of that book will be returned to librarian. Librarian then checks the validity of the member by verifying the library card. If the member is valid then the number of books issued to him is less than maximum allowed, books issued to him and transaction is created. Librarian then updates the number of books issued to member and status of book

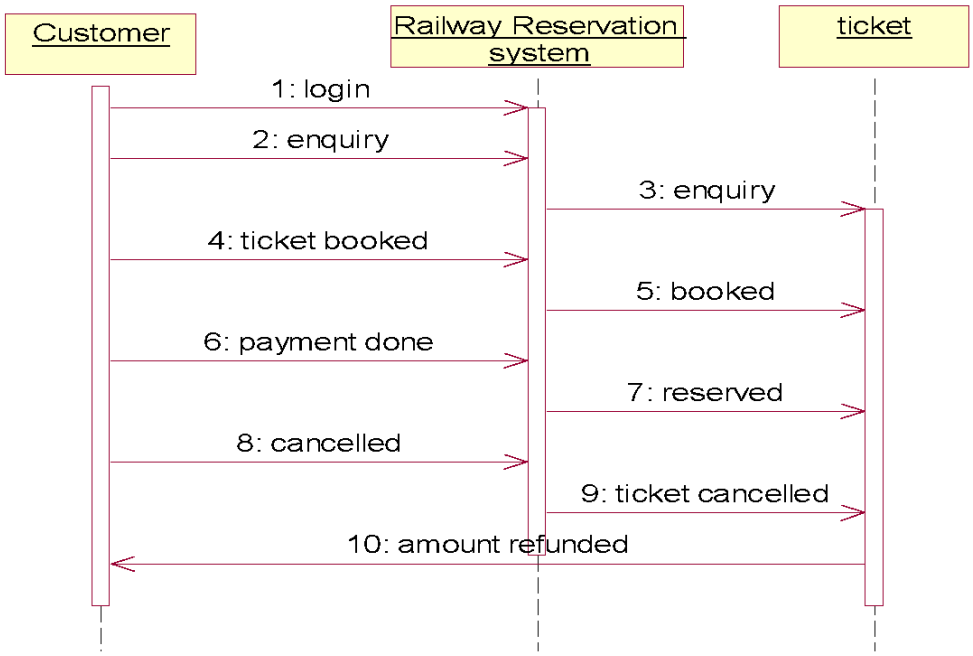


COLLABORATION DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

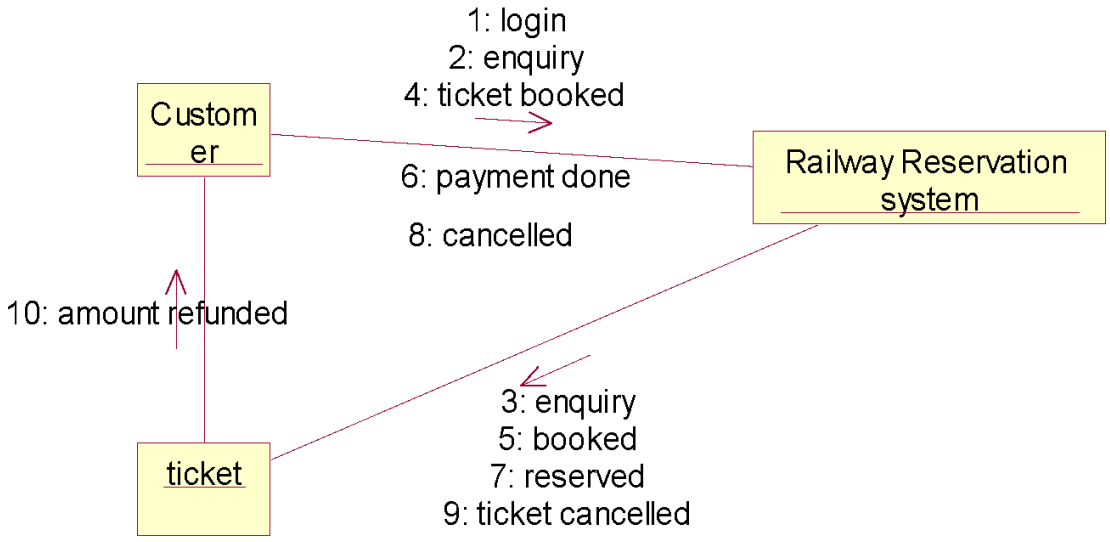
A Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. A collaboration diagram is very similar to a sequence diagram. Collaboration diagram shows the objects and their association with other objects. Modeling objects in a system and representing the association between the objects as links are easily represented in a collaboration diagram. if you are drawing diagrams in rational rose software after drawing a sequence diagram for an inventory management system press F5 to see the collaboration diagram. Sequence diagrams and collaboration diagrams show the same information but sequence diagrams focus on the temporal aspect and collaboration diagram focus on communication between the objects of the system



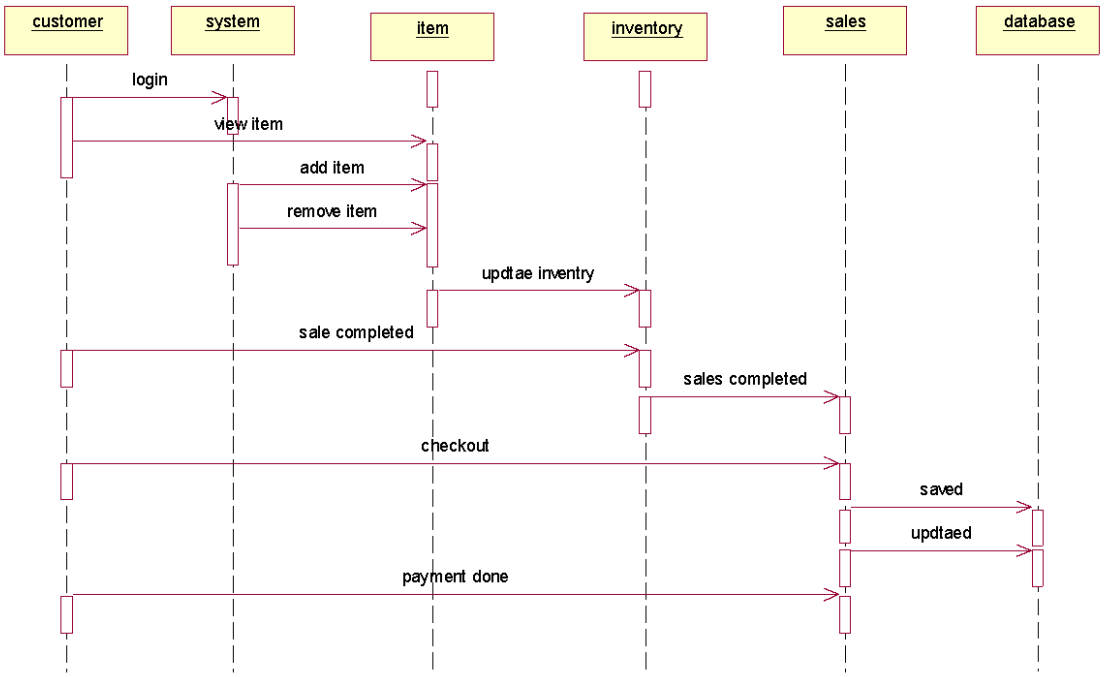
SEQUENCE DIAGRAM FOR RAILWAY TICKET RESERVATION SYSTEM (RRS)



