# Malla Reddy Engineering College (Autonomous)

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad). Accredited 2nd time by NAAC with 'A' Grade, Maisammaguda (H), Medchal-Malkajgiri District, Secunderabad Telangana-500100 www.mrec.ac.in

## LABORATORY RECORD

LABORATORY NAME: SOFTWARE ENGINEERING AND UML LAB

LABORATORY CODE: 80514

YEAR/SEMESTER: III/II

REGULATIONS: MR18 (2018-19 ADMITTED BATCH)

## DEPARTMENT OF INFORMATION TECHNOLOGY

JULY-2021

# Malla Reddy Engineering College (Autonomous)

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to
JNTUH, Hyderabad). Accredited 2nd time by NAAC with 'A' Grade, Maisammaguda (H),
Medchal-Malkajgiri District, Secunderabad Telangana-500100 www.mrec.ac.in

## DEPARTMENT OF INFORMATION TECHNOLOGY

## CERTIFICATE

THIS IS TO CERTIFY THAT IT IS BONAFIED RECORD OF LABORATORY WORK DONE BY

MR./MS.________________________________________BEARING THE ROLL NUMBER

_____________________OF_____________________YEAR/SEMESTER__________________________

DEPARTMENT IN THE _____________________________________________LABORATORY DURING

THE ACADEMIC YEAR____________________UNDER OUR OBSERVATION.

LAB IN-CHARGE HEAD OF DEPARTMENT

INTERNAL EXAMINER EXTERNAL EXAMINER

# Malla Reddy Engineering College (Autonomous)

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to
JNTUH, Hyderabad). Accredited 2$^{nd}$ time by NAAC with 'A' Grade, Maisammaguda (H),
Medchal-Malkajgiri District, Secunderabad Telangana-500100 www.mrec.ac.in

## Department of Information Technology

## INDEX

## Introduction

CASE tools known as Computer-aided software engineering tools is a kind of component-based development which allows its users to rapidly develop information systems. The main goal of case technology is the automation of the entire information systems development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation. Component based manufacturing has several advantages over custom development. The main advantages are the availability of high quality, defect free products at low cost and at a faster time. The prefabricated components are customized as per the requirements of the customers. The components used are pre-built, ready-tested and add value and differentiation by rapid customization to the targeted customers. However the products we get from case tools are only a skeleton of the final product required and a lot of programming must be done by hand to get a fully finished, good product. Characteristics of CASE:

Some of the characteristics of case tools that make it better than customized development are; It is a graphic oriented tool. It supports decomposition of process. Some typical CASE tools are: Unified Modeling Language

## UNIFIED MODELING LANGUAGE

### Introduction

The unified modeling language (UML) is a standard language for writing software blue prints of the system.

### Definition:

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

The artifacts of a software system.

- UML is a language that provides vocabulary and the rules for combing words in that vocabulary for the purpose of communication.
- Vocabulary and rules of a language tell us how to create and real well formed models, but they don't tell you what model you should create and when should create them.

**Building Blocks of the UML:**

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

1. Things
2. Relationships
3. Diagrams

Things are abstractions that are first-class citizens in a model; Relationships tie these things together; Diagrams group interesting collections of things.

**Things**

Things are the most important building blocks of UML. There are four kinds of things in the UML.

1. Structural things
2. Behavioral things
3. Grouping things
4. An notational things

**1) STRUCTURAL THINGS**:

Structural things are the nouns of the UML models.

These are static parts of the model, representing elements that are either conceptual or physical.

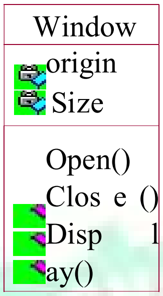
There are seven kinds of Structural things.

1. Class
2. Interface
3. Collaboration
4. Use case
5. Active class
6. Component
7. Node

**Class:**

A class is a description of a set of objects that shares the common attributes, operations, relationships, and semantics. A class implements one or more interfaces.

Graphically, a class is represented as a rectangle, usually including its name, attributes and operations, as shown below.



**Interface:**

An interface is a collection of operations that specify a service of a class or component. An interface describes the externally visible behavior of that element.

Graphically the interface is rendered as a circle together with its name.

**Collaboration:**

Collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Graphically, collaboration is rendered as an ellipse with dashed lines, usually including only its name as shown below.

**Use Case:**

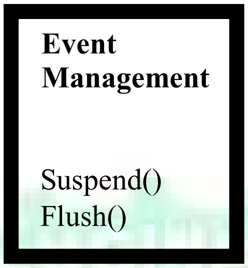
Use case is a description of a set of sequence of actions performed by a system for a specific goal for the system.

Graphically, Use Case is rendered as an ellipse with dashed lines, usually including only its name as shown below.

**Active Class:**

An active class is a class whose objects own one or more processes or threads and therefore can initiate control activity.

Graphically, an active class is rendered just like a class, but with heavy lines usually including its name, attributes and operations as shown below.



**Component:**

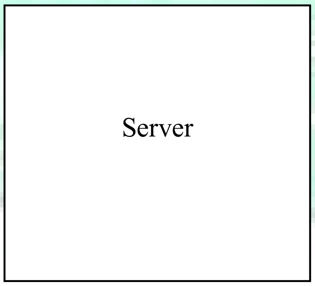
Component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces.

Graphically, a component is rendered as a rectangle with tabs, usually including only its name, as shown below.

**Node:**

A Node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and often, processing capability.

Graphically, a node is rendered as a cube, usually including only its name, as shown below.



**2) Behavioral things:**

Behavioral things are the dynamic parts of UML models.

***1) Interaction:***

An interaction is a behavior that consists of a set of messages exchanged among a set of objects(elements) within a particular context to accomplish a specific task.

Graphically, a message is rendered as a direct line, almost always including the name if its operation, as shown below.

***2) State Machine:***

A state machine is a behavior that specifies the sequence of states of an object in its life cycle. It defines the sequence of states an object goes through in response to events.

Graphically, a state is rendered as a rounded rectangle usually including its name and its sub- states, if any, as shown below.

***3)* Grouping things:**

Grouping things are the organizational parts of the UML models. These are the boxes into which a model can be decomposed.

There is one primary kind of grouping thing with "package".

**Package:**

A package is a general-purpose mechanism for organizing elements into groups.

Package is the only one grouping thing available for gathering structural and behavioral things.

Business Rules

***4)*** **Annotational things:**

Annotational things are the explanatory parts of the UML models.

Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.

**Note:**

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

Graphically a note is represented as a rectangle with dog-eared corner together, with a textual or graphical comment, as shown below.

**RELATIONSHIPS IN THE UML:**

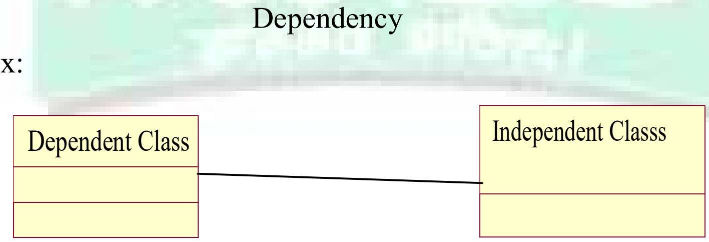
Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application. There are four kinds of relationships in the UML:

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

***Dependency***

Dependency is a relationship between two things in which change in one element also affects the other one.

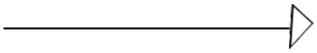
Ex:



***:***

***Generalization:***

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

Ex: 

***Realization:***

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.

Ex:

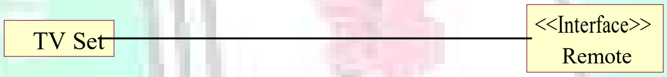


**Diagram sinuml:**

All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

The visual effect of the UML diagram is the most important part of the entire process.

Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction.

UML diagrams are the ultimate output of the entire system.

A diagram is the graphical presentation of a set of elements; most often rendered as a connected graph of vertices (things) arcs (relationships).

UML includes the following nine diagrams:

1) Class diagram
2) Object diagram
3) Use case diagram
4) Sequence diagram
5) Collaboration diagram
6) Activity diagram
7) State chart diagram
8) Deployment diagram
9) Component diagram

### 1. Class Diagram

Class diagram is a diagram that shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams address the static design view or the static process view of the system.

Graphically it is represented as follows:-

### 2. Object Diagram

Object diagram shows a set of objects and their relationships. These diagram the static design view or static process view of a system.

### 3. Use case Diagram

Use Case diagram shows a set of use cases and actors (a special kind of class) and their relationships.

## 4 Sequence Diagram

Sequence diagram are interaction diagrams. This diagram emphasizes the time-ordering of messages. These diagrams address the dynamic view of a system. Sequence Diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).Graphically it is represented as
follows:-

## 5 Collaboration Diagram

Collaboration diagram are also interaction diagrams. These diagrams emphasize the structural organization of the objects that send and receive messages. These diagrams address the dynamic view of a system. Collaboration Diagram displays an interaction organized around the objects and their links to one another. Numbers are used to show the sequence of messages. Graphically it is represented as follows:-

## 6. State chart Diagram

State chart diagram shows a state machine, consisting of states, transitions, events and activities. These diagrams address the dynamic view of the system. State Chart diagram displays the sequences of states that an object of an interaction goes through during its life in response to received stimuli, together with its responses and actions.

