# Lab File

Software Engineering

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING



|  |  |
| --- | --- |
| Submitted to: | Submitted By: |
| Teacher’s Name | XYZ |
| Assistant Professor | Enrollement No |
| CSE Department, ASET | Section |
|  | BTech |

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**

# AMITY UNIVERSITY UTTAR PRADESH

# NOIDA-201301

**LIST OF PROGRAMS**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.  No. | Name of Programs | Page No. | Sign. |
| 1. | Software Development Life Cycle. |  |  |
| 2. | Software Requirement Specification. |  |  |
| 3. | Use Case Diagram of Bank Application Management. |  |  |
| 4. | Class and object diagram for Bank Application System. |  |  |
| 5. | Activity Diagram for ATM. |  |  |
| 6. | Sequence Diagram for Bank Application System. |  |  |
| 7. | Deployment diagram for Vehicle Registration System. |  |  |
| 8. | Component diagram for Vehicle Registration System. |  |  |
| 9. | Write a program to calculate function point analysis. |  |  |
| 10. | Use case diagram for amizone. |  |  |
| 11. | Use case diagram for bank application. |  |  |
| 12. | Class diagram for amizone. |  |  |
| 13. | Class diagram for bank application. |  |  |
| 14. | Object diagram for amizone. |  |  |
| 15. | Object diagram for bank application. |  |  |
| 16. | Activity diagram for amizone. |  |  |
| 17. | Activity diagram for bank application. |  |  |

**PROGRAM NO. 1**

**Objective:-** Study the complete Software Development Life Cycle (SDLC) and analyse various activities conducted as a part of various phases. For each SDLC phase, identify the objective and summaries outcomes.

### Result:

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

A typical Software Development Life Cycle consists of the following stages −

## Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks. Outcome of this stage is a requirement specification.

## Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

## Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

## Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

## Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**PROGRAM NO. 2**

**Objective:-** Consider any project to be developed in any technology as a Software Architect or Project Manager. Construct Software Requirement Specification for the project.

### Result:

### 1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to build an online system to manage flights and passengers to ease the flight management. <<*Include the purpose as applicable to your project* >>

1.2 DOCUMENT CONVENTIONS

This document uses the following conventions.

|  |  |
| --- | --- |
| DB | Database |
| DDB | Distributed Database |
| ER | Entity Relationship |

1**.**3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is a prototype for the flight management system and it is restricted within the college premises. This has been implemented under the guidance of college professors. This project is useful for the flight management team and as well as to the passengers.

1.4 PROJECT SCOPE

The purpose of the online flight management system is to ease flight management and to create a convenient and easy-to-use application for passengers, trying to buy airline tickets. The system is based on a relational database with its flight management and reservation functions. We will have a database server supporting hundreds of major cities around the world as well as thousands of flights by various airline companies. Above all, we hope to provide a comfortable user experience along with the best pricing available.

1.5 REFERENCES

▪ Fundamentals of database systems by ramez elmarsi and shamkant b.navathe

### 2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

A distributed airline database system stores the following information.

* Flight details:

It includes the originating flight terminal and destination terminal, along with the stops in between, the number of seats booked/available seats between two destinations etc.

* Customer description:

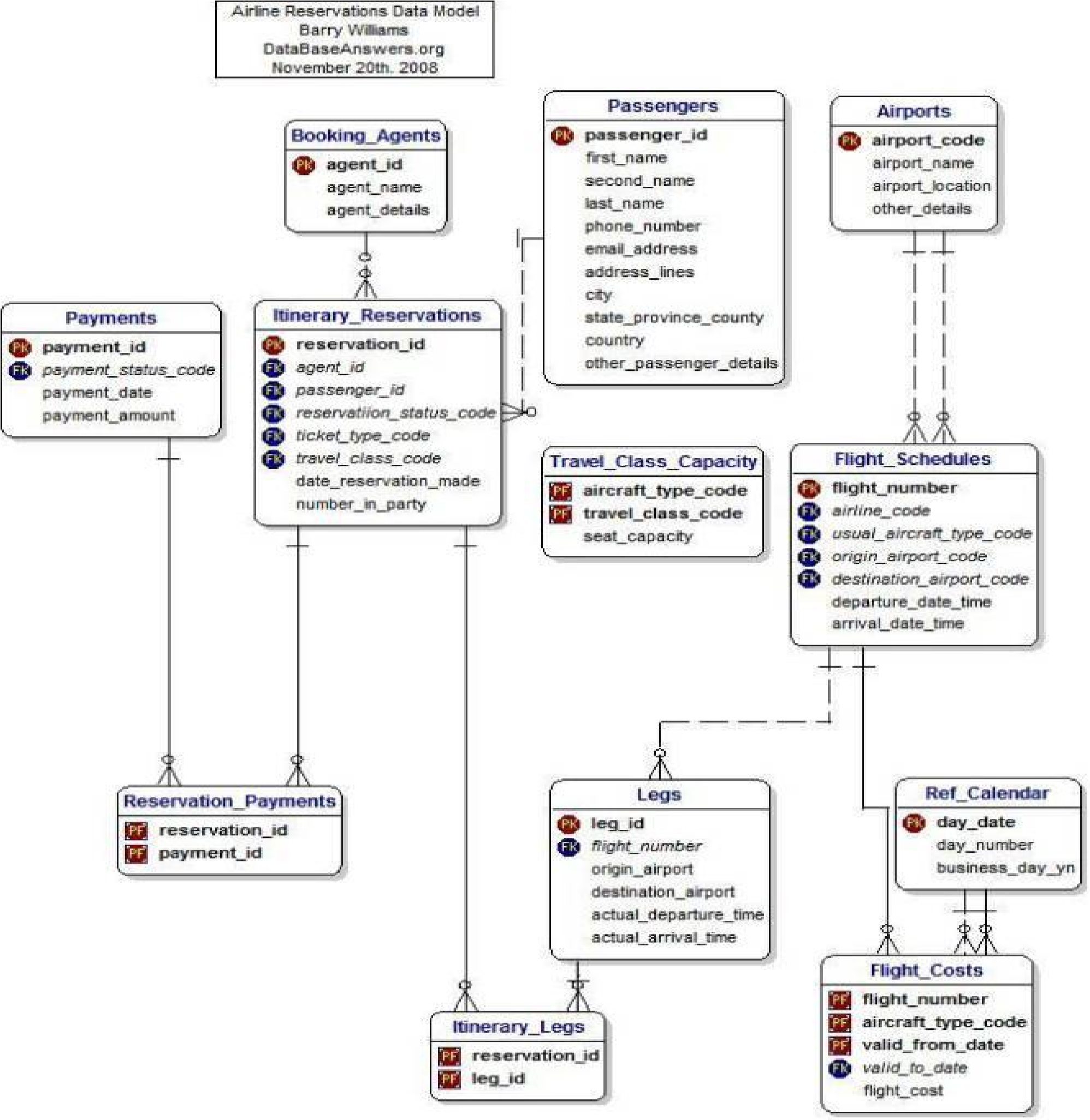
It includes customer code, name, address and phone number. This information may be used for keeping the records of the customer for any emergency or for any other kind of information.