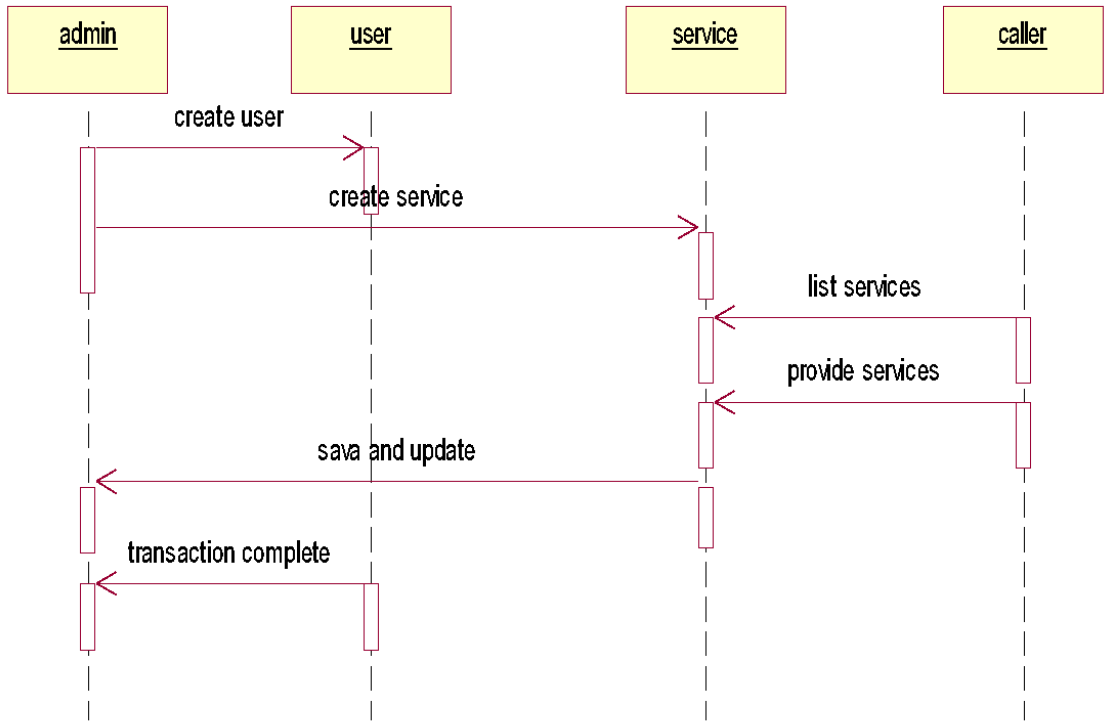
COLLABORATION DIAGRAM FOR RAILWAY TICKET RESERVATION SYSTEM



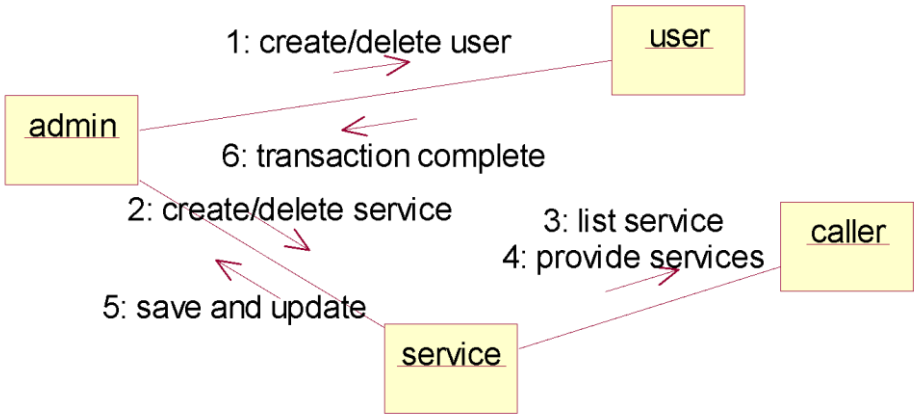
SEQUENCE DIAGRAM FOR POINT OF SALE



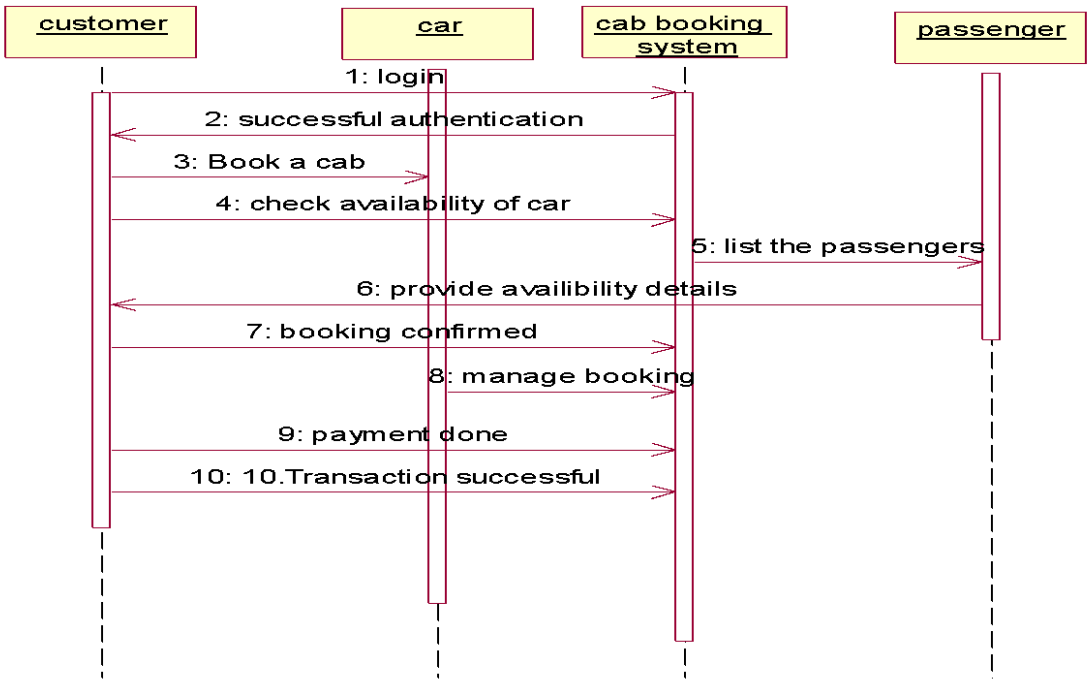
SEQUENCE DIAGRAM FOR CUSTOMER SUPPORT SERVICE OPERATIONS