## 7 .Activity Diagram

Activity diagram is a special kind of a state chart diagram that shows the flow from activity to activity within a system. These diagrams address dynamic view of a system. Activity Diagram displays a special

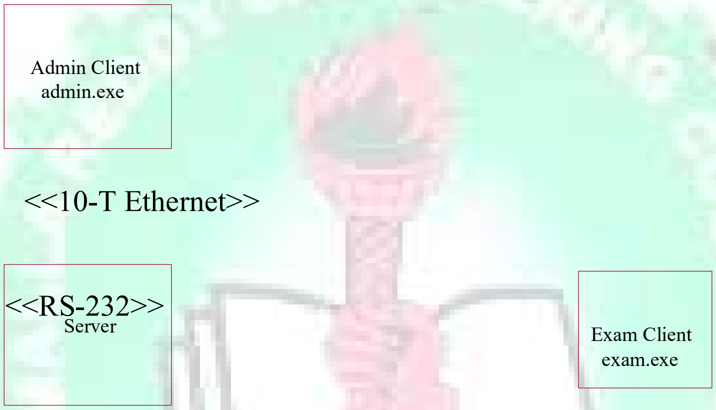
## 8. Component Diagram

Component diagram shows the organizations and dependencies among a set of components. These diagrams address the static implementation of view of a system. Component Diagram displays the high level packaged structure of the code itself. Dependencies among components are shown, including source code components, binary code components, and executable components. Some components exist at compile time, at link time, at run times well as at more than one time. Graphically it is represented as follows:-

## 8. Deployment Diagram

Deployment diagram shows the configuration of run-time processing nodes and the components that live on them. These diagrams address the static deployment view of architecture. Deployment Diagram displays the configuration of run-time processing elements and the software components, processes, and objects that live on them. Software component instances represent run-time manifestations of code.

Graphically it is represented as follows:-



## Automatic Teller Machine (ATM)

### Description of ATM System

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash, a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) – both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned – except as noted below.

The ATM must be able to provide the following services to the customer:

A customer must be able to make a cash withdrawal from any suitable account linked to the card. Approval must be obtained from the bank before cash is dispensed.

A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.

A customer must be able to make a transfer of money between any two accounts linked to the card.

A customer must be able to make a balance inquiry of any account linked to the card.

A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

**Objectives**

The objective of this software is similar to ATM software installed in ATM center. It should first validate the pin in the ATM card. Then the type of transaction is enquired and the information from the customer is validated. If it is a withdrawal the amount is asked. After the money is delivered the transaction just made is updated in the database where the customer's information is stored.

**Scope**

The scope of the project is to design an ATM system that will help in completely automatic banking this software is going to be designed for withdrawal and deposit of money and register the transaction in the database where the customer's information is stored.

**A) Name of the experiment: C**lass diagram for ATM System

**1. AIM**: To design and implement ATM system through Class Diagram

**Purpose:**

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community. So the purpose of the class diagram can be summarized as:

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

**Contents:**

Class diagrams commonly contain the following things

- Classes
- Interfaces
- Collaborations
- Dependency, generalization and association relationships

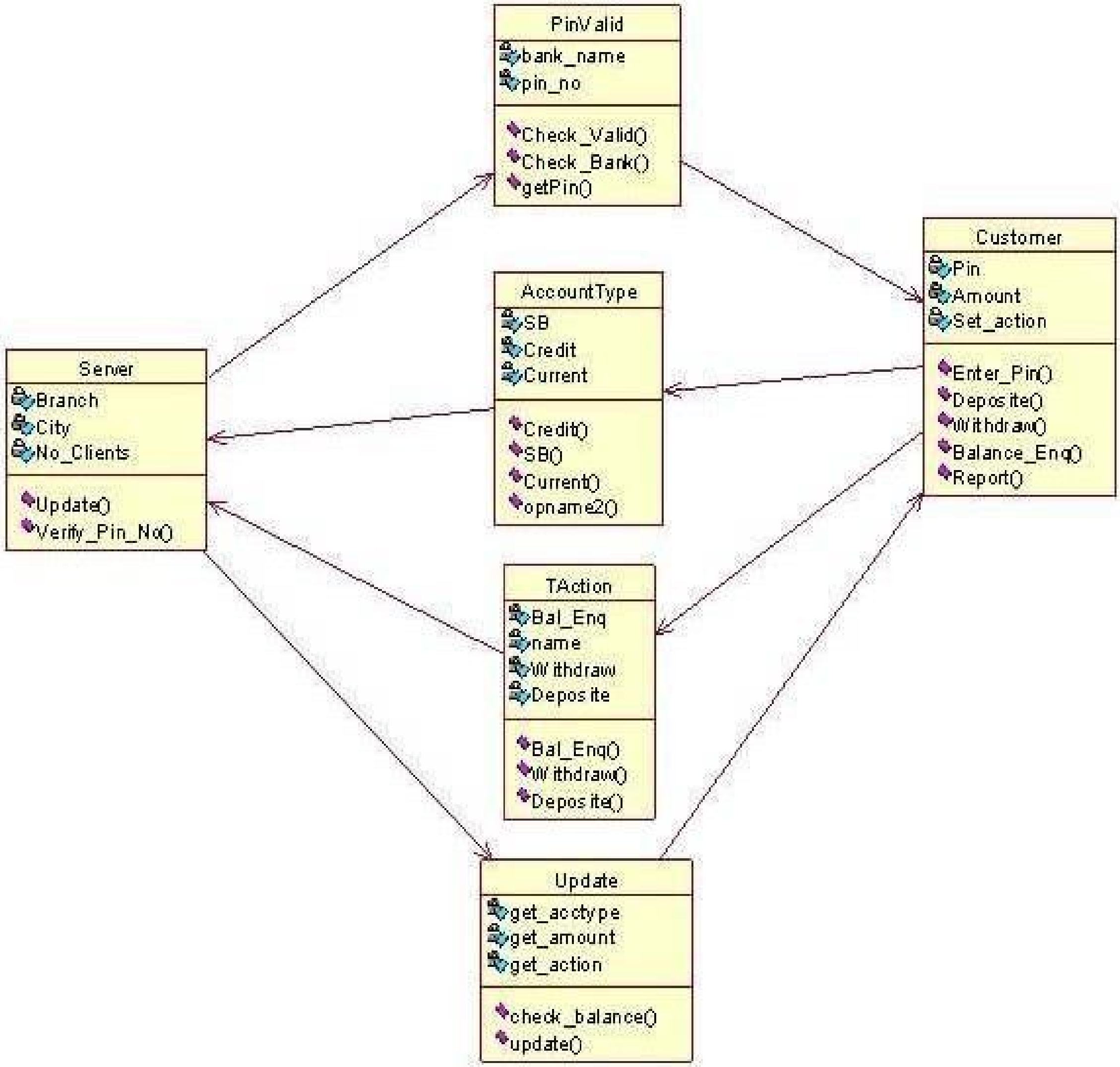
**Procedure:-**

**Step1**: First Classes are created.

**Step2**: Named as Pin Valid, Account Type, Transaction, Update, Server, Customer classes are created.

**Step3:** Appropriate relationships are provided between them as association.

**DIAGRAM:**



**Inferences:**

1. understand the concept of classes
2. identify classes and attributes and operations for a class
3. model the class diagram for the system

**Applications:**

- Online transaction
- Online banking

**B) Name of experiment:** Use case diagram for ATM System.

**AIM: To** design and implement ATM System through Use case Diagram.

**Purpose:**

The purpose of use case diagram is to capture the dynamic aspect of a system. Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

So in brief, the purposes of use case diagrams can be as follows:

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

**Procedure:**

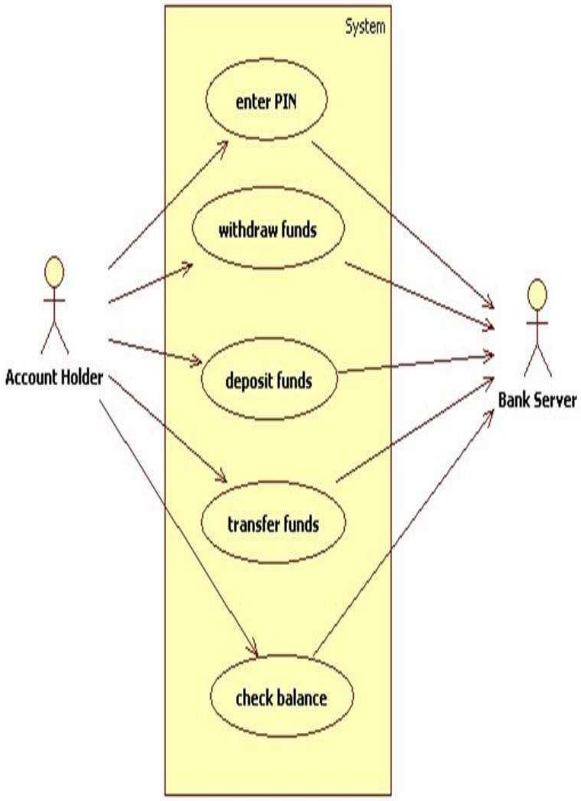
**Step1:** First an Actor is Created and named as User/Customer.