* Reservation description:

It includes customer details, code number, flight number, date of booking, date of travel.

* 1. PRODUCT FEATURES

The major features of airline database system as shown in below entity–relationship model (ER model)



The diagram shows the layout of airline database system – entity–relationship model

* 1. USER CLASS and CHARACTERISTICS

Users of the system should be able to retrieve flight information between two given cities with the given date/time of travel from the database. A route from city A to city B is a sequence of connecting flights from A to B such that: a) there are at most two connecting stops, excluding the starting city and destination city of the trip, b) the connecting time is between one to two hours. The system will support two types of user privileges, Customer, and Employee. Customers will have access to customer functions, and the employees will have access to both customer and flight management functions. The customer should be able to do the following functions:

* Make a new reservation
  + - One-way
    - Round-Trip
    - Multi-city
    - Flexible Date/time
    - Confirmation
* Cancel an existing reservation
* View his itinerary

The Employee should have following management functionalities:

* CUSTOMER FUNCTIONS.
  + - Get all customers who have seats reserved on a given flight.
    - Get all flights for a given airport.
    - View flight schedule.
    - Get all flights whose arrival and departure times are on time/delayed.
    - Calculate total sales for a given flight.
* ADMINISTRATIVE
  + - Add/Delete a flight
    - Add a new airport
    - Update fare for flights.
    - Add a new flight leg instance.
    - Update departure/arrival times for flight leg instances.

Each flight has a limited number of available seats. There are a number of flights which depart from or arrive at different cities on different dates and time.

2.4 OPERATING ENVIRONMENT

Operating environment for the airline management system is as listed below. <<*Include the details as per your application* >>

* distributed database
* client/server system
* Operating system: Windows.
* database: sql+ database
* platform: vb.net/Java/PHP

2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

1. The global schema, fragmentation schema, and allocation schema.
2. SQL commands for above queries/applications
3. How the response for application 1 and 2 will be generated. Assuming these are global queries. Explain how various fragments will be combined to do so.
4. Implement the database at least using a centralized database management system.

2.6 ASSUMPTION DEPENDENCIES

Let us assume that this is a distributed airline management system and it is used in the following application:

* A request for booking/cancellation of a flight from any source to any destination, giving connected flights in case no direct flight between the specified SourceDestination pair exist.
* Calculation of high fliers (most frequent fliers) and calculating appropriate reward points for these fliers.

Assuming both the transactions are single transactions, we have designed a distributed database that is geographically dispersed at four cities Delhi, Mumbai, Chennai, and Kolkatta as shown in fig. below.

### 3. SYSTEM FEATURES

* DESCRIPTION and PRIORITY

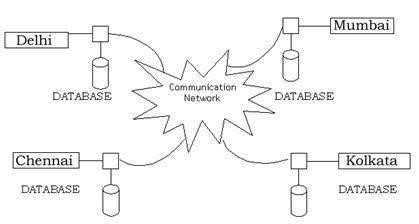
The airline reservation system maintains information on flights, classes of seats, personal preferences, prices, and bookings. Of course, this project has a high priority because it is very difficult to travel across countries without prior reservations.

* STIMULUS/RESPONSE SEQUENCES
* Search for Airline Flights for two Travel cities
* Displays a detailed list of available flights and make a “Reservation” or Book a ticket on a particular flight.
* Cancel an existing Reservation.
* FUNCTIONAL REQUIREMENTS

Other system features include:

DISTRIBUTED DATABASE:

Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network as shown in below figure.



*Distributed database located in four different cities*

CLIENT/SERVER SYSTEM

The term client/server refers primarily to an architecture or logical division of responsibilities, the client is the application (also known as the front-end), and the server is the DBMS (also known as the back-end).

A client/server system is a distributed system in which, ▪ Some sites are client sites and others are server sites.

* All the data resides at the server sites.
* All applications execute at the client sites.