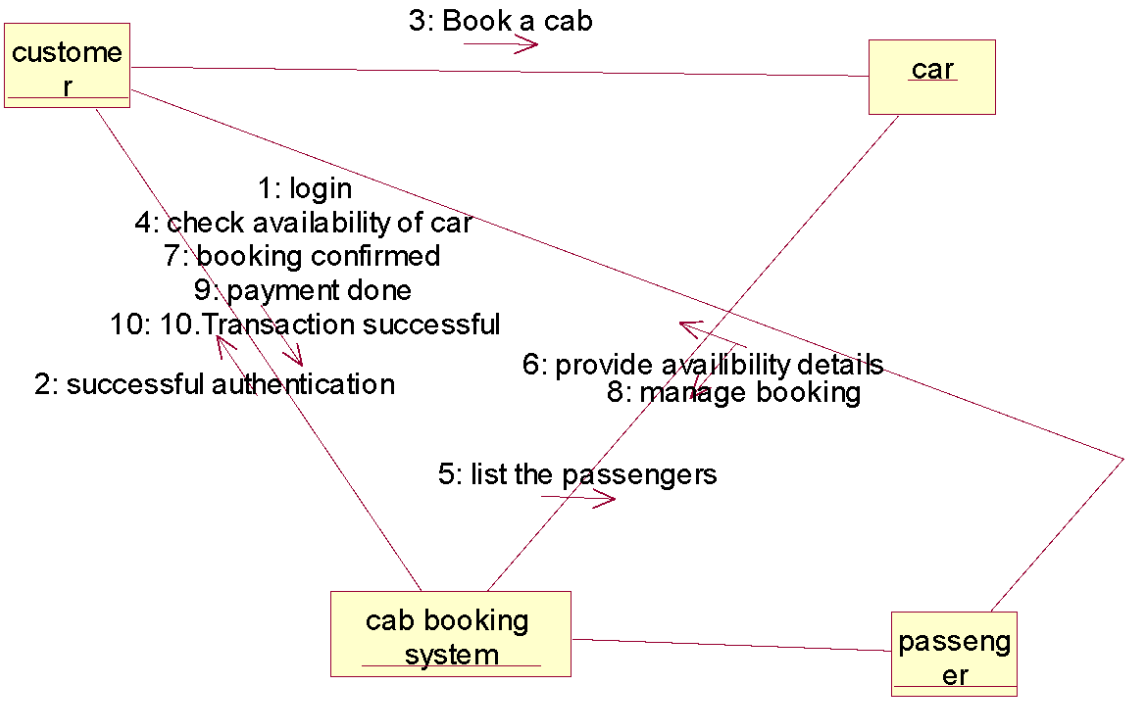
COLLABORATION DIAGRAM FOR CUSTOMER SUPPORT SERVICE OPERATIONS



SEQUENCE DIAGRAM FOR CAB BOOKING SERVICE



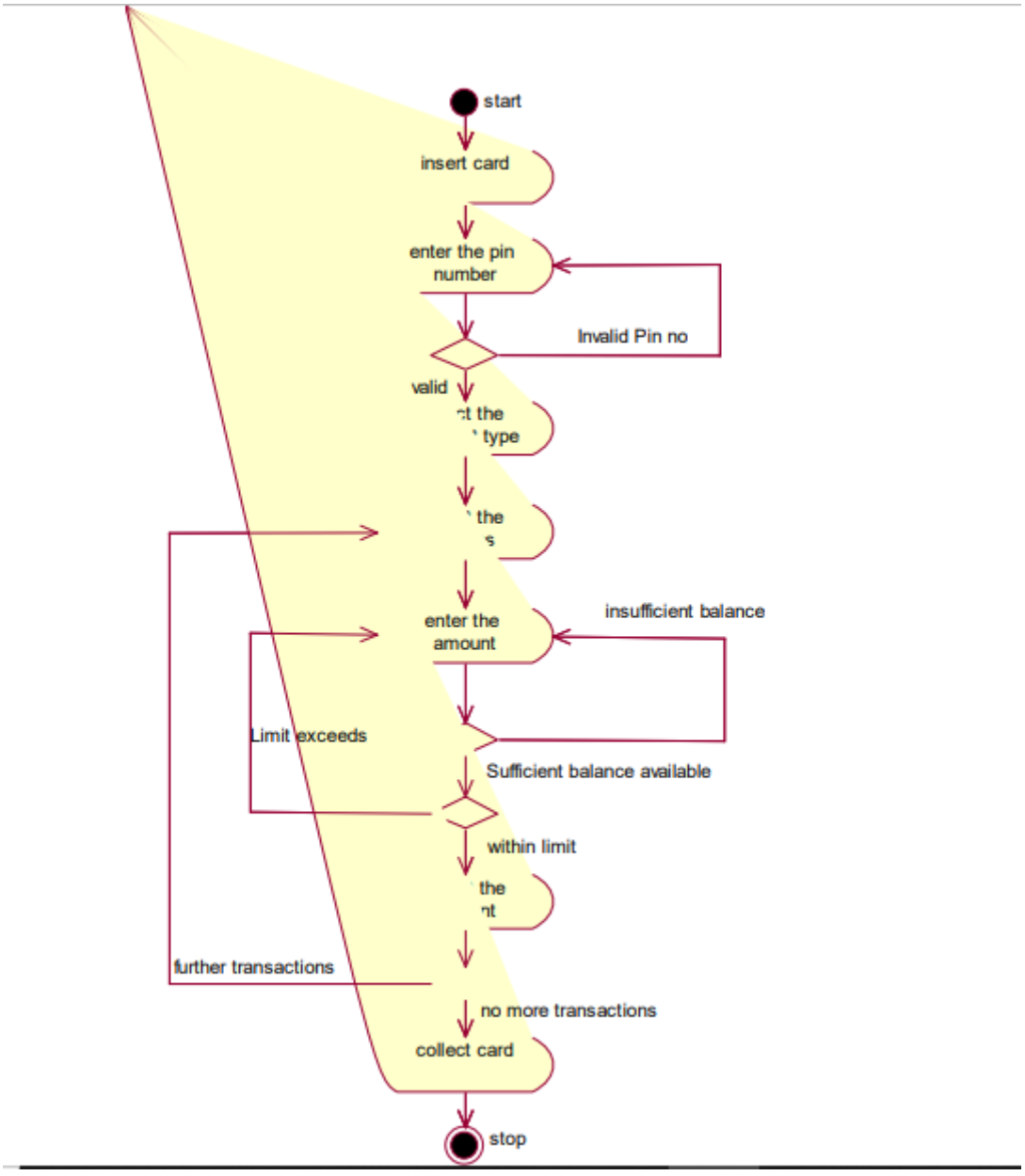
COLLABORATION DIAGRAM FOR CAB BOOKING SERVICE



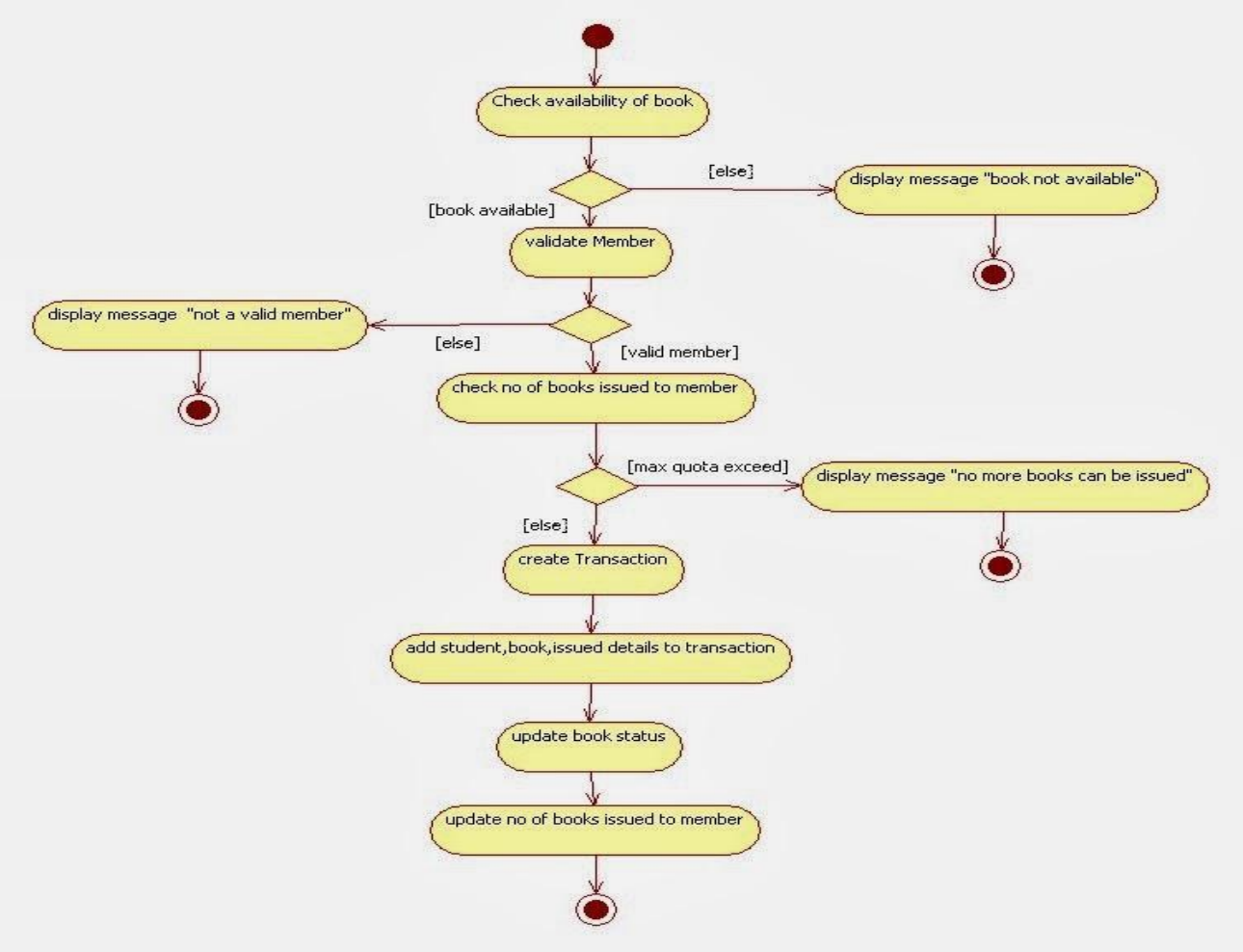