**Step2:** Secondly a system is created for ATM.

**Step3:** A use case Enter PIN, Withdraw money is created and connected with user as association relationship.

**Step4**: Similarly various use cases like Deposit money, Balance Enquiry, Manage Account etc are created and appropriate relationships are associated with each of them.

**DIAGRAM:**


System
enter PIN
withdraw funds
deposit funds
transfer funds
check balance
Account Holder
Bank Server

**Withdrawal Use Case**

A withdrawal transaction asks the customer to choose a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and to choose an amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request before sending the transaction to the bank. (If not, the customer is informed and asked to enter a different amount.) If the transaction is approved by the bank, the appropriate amount of cash is dispensed by the machine before it issues a receipt. A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the amount.

**Deposit Use Case**

A deposit transaction asks the customer to choose a type of account to deposit to (e.g. checking) from a menu of possible accounts, and to type in amount on the keyboard. The transaction is initially sent to the bank to verify that the ATM can accept a deposit from this customer to this account. If the transaction is approved, the machine accepts an envelope from the customer containing cash and/or checks before it issues a receipt. Once the envelope has been received, a second message is sent to the bank, to confirm that the bank can credit the customer's account – contingent on manual verification of the deposit envelope contents by an operator later.

A deposit transaction can be cancelled by the customer pressing the Cancel key any time prior to inserting the envelope containing the deposit. The transaction is automatically cancelled if the customer fails to insert the envelope containing the deposit within a reasonable period of time after being asked to do so.

**Inquiry Use Case**

An inquiry transaction asks the customer to choose a type of account to inquire about from a menu of possible accounts. No further action is required once the transaction is approved by the bank before printing the receipt. An inquiry transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the account to inquire about.

**Validate User Use case:**

This use case is for validate the user i.e. check the pin number, when the bank reports that the customer's transaction is disapproved due to an invalid PIN. The customer is required to re-enter the PIN and the original request is sent to the bank again. If the bank now approves the transaction, or disapproves it for some other reason, the original use case is continued; otherwise the process of re- entering the PIN is repeated. Once the PIN is successfully re-entered

If the customer fails three times to enter the correct PIN, the card is permanently retained, a screen is displayed informing the customer of this and suggesting he/she contact the bank, and the entire customer session is aborted.

**Print Bill use case**

This use case is for printing corresponding bill after transactions (withdraw or deposit, or balance enquiry, transfer) are completed.

**Manage Account**

This use case is for updating corresponding user accounts after transactions (withdraw or deposit or transfer) are completed.

**RESULT:**

**Inferences:**

1. Identification of use cases.
2. Identification of actors.

**Interaction Diagrams**

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

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

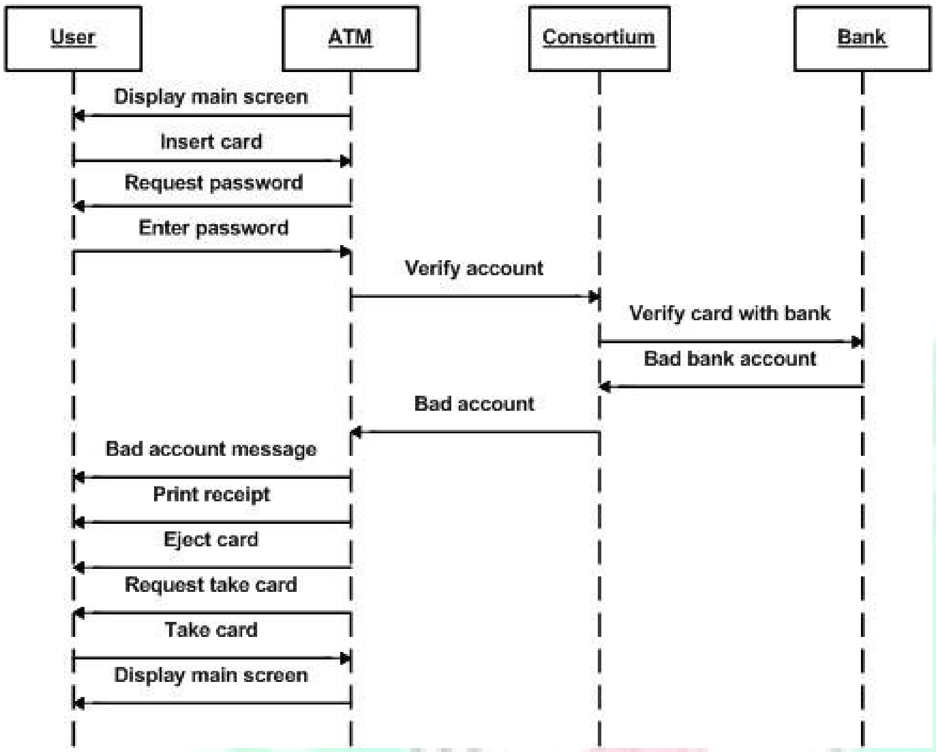
1. Objects taking part in the interaction.
2. Message flows among the objects.
3. The sequence in which the messages are flowing.
4. Object organization.

**Purpose:**

1. To capture dynamic behavior of a system.
2. To describe the message flow in the system.
3. To describe structural organization of the objects.
4. To describe interaction among objects.

**C) Name of Experiment:** Sequence diagram for ATM System.

**AIM: To** design and implement ATM System through Sequence Diagram.



**Procedure:-**

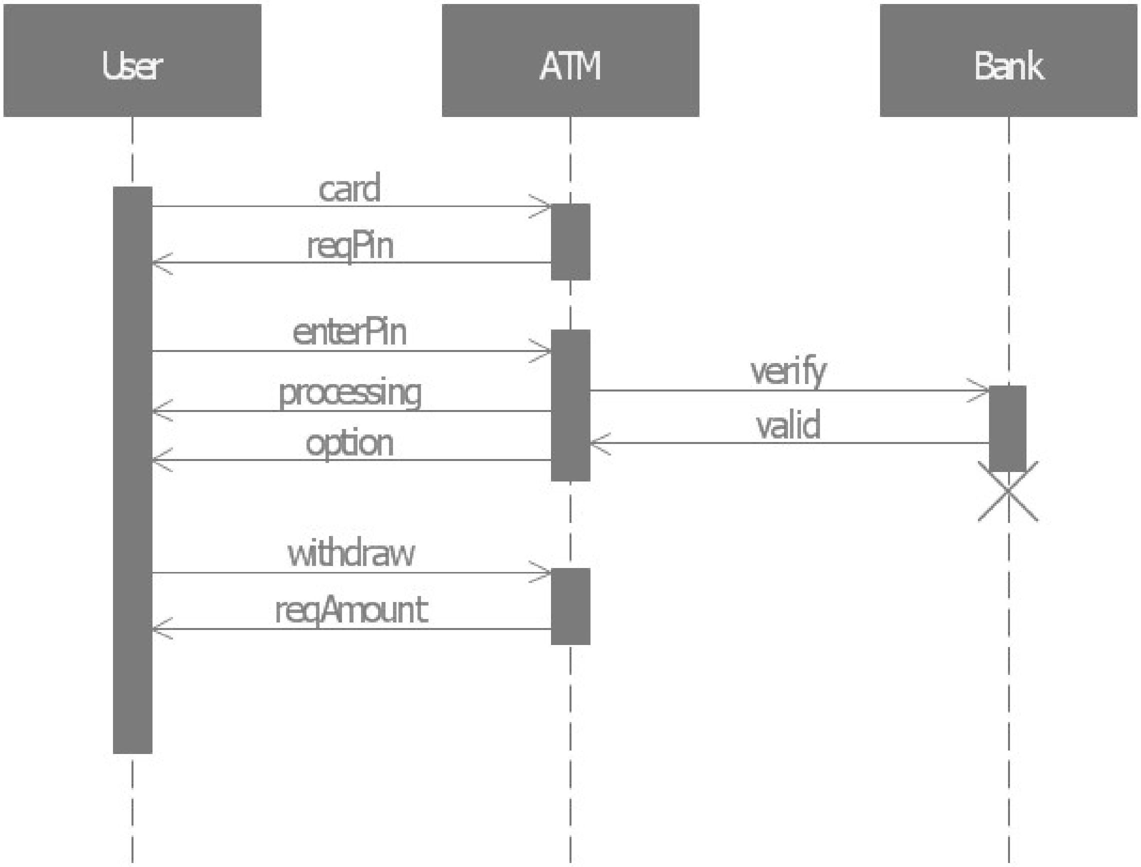
**Step1:** First An actor is created and named as user.

**Step2:** Secondly an object is created for Atm.

**Step3:** Timelines and lifelines are created automatically for them.

**Step4**: In sequence diagram interaction is done through time ordering of messages. So appropriate messages are passed between user and ATM is as shown in the figure.