### 4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

* Front-end software: Vb.net version
* Back-end software: SQL+ 4.2 HARDWARE INTERFACES ▪ Windows.
* A browser which supports CGI, HTML & Javascript.
  1. SOFTWARE INTERFACES

Following are the software used for the flight management online application.

|  |  |
| --- | --- |
| Software used | Description |
| Operating system | We have chosen Windows operating system for its best support and user-friendliness. |
| Database | To save the flight records, passengers’ records we have chosen SQL+ database. |
| VB.Net | To implement the project we have chosen Vb.Net language for its more interactive support. |

* 1. COMMUNICATION INTERFACES

This project supports all types of web browsers. We are using simple electronic forms for the reservation forms, ticket booking etc.

### 5. NONFUNCTIONAL REQUIREMENTS

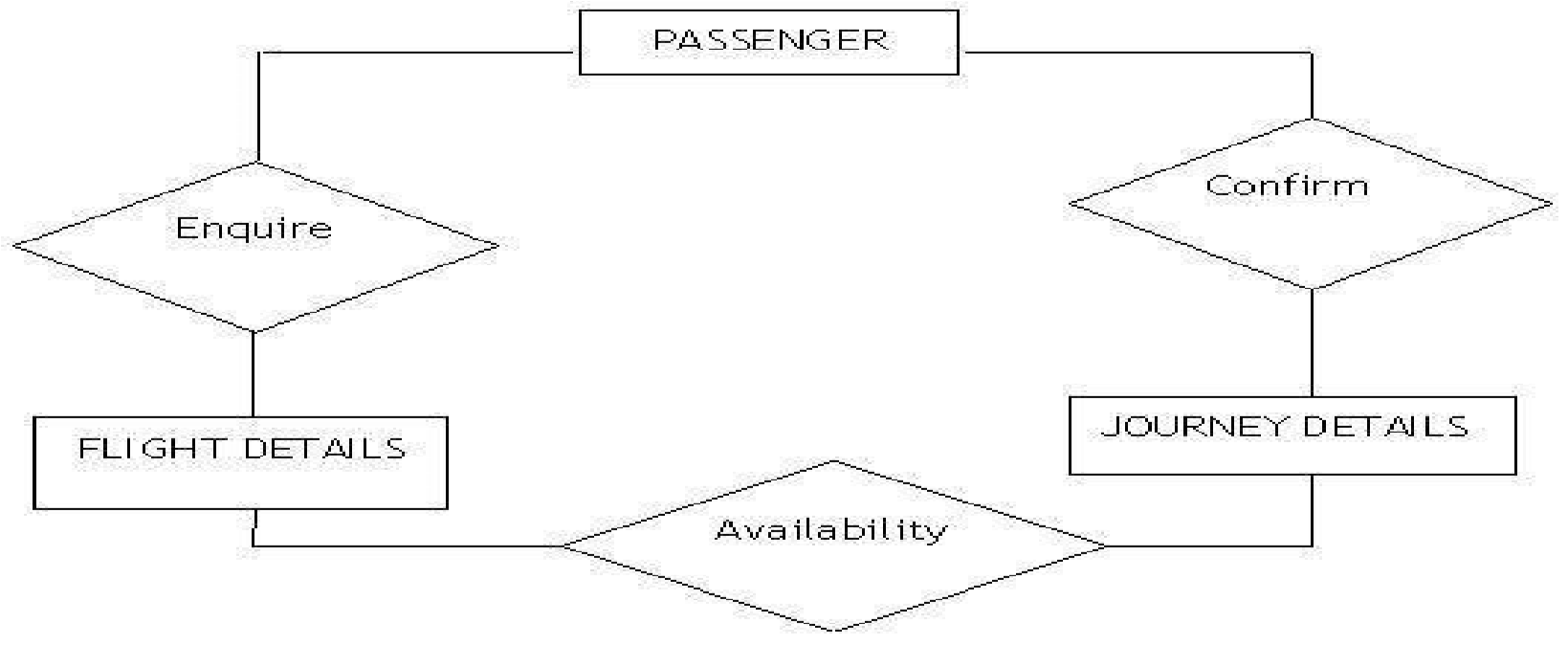
5.1 PERFORMANCE REQUIREMENTS

The steps involved to perform the implementation of airline database are as listed below.

A) E-R DIAGRAM

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

* ENTITIES: Which specify distinct real-world items in an application.
* PROPERTIES/ATTRIBUTES: Which specify properties of an entity and relationships.
* RELATIONSHIPS: Which connect entities and represent meaningful dependencies between them.

the diagram shows the ER diagram of airline database

B) NORMALIZATION:

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

5.2 SAFETY REQUIREMENTS

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

5.3 SECURITY REQUIREMENTS

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

5.4 SOFTWARE QUALITY ATTRIBUTES

* AVAILABILITY: The flight should be available on the specified date and specified time as many customers are doing advance reservations.
* CORRECTNESS: The flight should reach start from correct start terminal and should reach the correct destination.
* MAINTAINABILITY: The administrators and flight in chargers should maintain correct schedules of flights.
* USABILITY: The flight schedules should satisfy a maximum number of customers needs