Cycle-8

For each case study given earlier, Construct Activity Diagram in the following

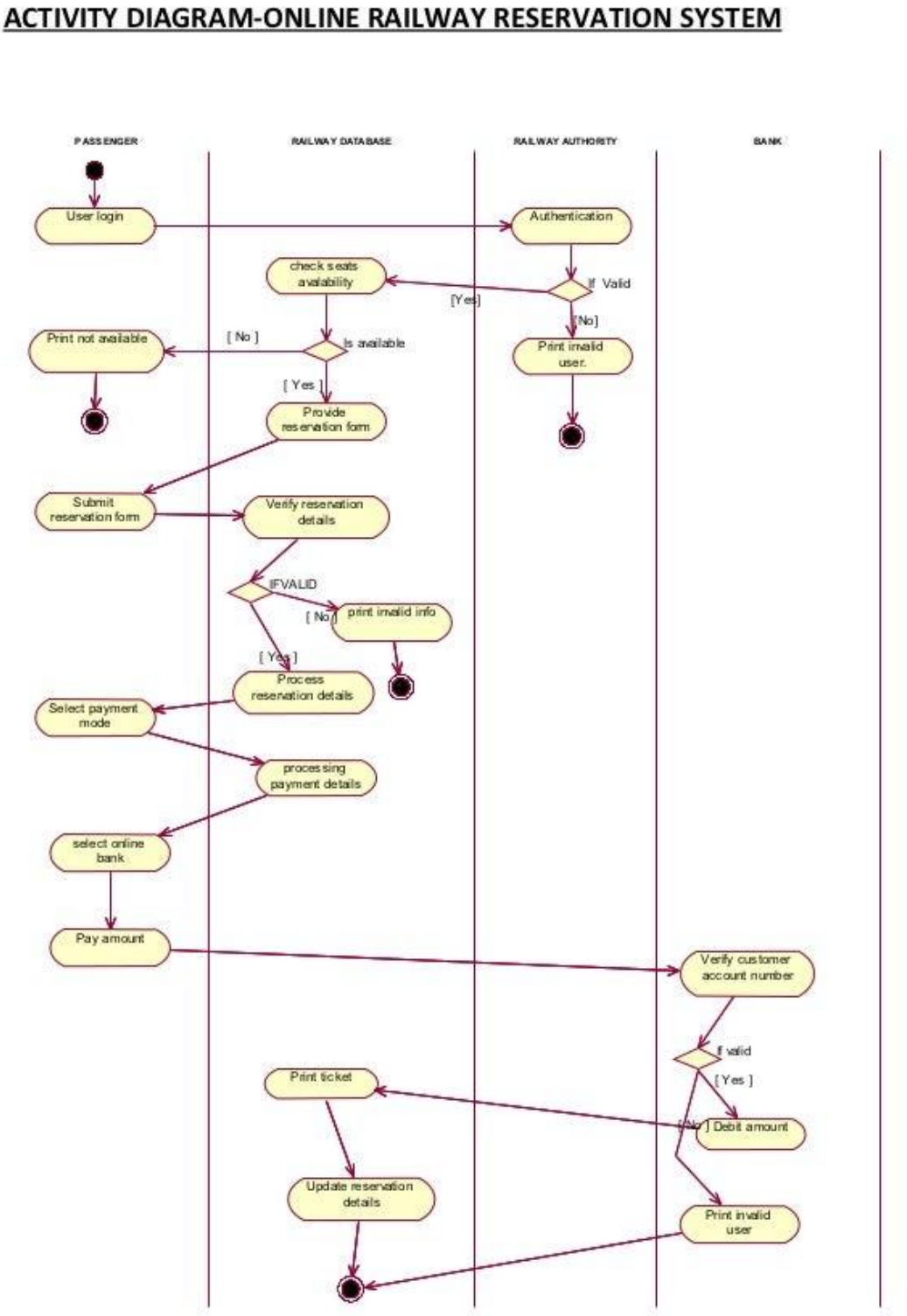
ACTIVITY DIAGRAM FOR ATM



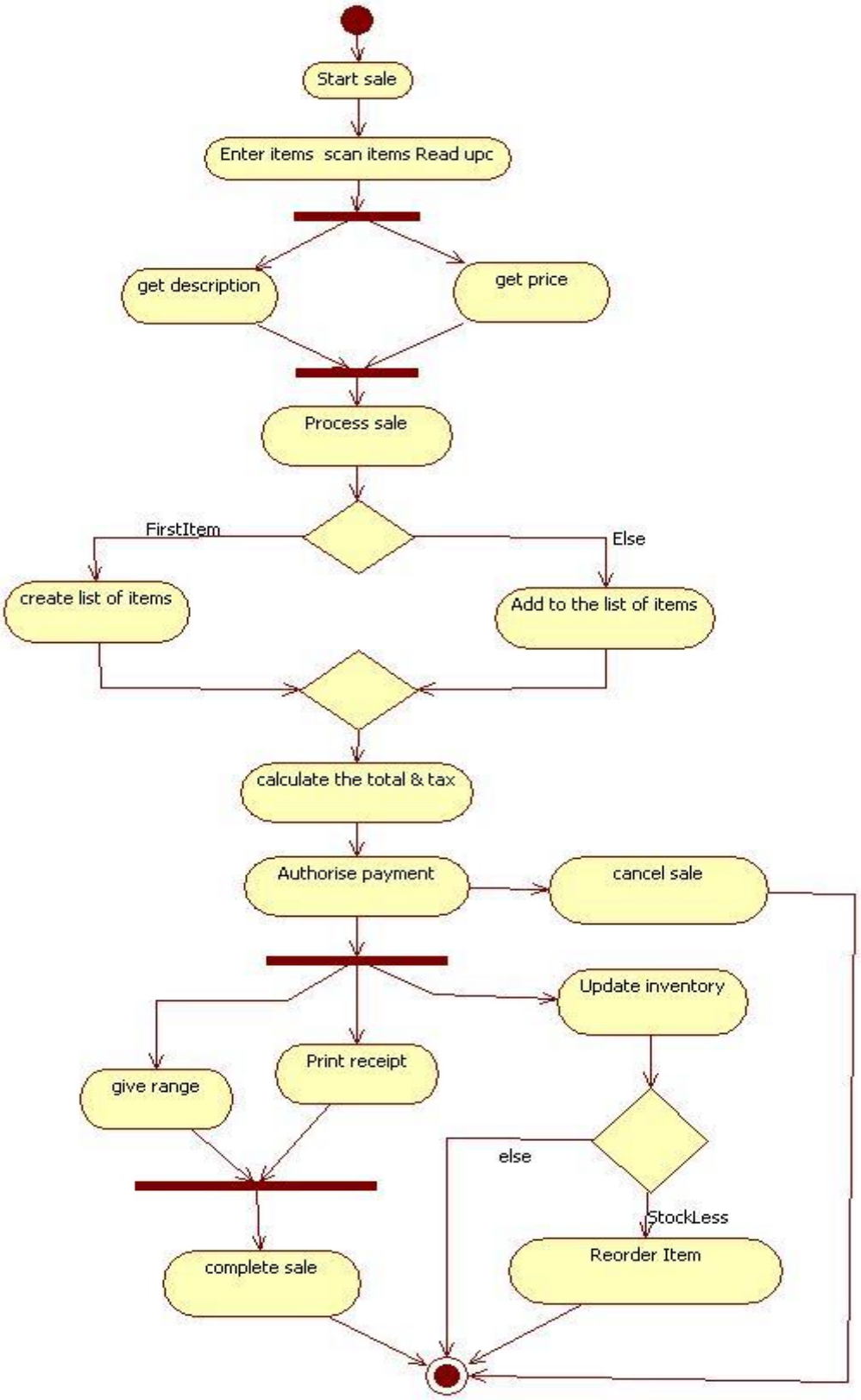
ACTIVITY DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM



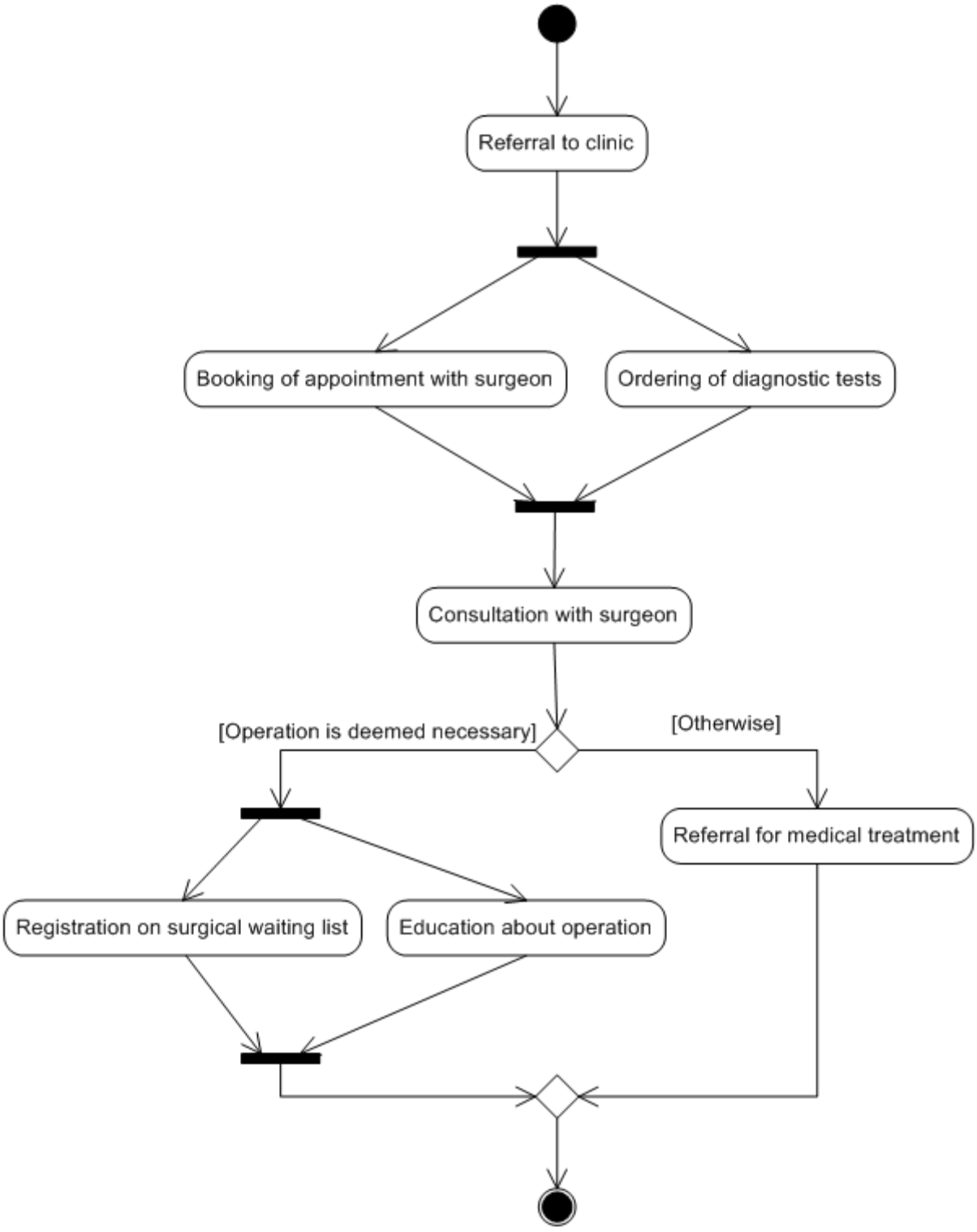
ACTIVITY DIAGRAM FOR RAILWAY TICKET RESERVATION SYSTEM



ACTIVITY DIAGRAM FOR POINT OF SALES

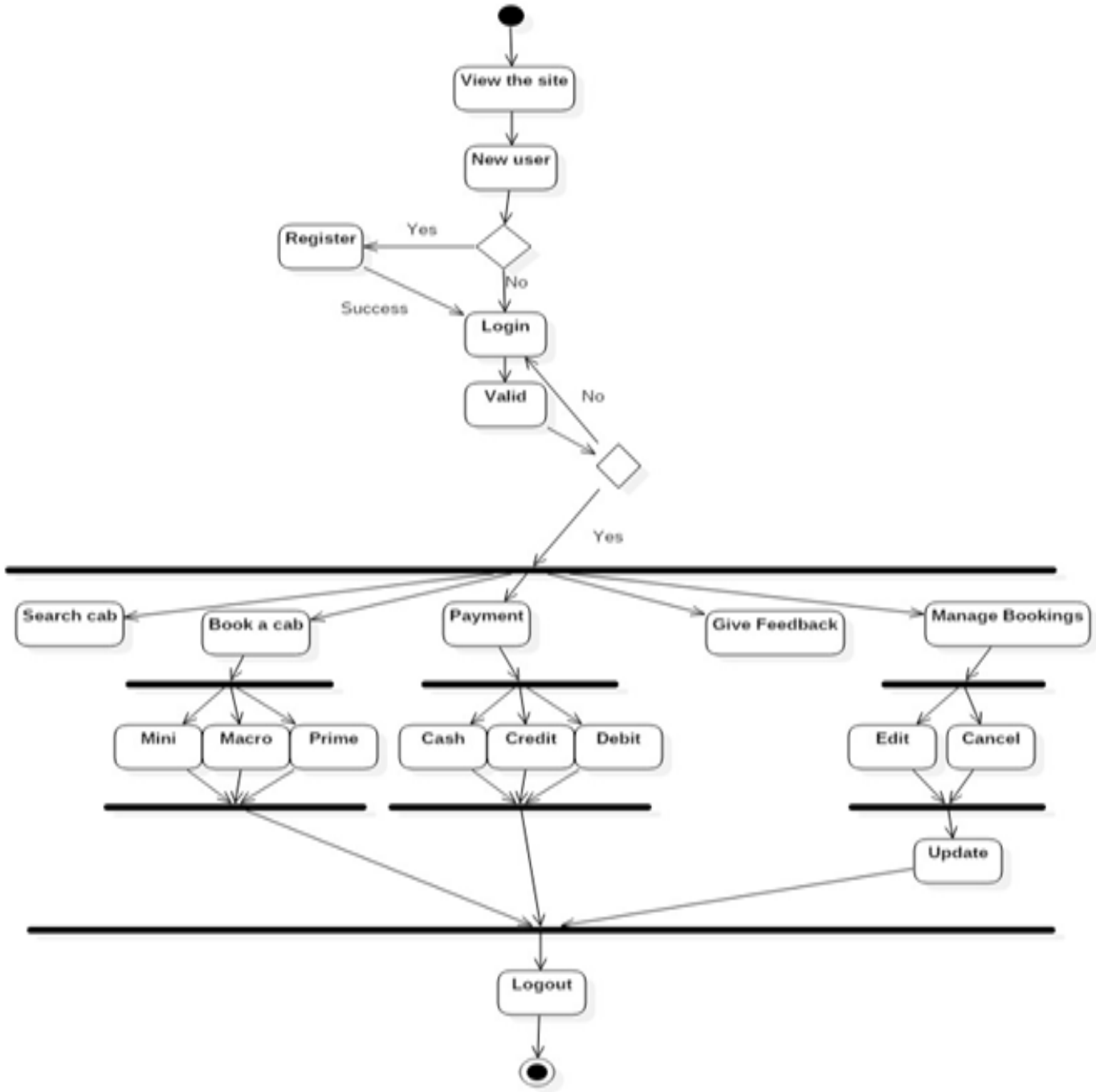


ACTIVITY DIAGRAM FOR CUSTOMER SUPPORT SERVICE OPERATIONS



ACTIVITY DIAGRAM FOR CAB BOOKING SYSTEM

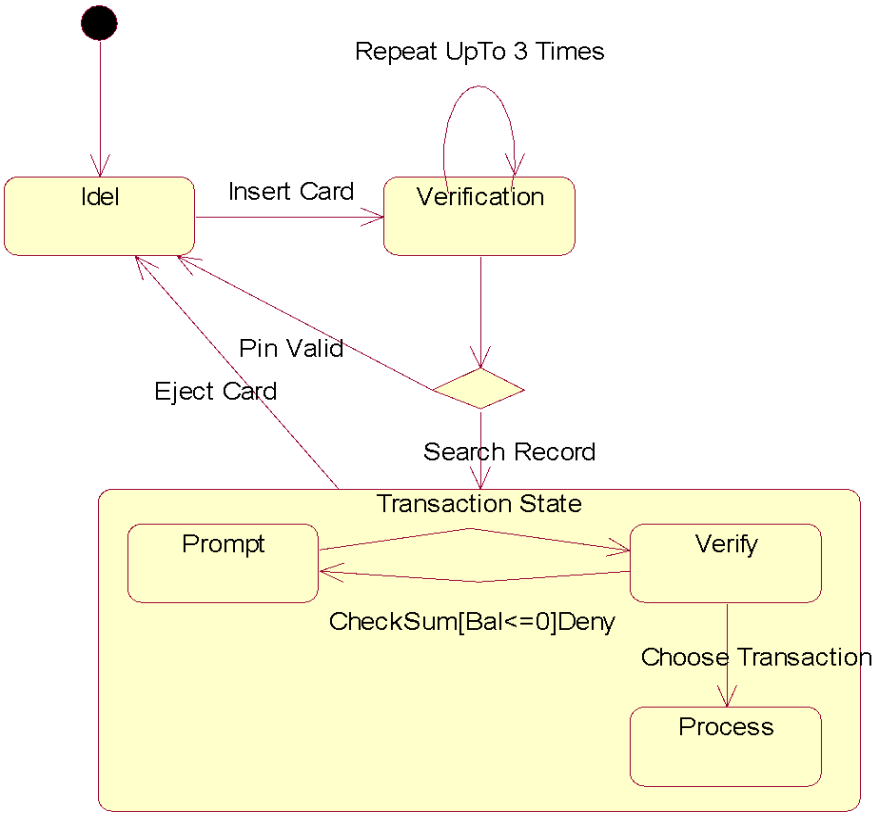
ACTIVITY DIAGRAM:-



Cycle-9

For each case study given earlier, Construct State Chart Diagram

STATE CHART DIAGRAM FOR ATM

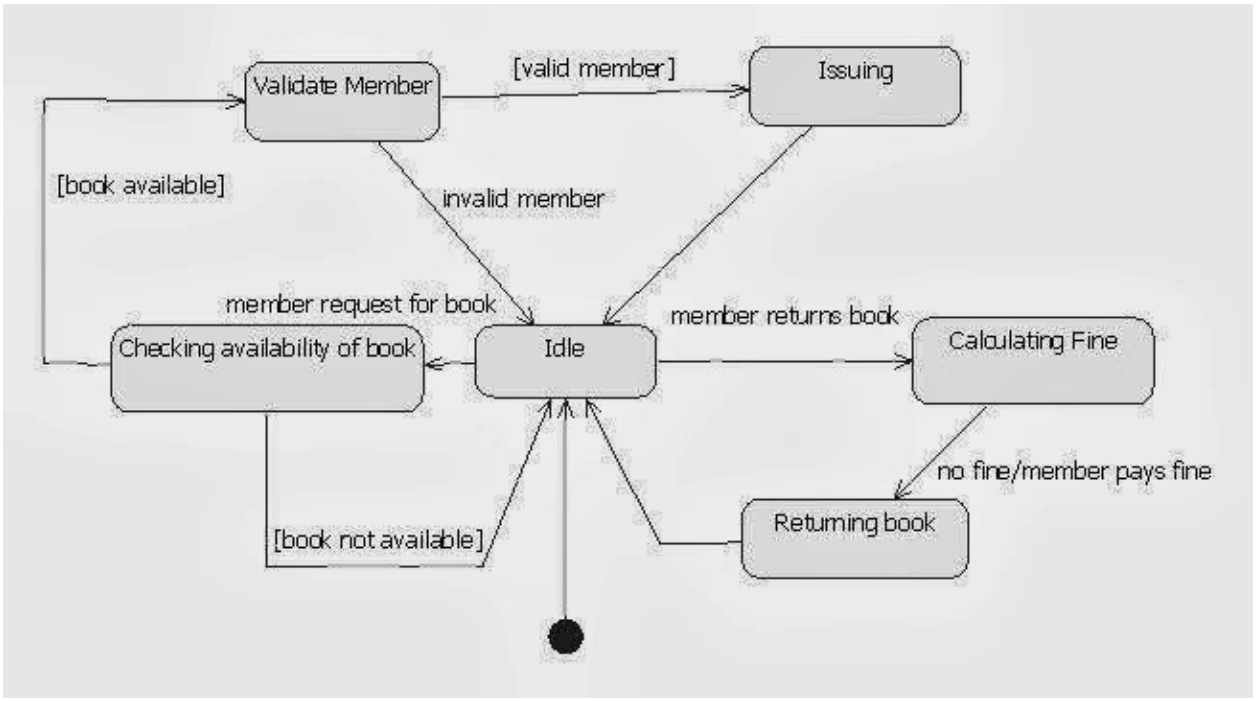


STATE CHART DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

A State chart diagram is used to help the developers better understand any complex functionalities or business flow of a specialized area of the system. In short, the state chart diagram depicts the dynamic behavior of the entire system. A state chart diagram shows a state machine. State chart diagrams can be used to graphically represent a finite state machine.