Diagram: sequence diagram



**D) Name of experiment:** collaboration for ATM System.

**AIM:** To design and implement ATM System through Collaboration diagram.

**Procedure:-**

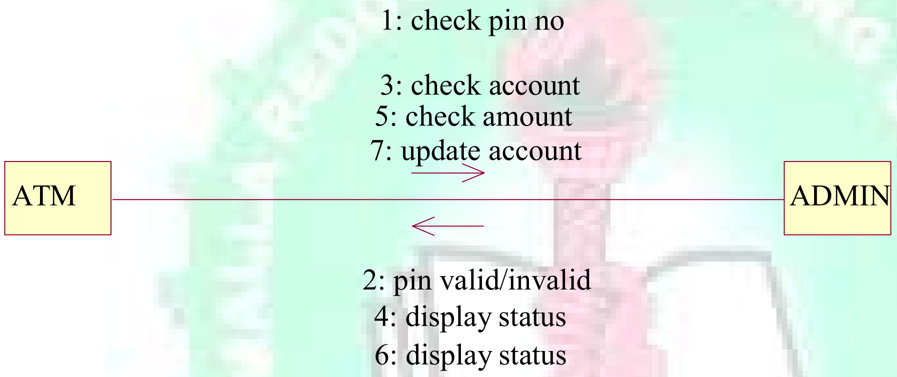
**Step1:** First an actor is created and named as user.

**Step2:** Secondly an object is created for ATM.

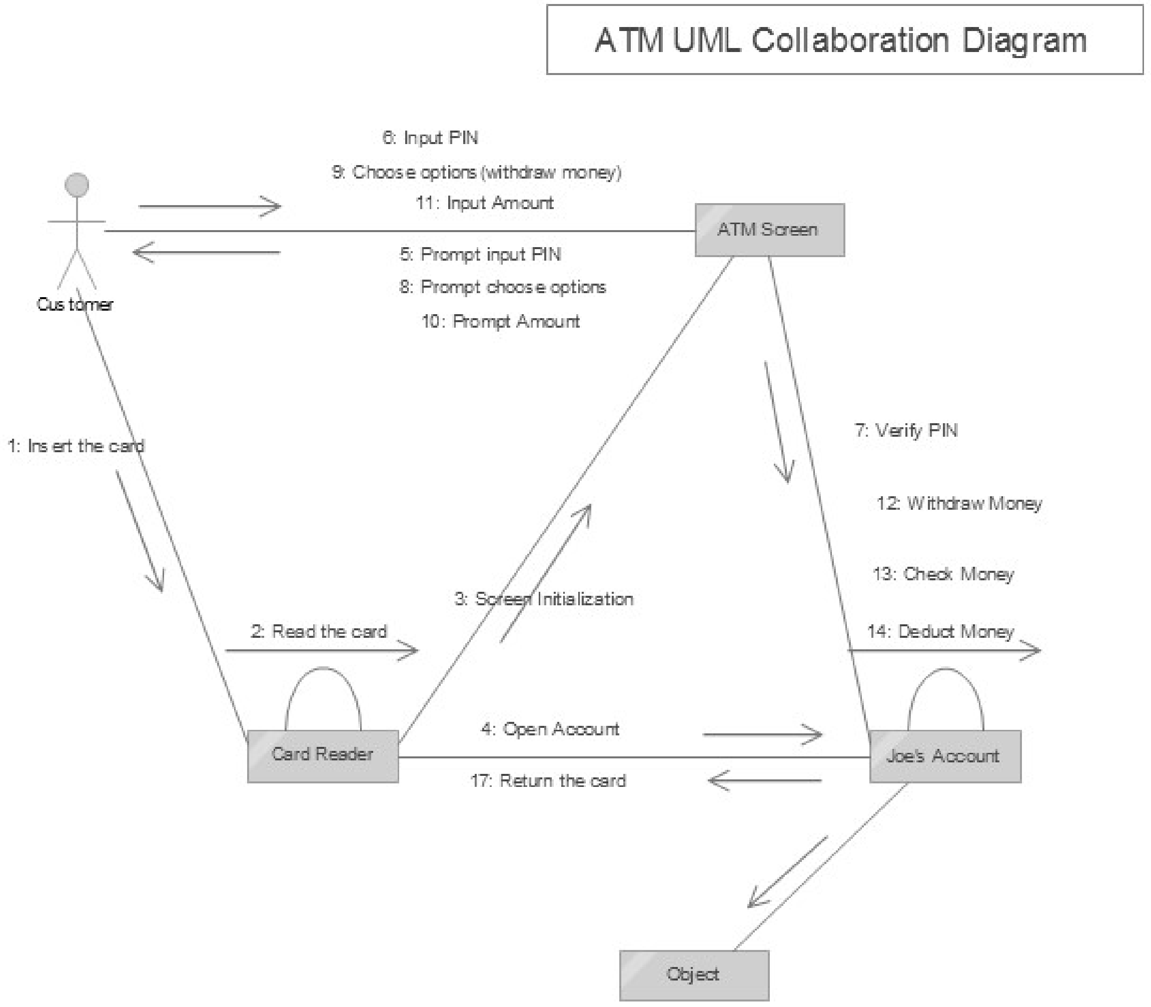
**Step3:** In collaboration diagram interaction is done through organization.

**Step4:** So appropriate messages are passed between user and ATM as shown in the figure.

**DIAGRAM:**



Collaboration diagram


ATM UML Collaboration Diagram
6: Input PIN
9: Choose options (withdraw money)
11: Input Amount
5: Prompt input PIN
8: Prompt choose options
10: Prompt Amount
Customer
ATM Screen
1: Insert the card
7: Verify PIN
12: Withdraw Money
13: Check Money
14: Deduct Money
3: Screen Initialization
2: Read the card
4: Open Account
17: Return the card
Card Reader
Joe's Account
Object

### State Chart Diagram

State Chart diagram is used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. State chart diagram describes the flow of control from one state to another state.

**Contents**

- Simply state and composite states
- Transitions, including events and actions

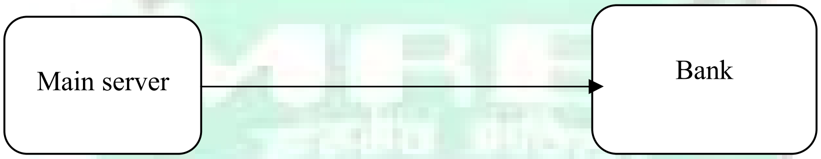
**E) Name of experiment:** State chart diagram for ATM System.

**Aim:** To design and implement ATM System through State Chart diagram.

**Purpose:**

Following are the main purposes of using State chart diagrams:

1. To model dynamic aspect of a system.
2. To model life time of a reactive system.



3. To describe different states of an object during its life time.
4. Defines a state machine to model states of an object.

**Procedure:-**

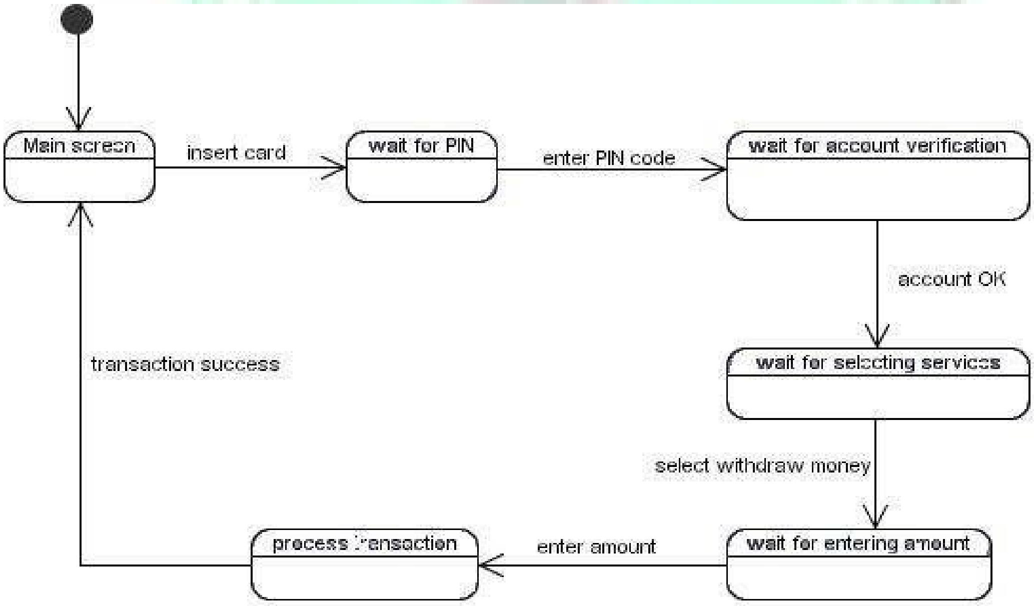
**Step1:** First after initial state control undergoes transition to ATM screen.