**PROGRAM NO. 3**

**Objective:-**Draw a use case diagram of Bank Application Management.

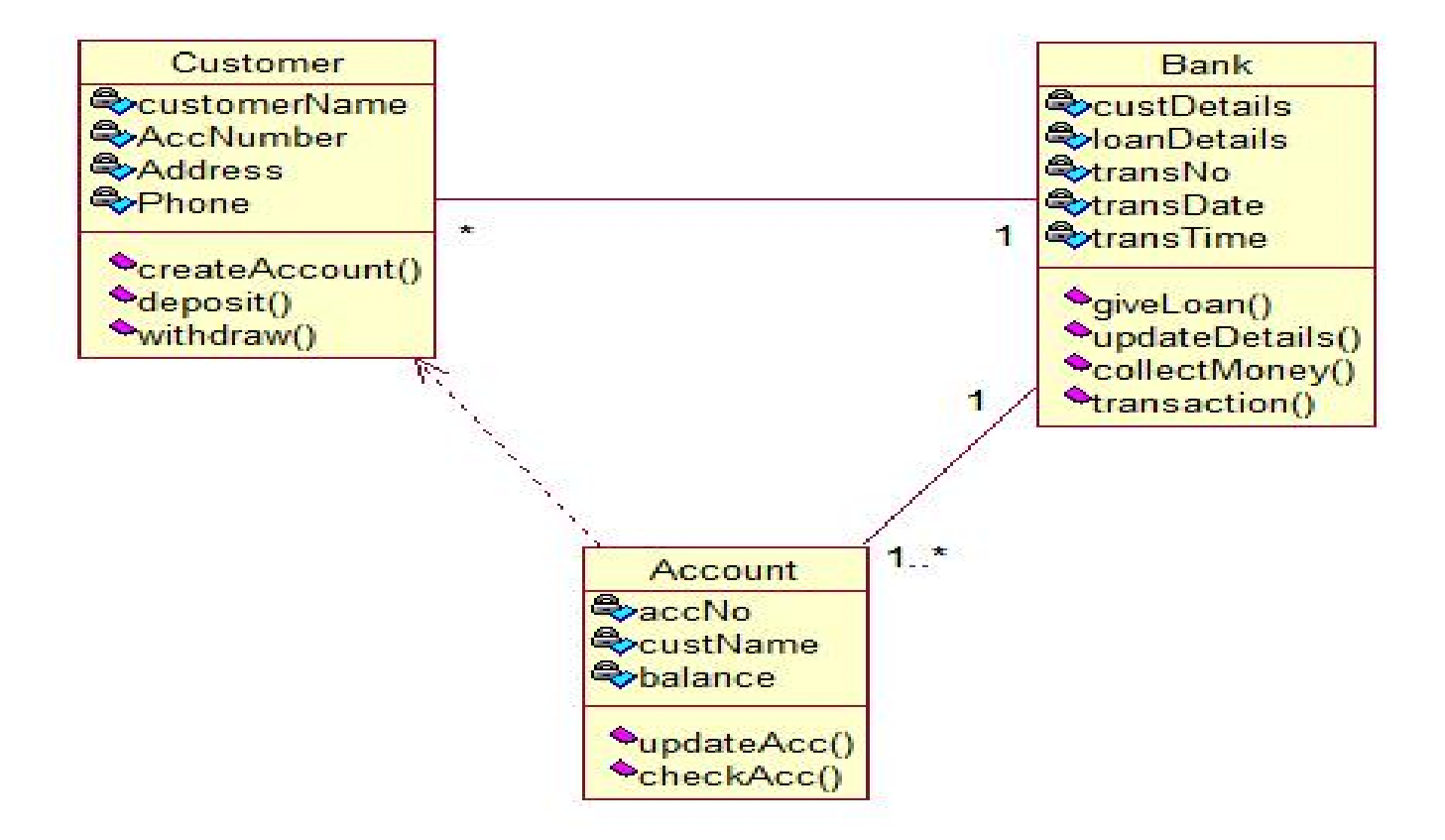
**Result:-**

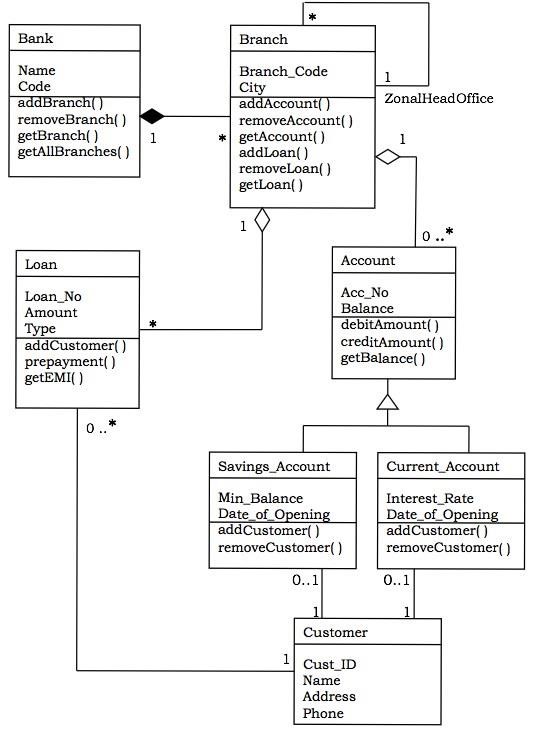


**PROGRAM NO. 4**

**Objective:-**Draw the class diagram and