1 available: the book is available in the library and can be issued to members.
2 issued to member: Book is with member and is not available in library

State chart diagrams contain the following states.
states are librarians are :
1)Validate member
2)Issuing book
3)returning book
4)checking availability of book
5)calculating fine
6)idle

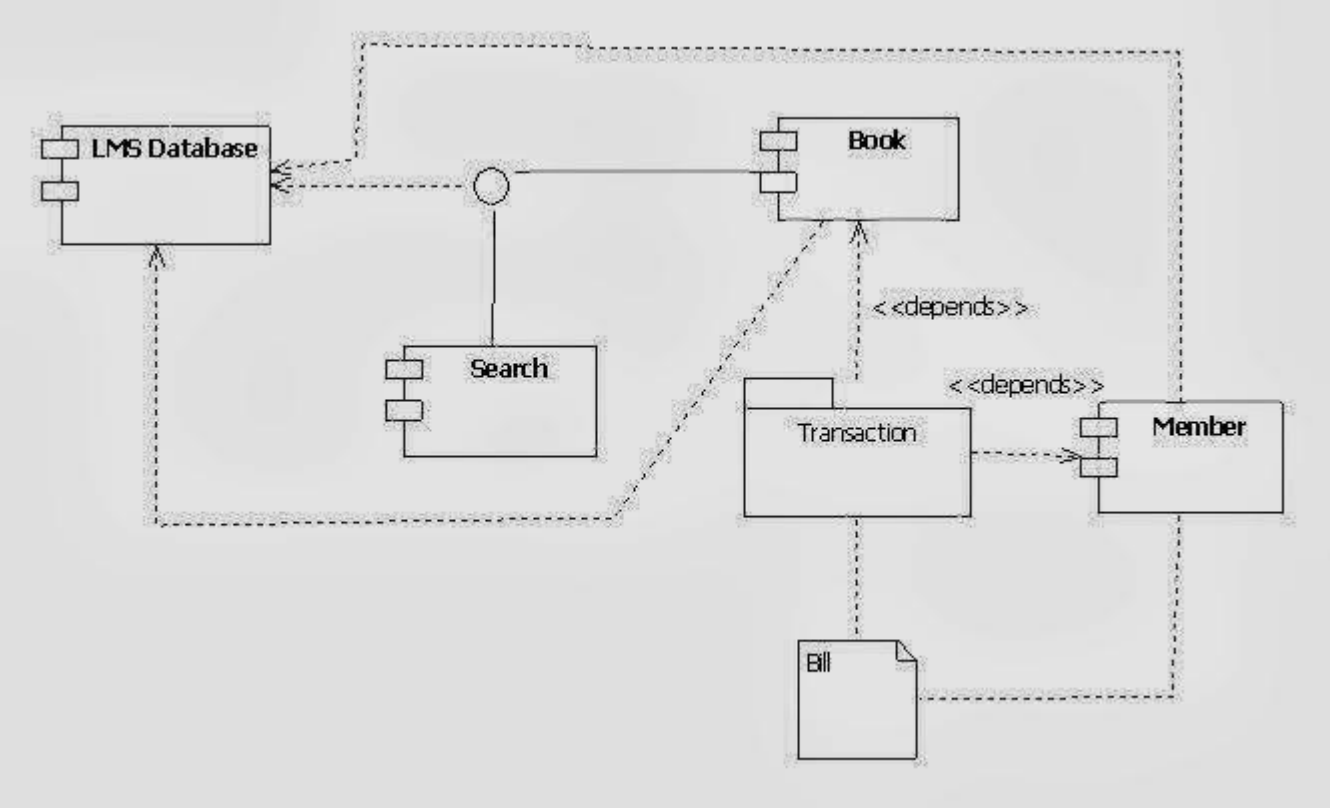

Validate Member
[valid member]
Issuing
[book available]
invalid member
member request for book
member returns book
Checking availability of book
Idle
Calculating Fine
no fine/member pays fine
Returning book
[book not available]

Cycle-10

For each case study given earlier, Construct Component Diagram

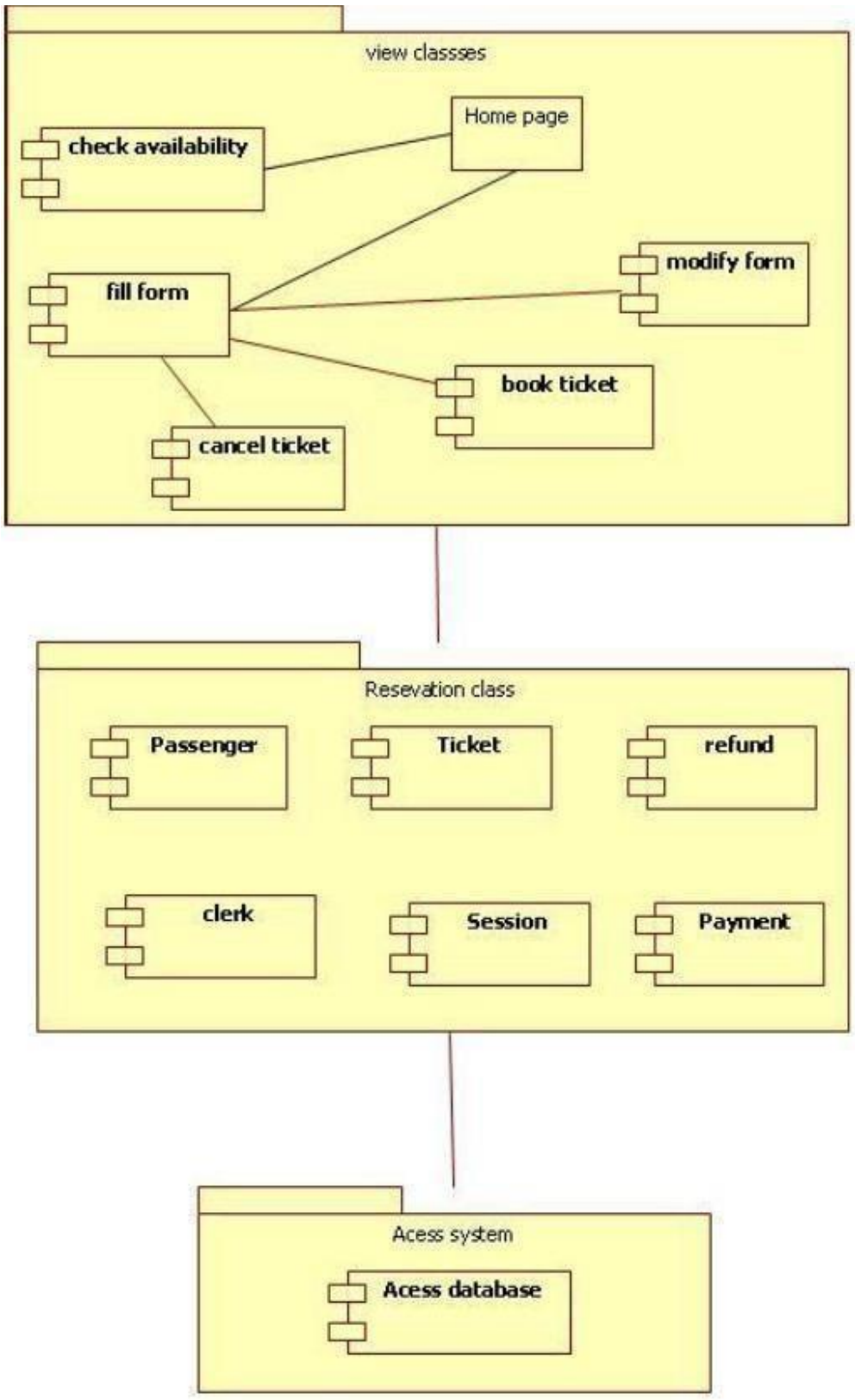
COMPONENT DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

Component diagrams are one of the two kinds of diagrams found in modeling the physical aspects of an object oriented system. A component diagram shows the organization and inter relationship between a set of components. We use component diagrams to visualize the static aspect of the physical component and their relationship and to specify their details for construction. Component diagram is essentially a class diagram that focuses on a system's components.