**Step2:** After inserting card it goes to the state wait for pin.

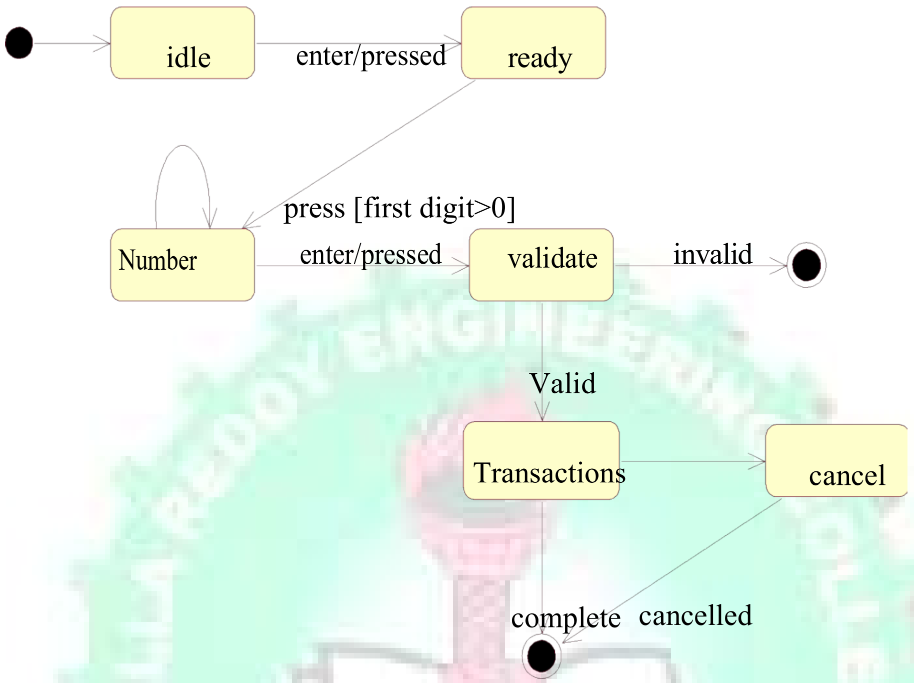
**Step3:** After entering pin it goes to the state account verification.

**Step4:**. In this way it undergoes transitions to various states and finally reaches the ATM screen state as shown in the fig.

**DIAGRAM:**



STATE CHART FOR ATM



**F) Name of experiment:** Activity diagram forATM System.

**Aim:** To design and implement ATM System through Activity Diagram**.**

**Theory:** An activity diagram shows the flow from activity to activity .An activity is an ongoing non atomic execution within a state machine .Activities ultimately results in some action, which is made up of executable atomic computations. We can use these diagrams to model the dynamic aspects of a system.

Activity diagram is basically a flow chart to represent the flow form one activity to another . 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 by using elements like fork, join etc.

**Contents**

Initial/Final State, Activity, Fork & Join, Branch, Swim lanes

**Fork**

A fork represents the splitting of a single flow of control into two or more concurrent Flow of control. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. Below fork the activities associated with each of these path continues in parallel.

**Join**

A join represents the synchronization of two or more concurrent flows of control. A join may have two or more incoming transition and one outgoing transition. Above the join the activities associated with each of these paths continues in parallel.

**Branching**

A branch specifies alternate paths takes based on some Boolean expression Branch is represented by diamond Branch may have one incoming transition and two or more outgoing one on each outgoing transition, you place a Boolean expression shouldn't overlap but they should cover all possibilities.

**Swim lane:**

Swim lanes are useful when we model workflows of business processes to partition the activity states on an activity diagram into groups. Each group representing the business organization responsible for those activities, these groups are called Swim lanes.

**Procedure:-**

**Step1:** First initial state is created.

**Step2:** After that it goes to the action state insert card.

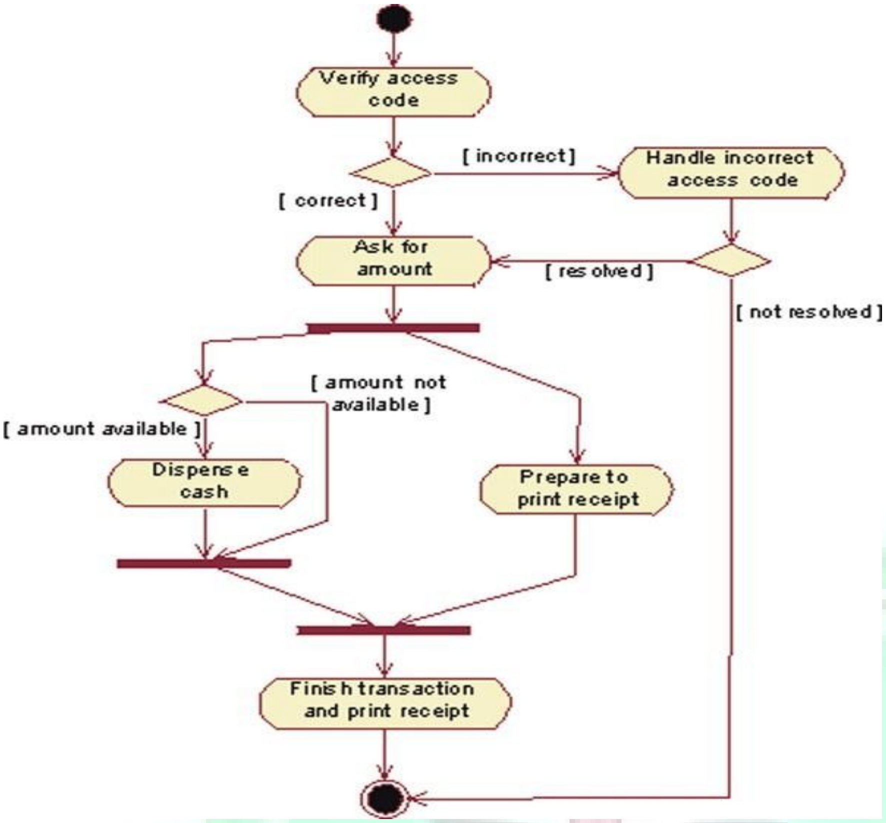
**Step3:** Next it undergoes transition to the state enter pin

**Step4**: In this way it undergoes transitions to the various

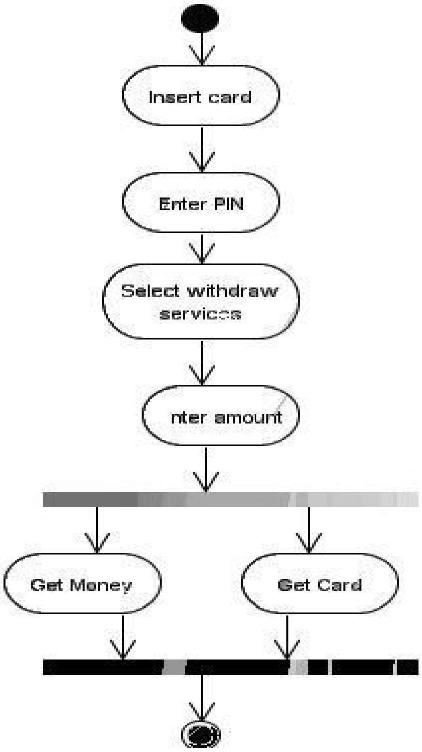
states. **Step5:** Use forking and joining wherever necessary.

**DIAGRAM:**

Activity diagram for Transactions:



Activity diagram for Withdraw:



**G) Name of experiment:** Component diagram for ATM System.

**AIM:** To design and implement Component diagram for ATM

System**. Theory:**

Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

**Purpose:**

Component diagrams can 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.

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.

**Contents**

Components, Interfaces, Relationships

**Procedure:-**

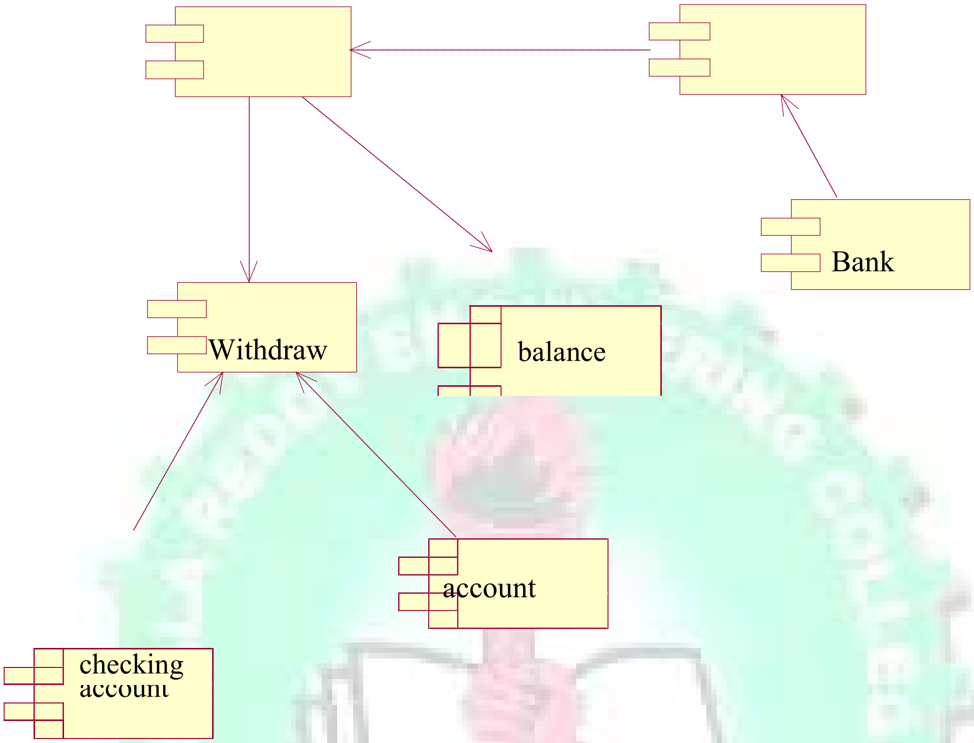
**Step1:** First user component is created.