object model for banking Application System.

**Result:**-



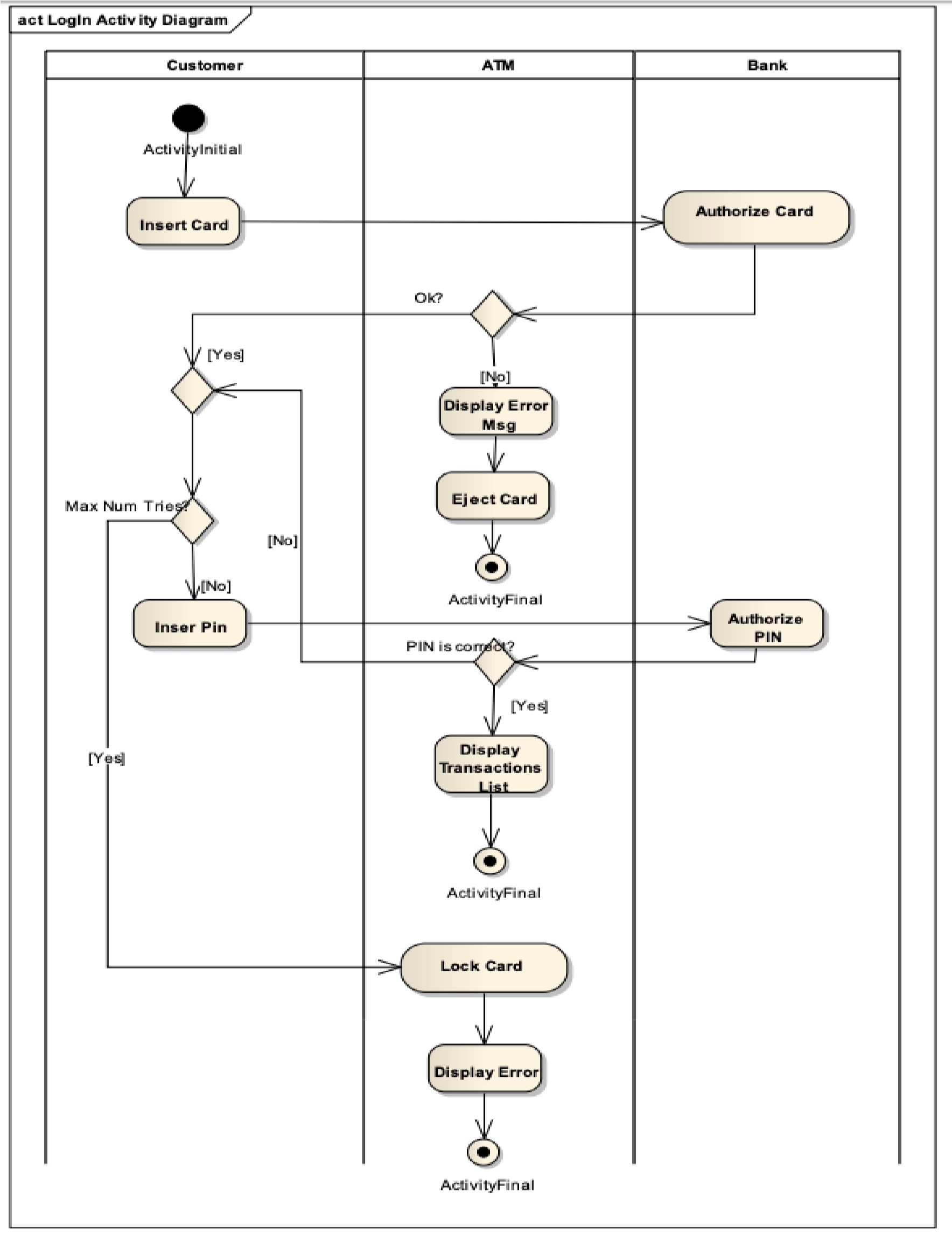


**PROGRAM NO. 5**

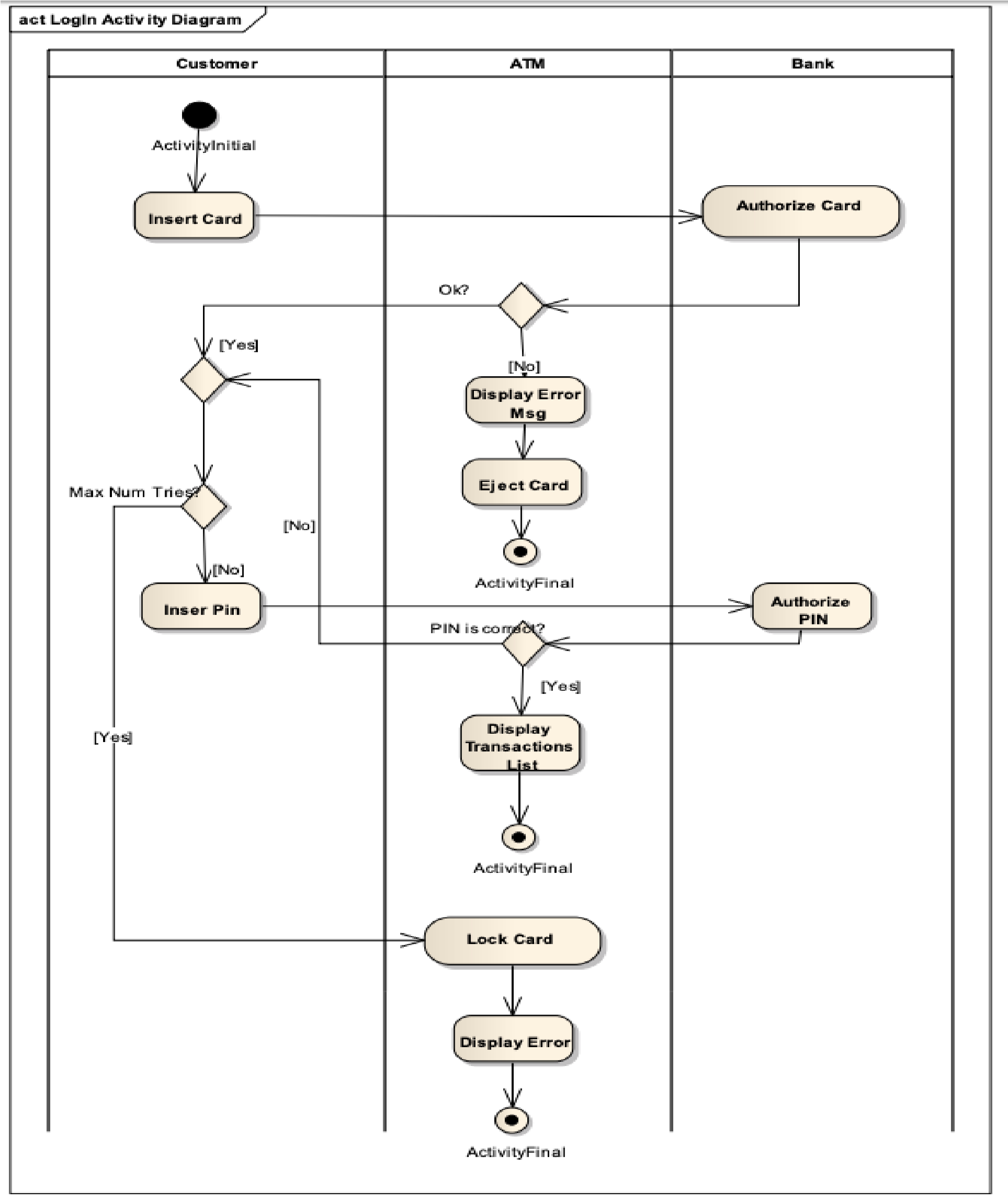
**Objective:**-Draw the activity diagram for ATM, discuss three activities for the same.