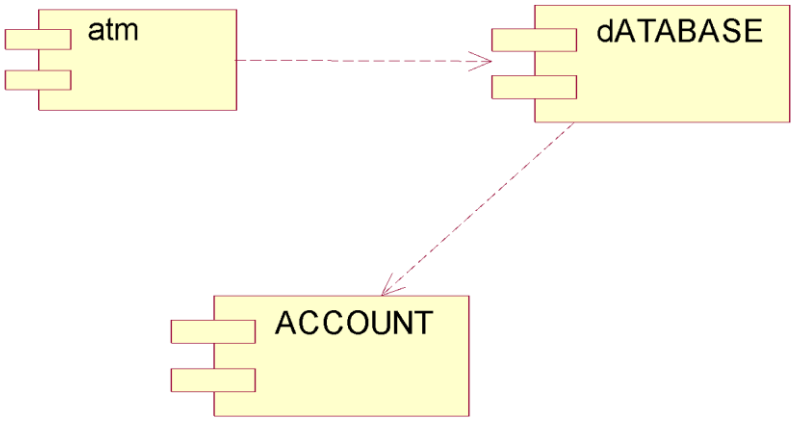


uml component diagram for library management System

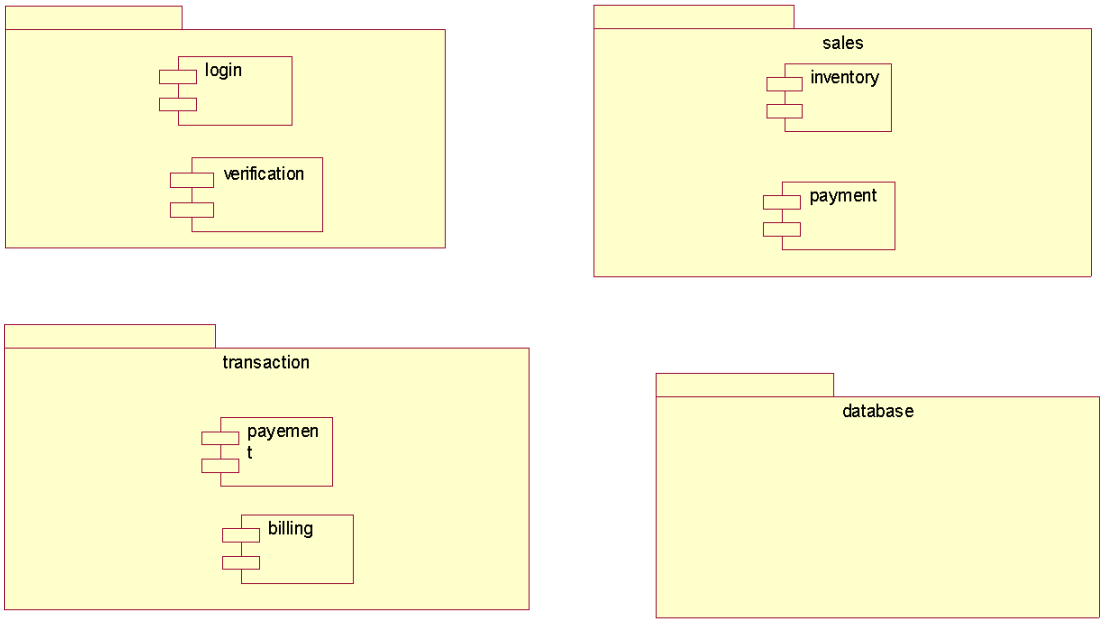
COMPONENT DIAGRAM FOR RAILWAY RESERVATION SYSTEM



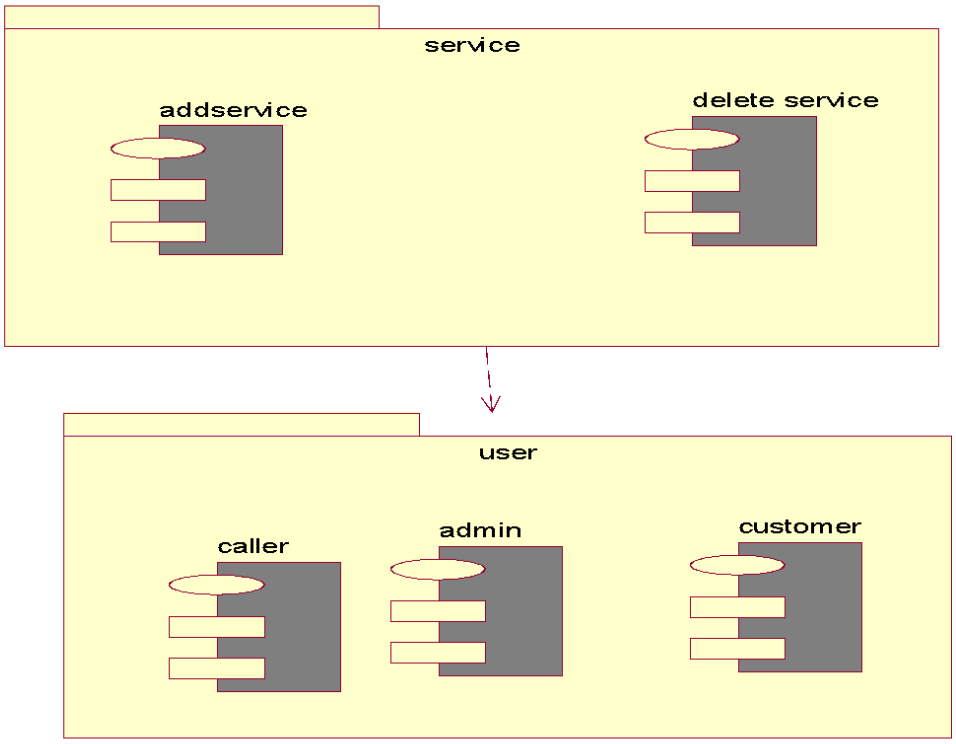
ATM COMPONENT DIAGRAM



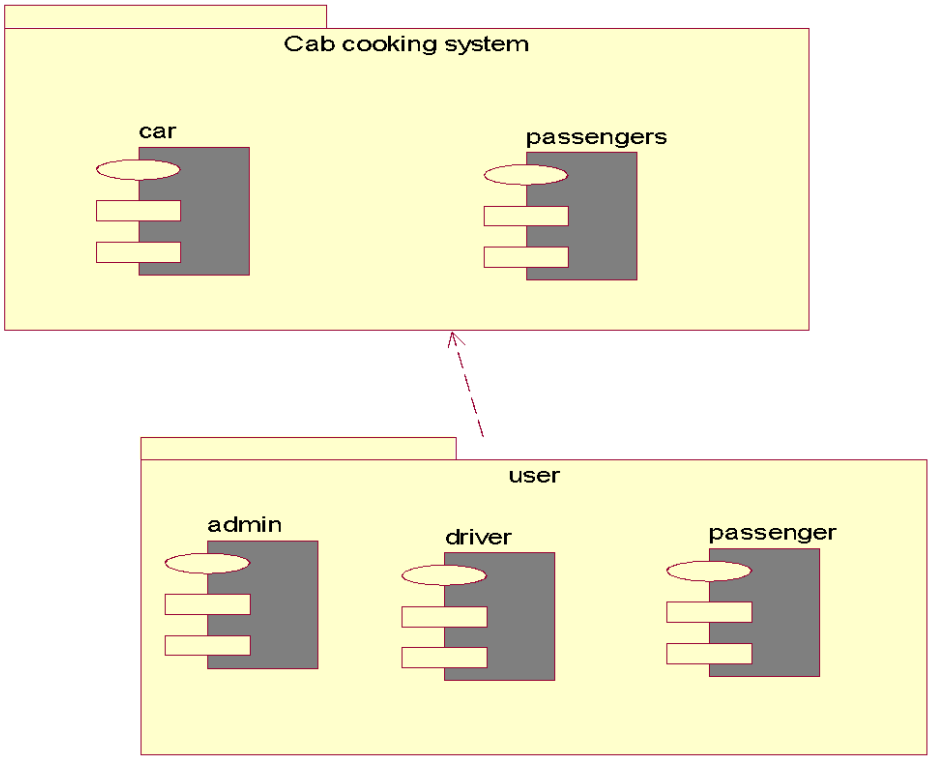
POINT OF SALE



CUSTOMER SERVICE SUPPORT SYSTEM



CAB BOOKING SYSTEM

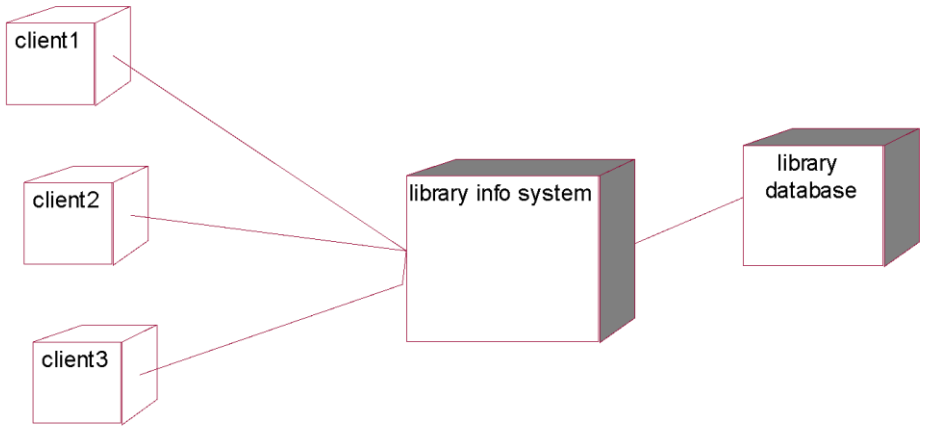


Cycle-11

For each case study given earlier, Construct Deployment

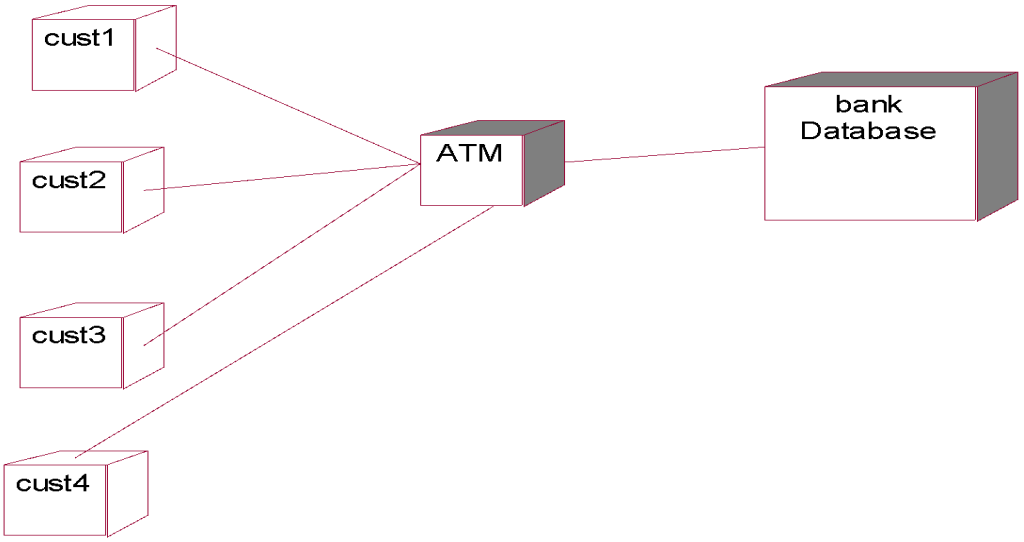
DEPLOYMENT DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

Deployment diagram is a diagram that shows the configuration of run time processing nodes and components that live on them. The deployment diagram provides a different perspective of the application. The deployment diagram captures the configuration of the run time element of the application. We use deployment diagrams to model the static and dynamic view of a system. Deployment diagrams are not only important for visualizing, specifying, and documenting embedded, client server, and distributed systems, but also for mapping executable systems through forward and reverse engineering