**Step2:** ATM system package is created.

**Step3:** In it various components such as withdraw money, deposit money, check balance, transfer money etc. are created.

**Step 4:** Association relationship is established between user and other components.

**DIAGRAM:**



**Name of experiment:** Deployment diagram for ATM System**.**

**Aim:** To design and implement ATM System through Deployment diagram.

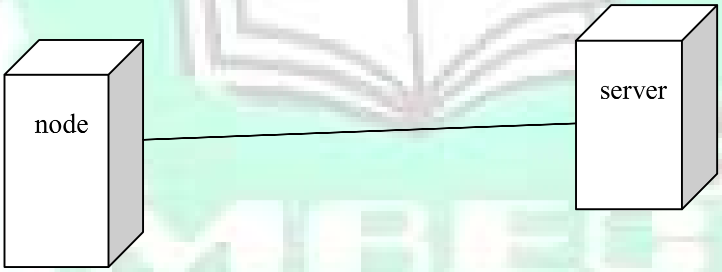
**Purpose:**

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 are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

**Contents:** Nodes, Dependency & Association relationship

**Procedure**:

First user node is created

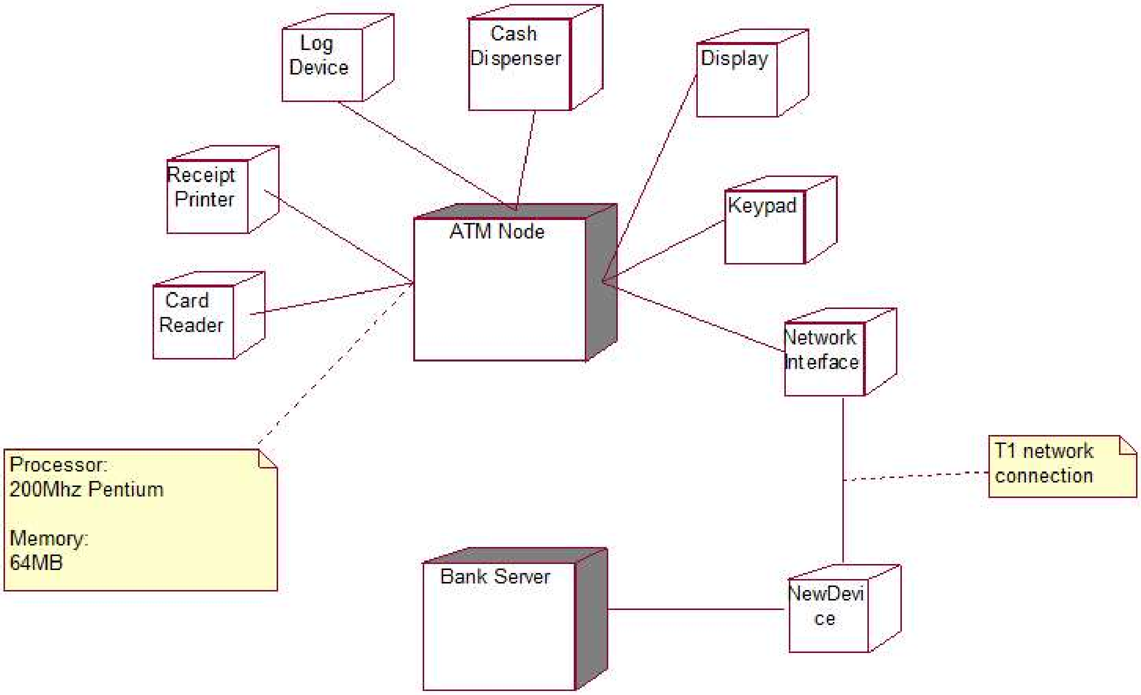


**Step1:** various nodes withdraw money, deposit money, and check balance, transfer money etc. are created.

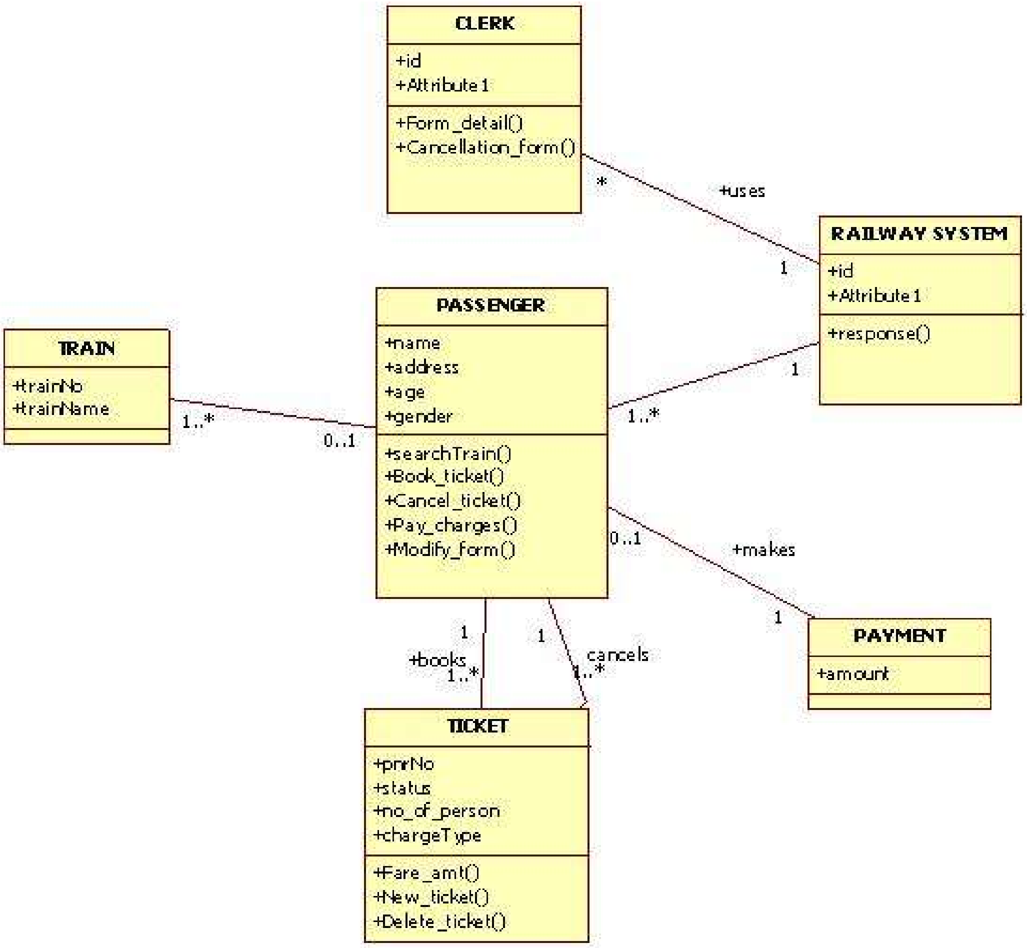
**Step2:** Association relationship is established between user and other nodes.