**Result:-**

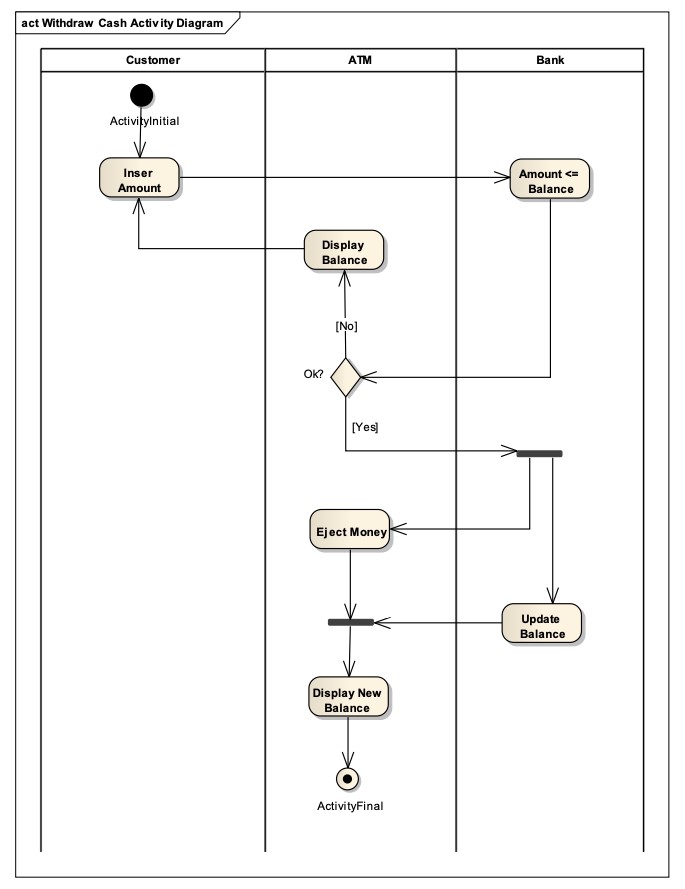
Activity1:



Activity 2:



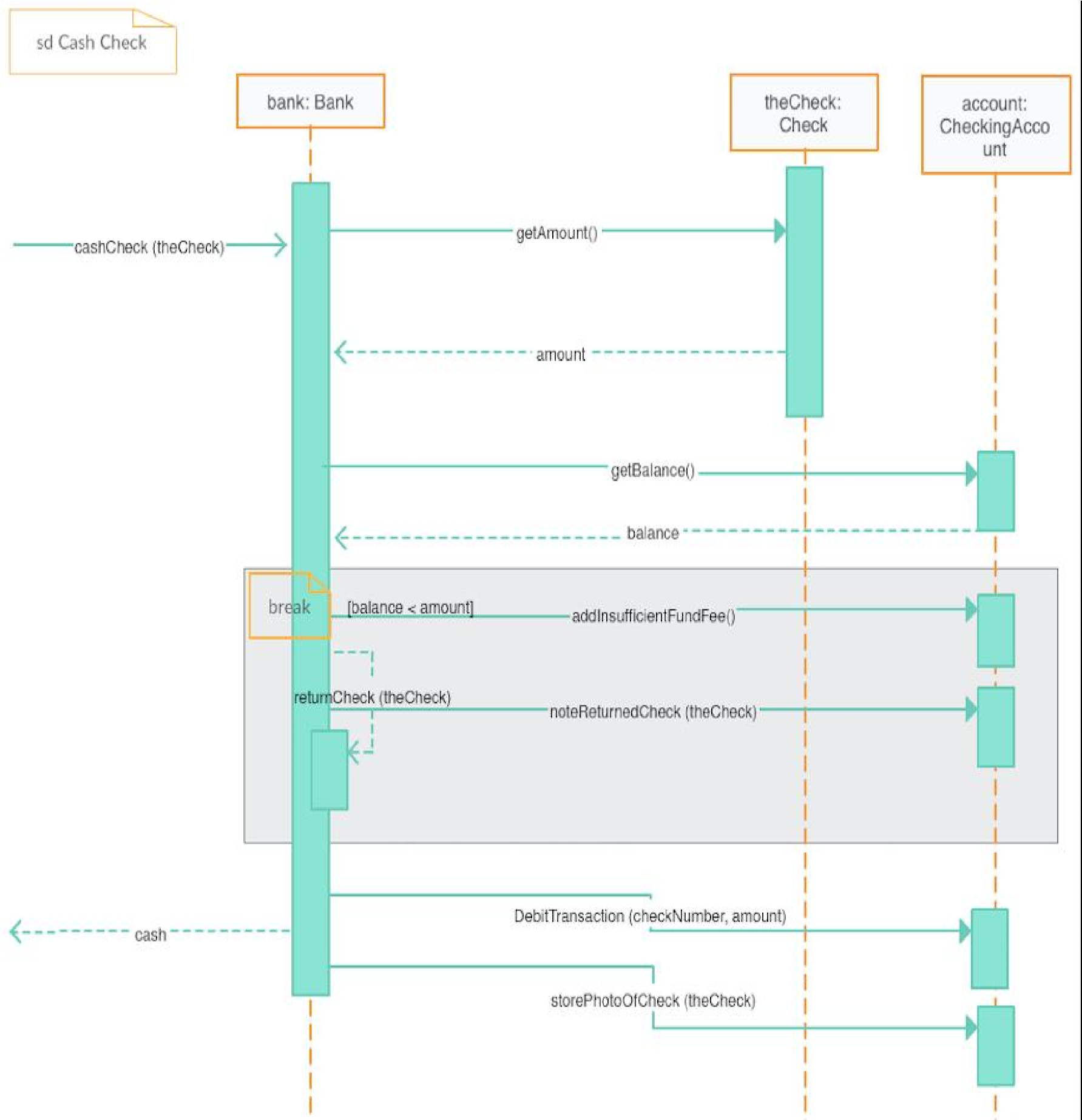
Activity 3:



**PROGRAM NO. 6**

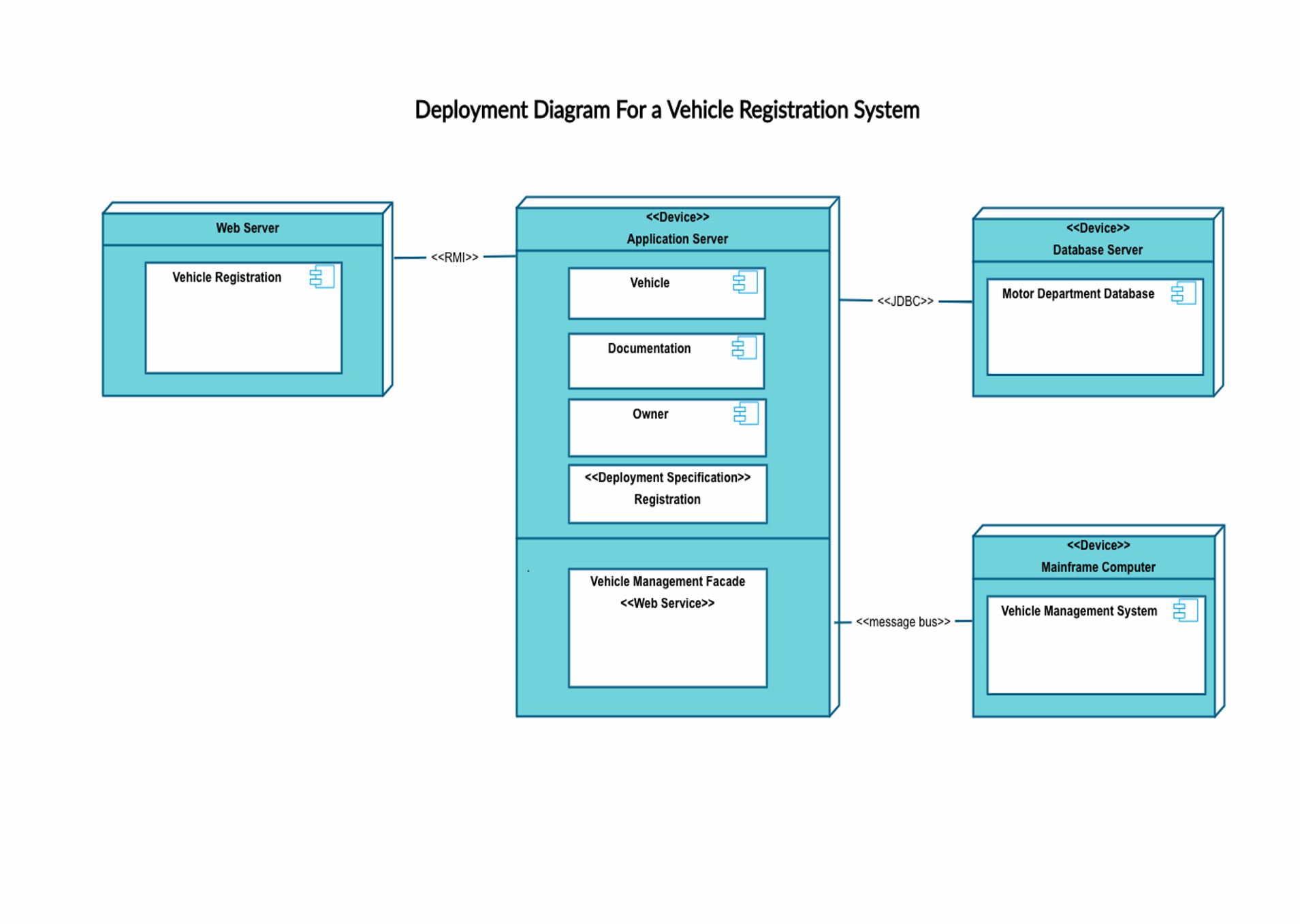
**Objective**:-Draw the sequence diagram for Bank Application System