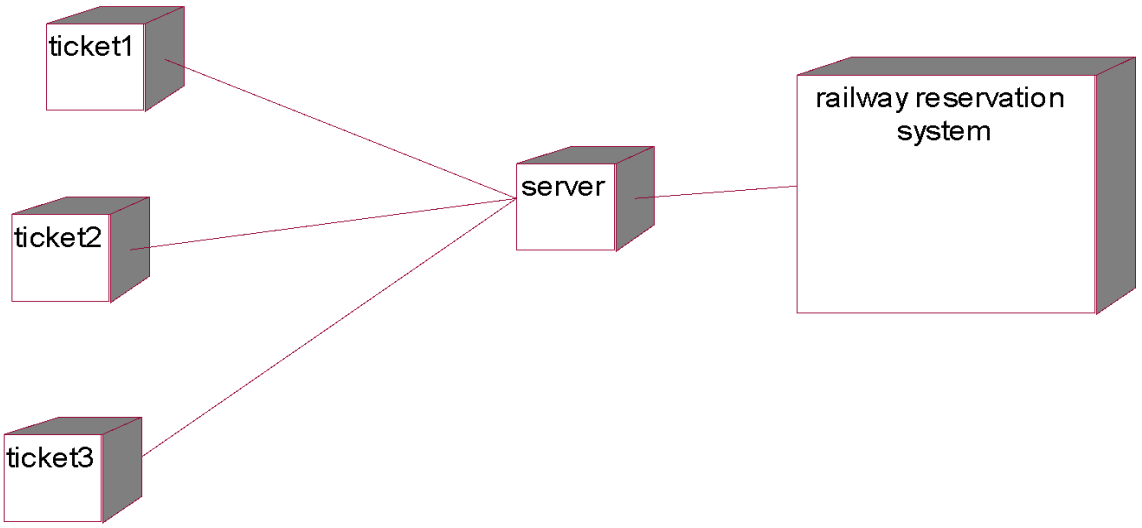


DEPLOYMENT DIAGRAM FOR ATM

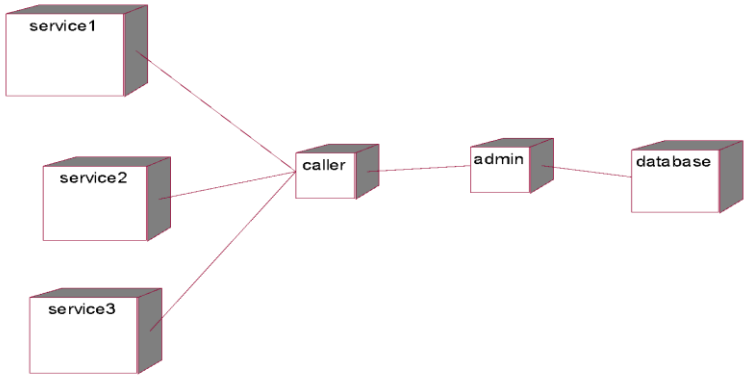
The above activity diagram starts when the customer inserts his card. He enters the pin no on the screen. The system then verifies whether the pin number is valid. In case it is not valid the customer is asked to re-enter the pin no. In case it is valid, the customer selects the account type he wants to use and the option (in this case withdrawing money).The customer then enters the amount he wants to withdraw. This is scrutinized first by checking if sufficient balance is present in the account and second if it is within the limit. In case both satisfy, the cash is issued to the customer. The customer collects the cash. In case he wants to further select some options, he is provided with the option to do so. If not, he collects his card and the activity diagram stops



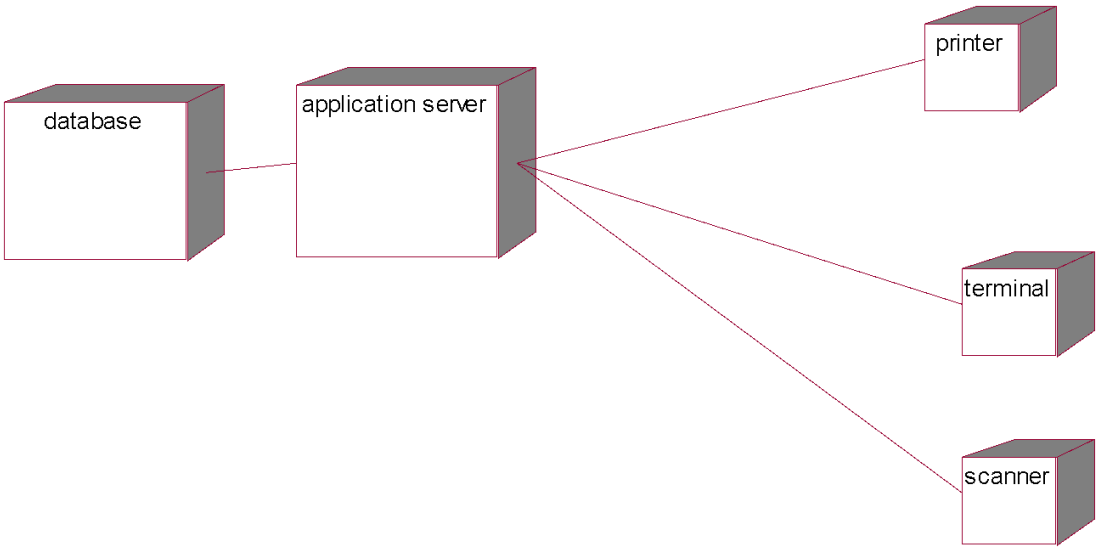
RAILWAY RESERVATION SYSTEM



CUSTOMER SUPPORT SERVICE SYSTEM



DEPLOYMENT DIAGRAM FOR POINT OF SALE



CAB BOOKING