**Step3:** Dependency is established between deposit money and check balance.

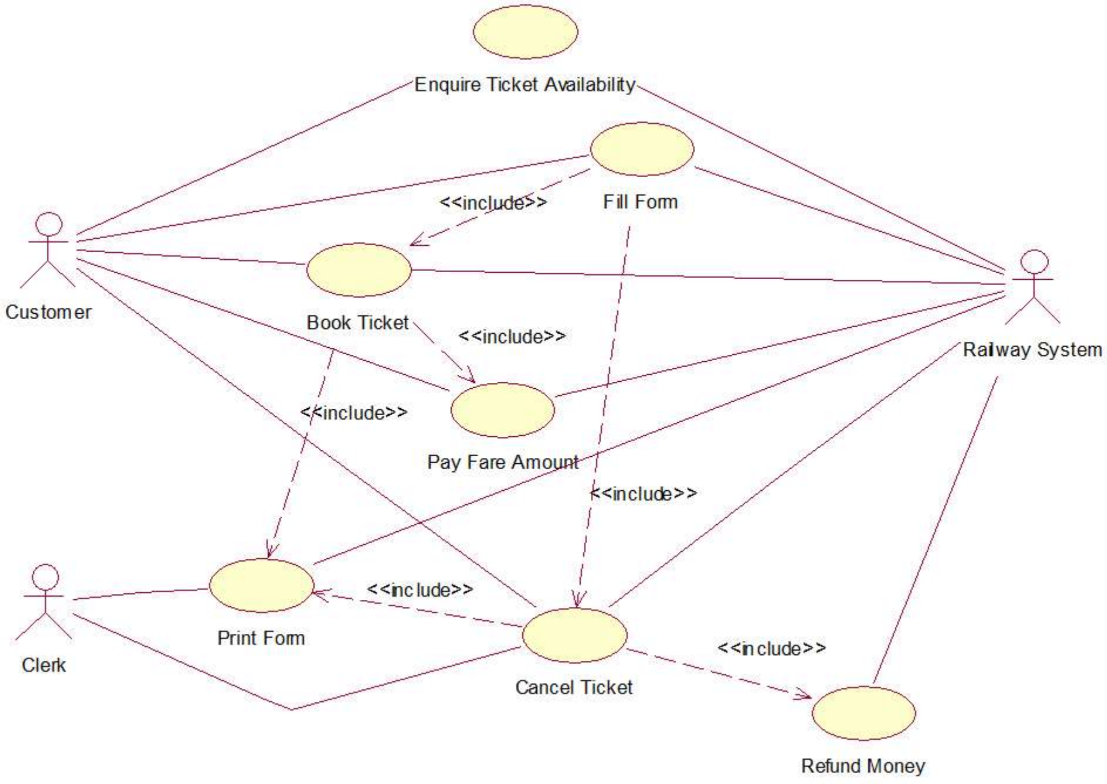
**Deployment diagram**



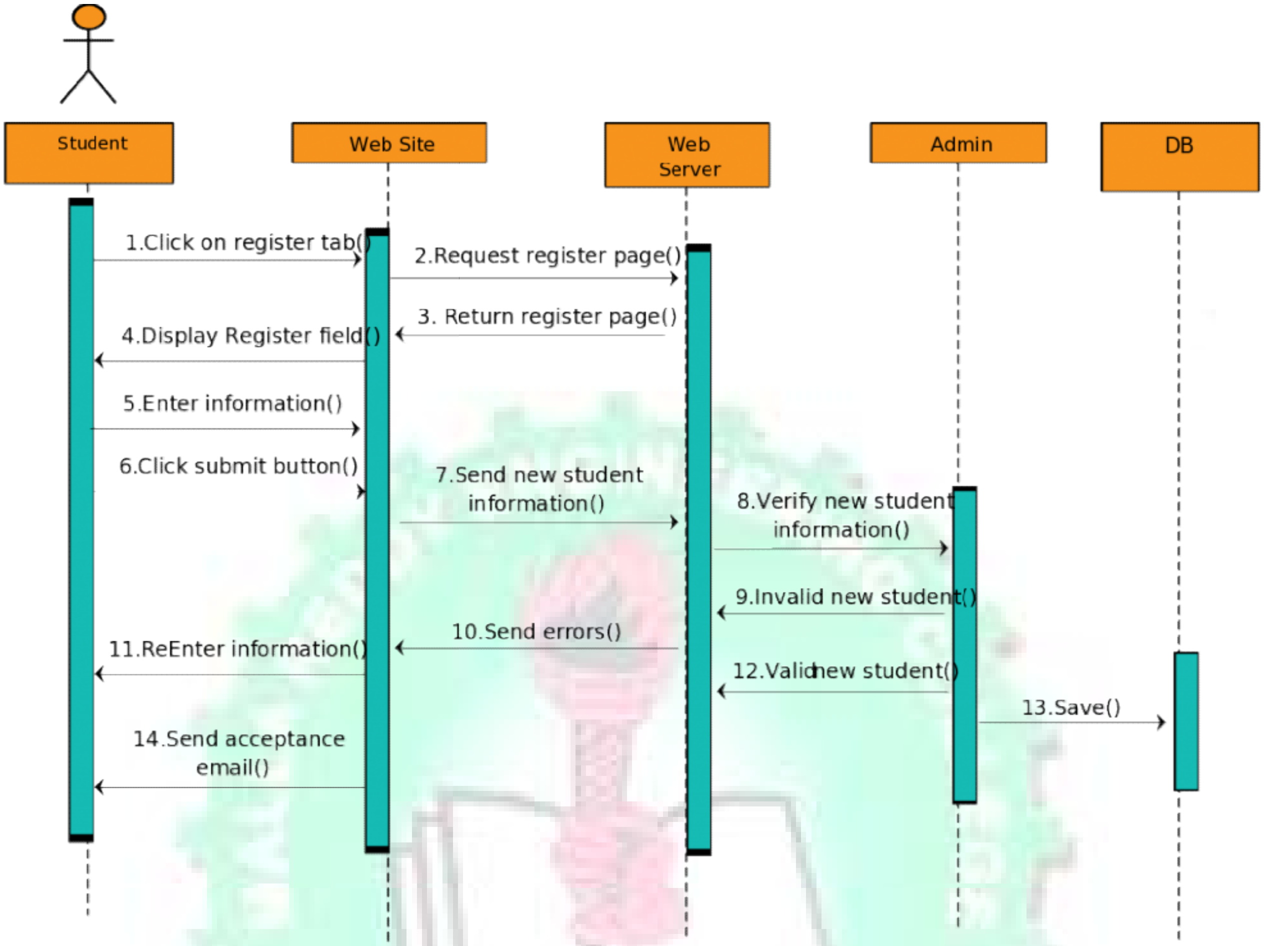
Class Diagram for Online Reservation System



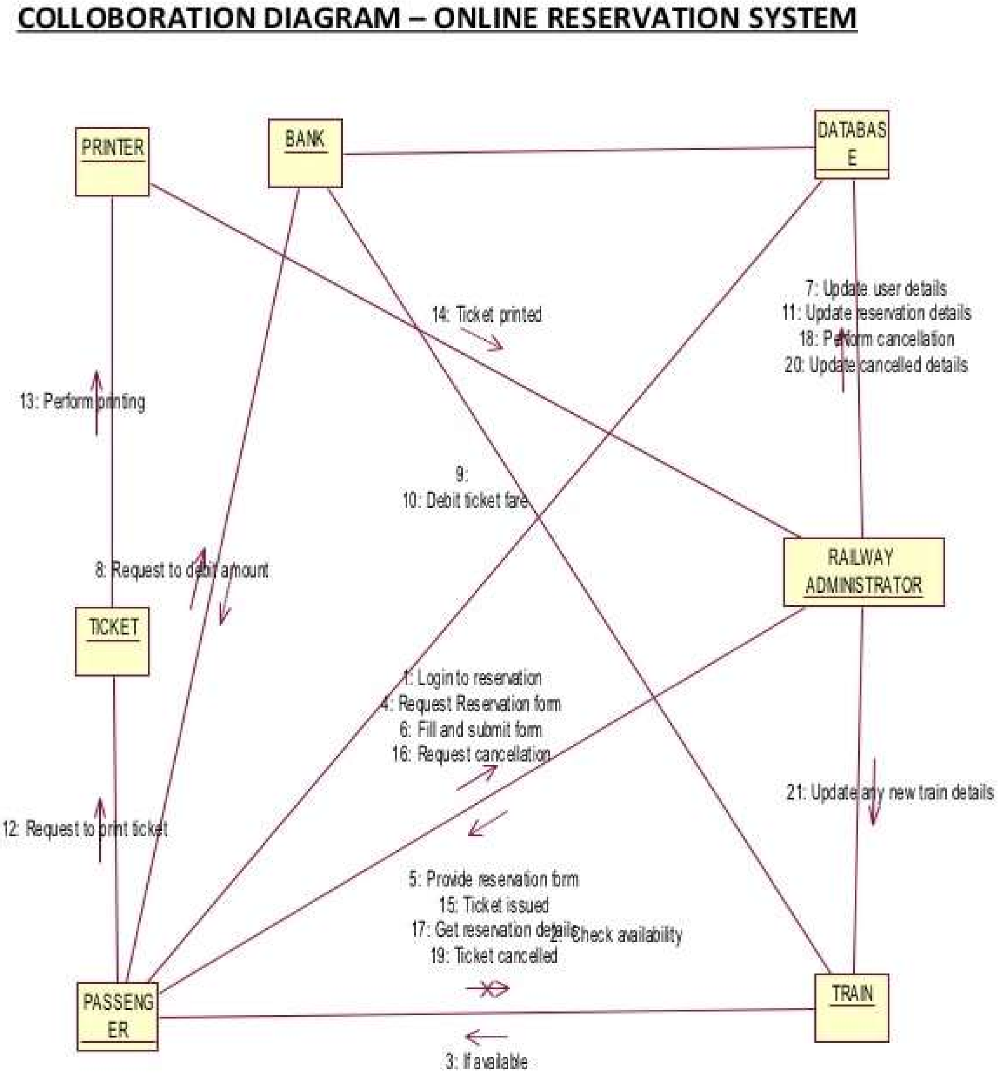
Use case Diagram for online reservation system diagram



Sequence Diagram for Online Reservation System

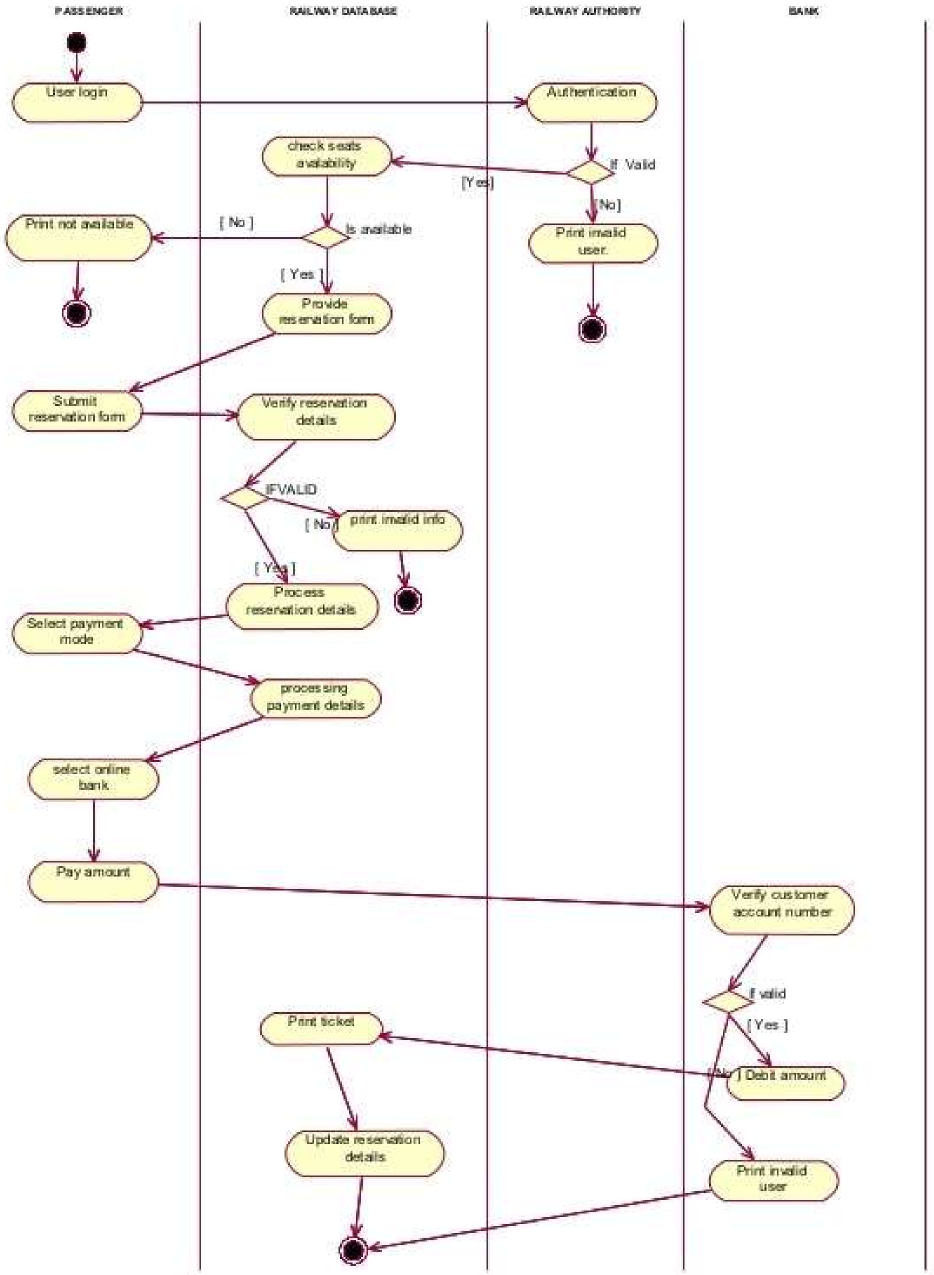

Student
Web Site
Web Server
Admin
DB
1.Click on register tab()
2.Request register page()
3. Return register page()
4.Display Register field()
5.Enter information()
6.Click submit button()
7.Send new student information()
8.Verify new student information()
9.Invalid new student()
10.Send errors()
11.ReEnter information()
12.Validnew student()
13.Save()
14.Send acceptance email()

Collaboration Diagram for Online Reservation System.


COLLOBORATION DIAGRAM – ONLINE RESERVATION SYSTEM

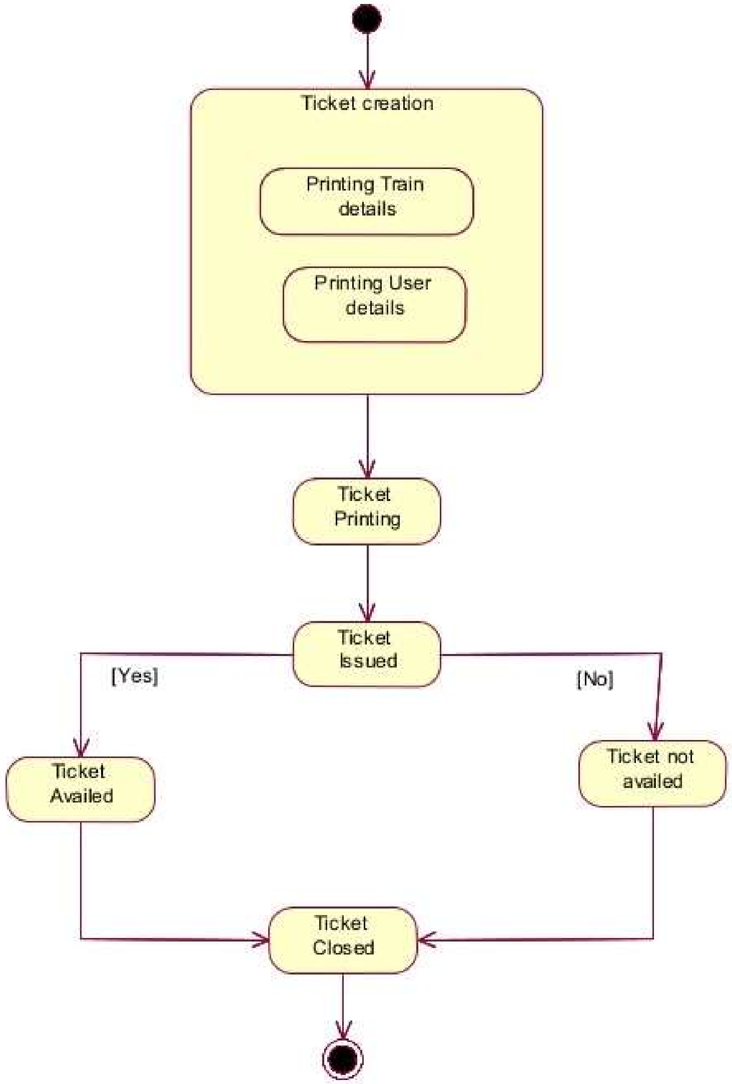
Activity Diagram for Online Reservation System

## ACTIVITY DIAGRAM-ONLINE RAILWAY RESERVATION SYSTEM

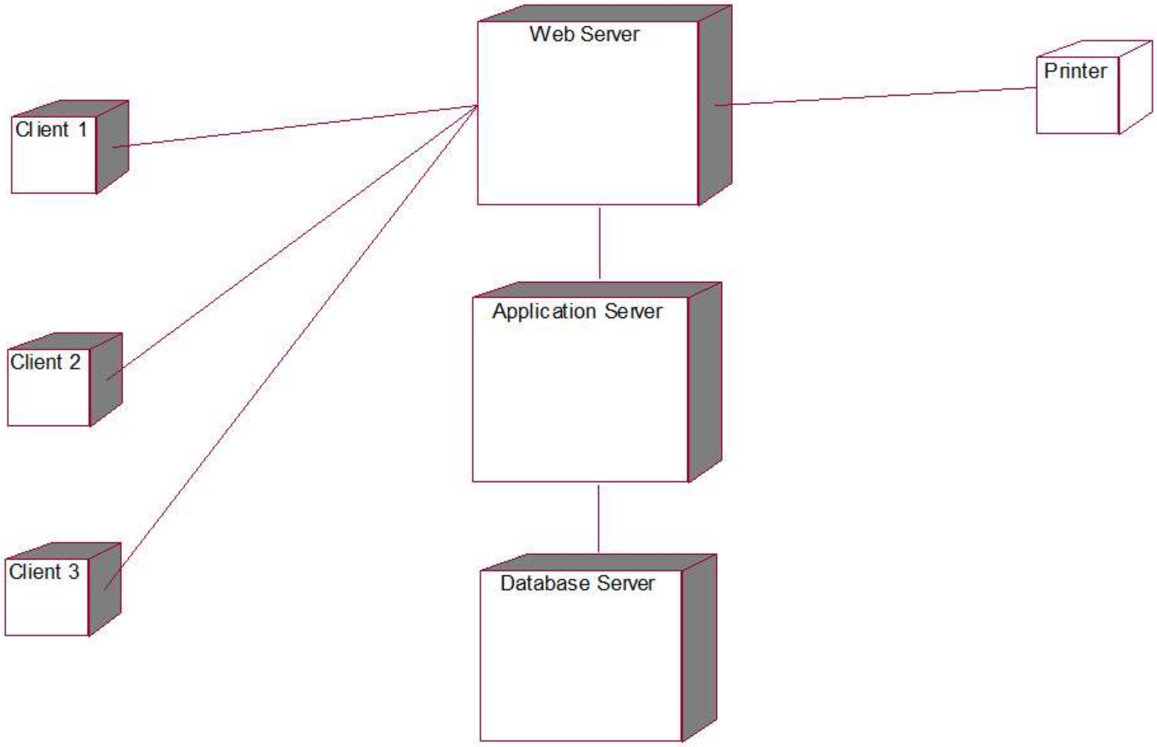


State Chart Diagram for Online Reservation System.

## STATECHART DIAGRAM-ONLINE RAILWAY RESERVATION SYATEM



Component Diagram for Online Reservation System.


Web Server
Printer
Client 1
Client 2
Client 3
Application Server
Database Server