## Result:-



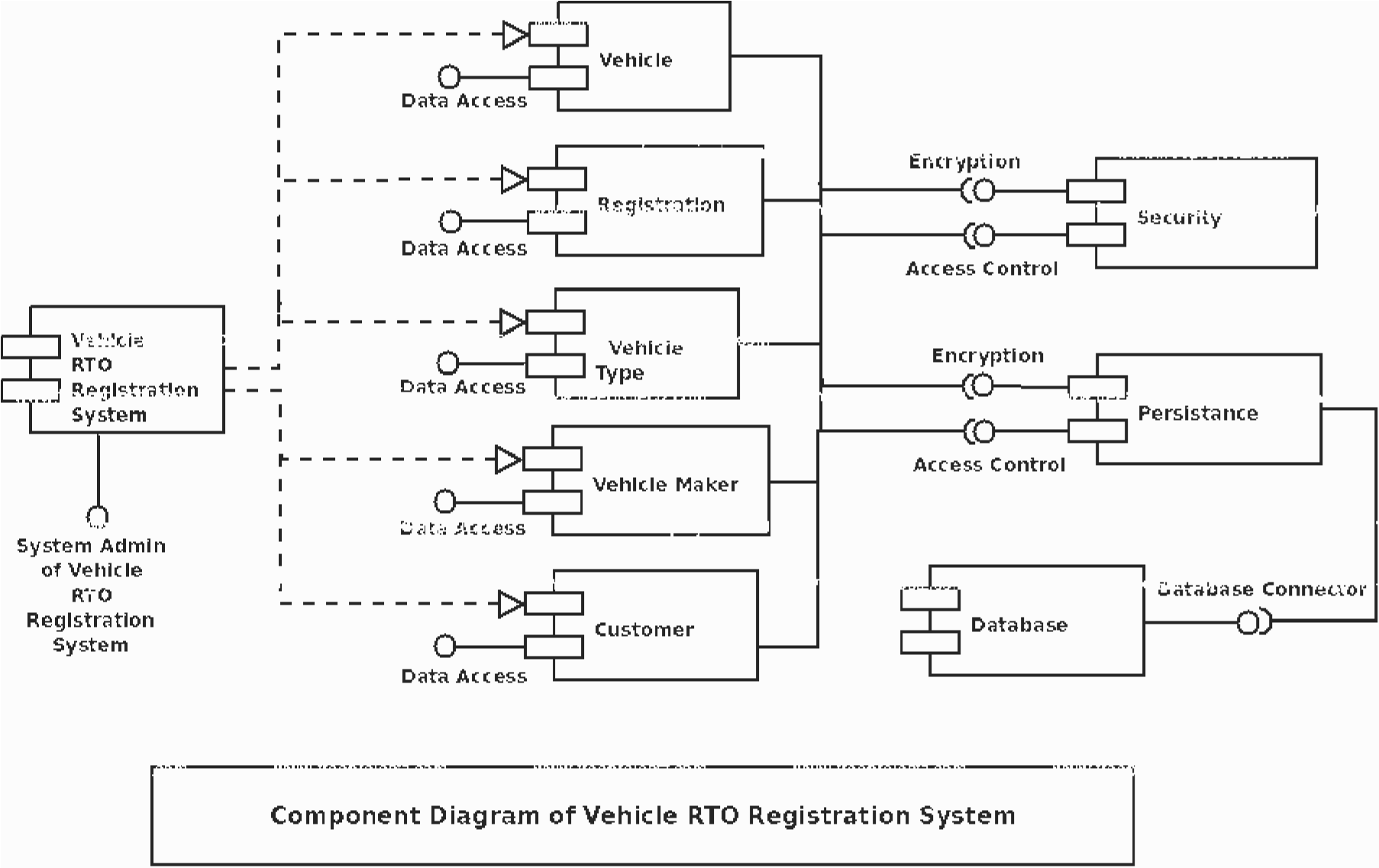
**PROGRAM NO.7**

Draw the Deployment diagram for the Vehicle Registration System.



**PROGRAM NO. 8**

**Objective:** Draw a component diagram for vehicle registration system.



**PROGRAM NO. 9**

**Objective:** Write a program to calculate function point analysis.

#include<stdio.h> #include<conio.h> int fround(float x)

{

int a; x=x+0.5; a=x; return(a);

}

void main()

{

int weights[5][3]={3,4,6,4,5,6,3,4,6,7,10,15,5,7,10}; int UFP=0,F=0,rating,i,j;

char func\_units[][30]={"External Inputs","External Outputs","External Enquiries","Internal Logical Files","External Interface Files"}; char complexity[3][10]={"low","average","high"}; int input[5][3];

float FP,CAF; clrscr(); for(i=0;i<5;i++)

{

for(j=0;j<3;j++)

{

printf("\nEnter number of %s %s : ",complexity[j],func\_units[i]); scanf("%d",&input[i][j]);

}

}

//calculating UFP for(i=0;i<5;i++)

{

for(j=0;j<3;j++)

{

UFP=UFP+(input[i][j]\*weights[i][j]);

}

}

printf("\nUnadjusted Function Point(UFP) = %d",UFP); printf("\nEnter Rating of 14 factors on the scale of 0-5 :\n"); printf("\n 0 - No Influence"); printf("\n 1 - Incidental");

printf("\n 2 - Moderate"); printf("\n 3 - Average"); printf("\n 4 - Significant"); printf("\n 5 - Essential"); printf("\n"); for(i=0;i<14;i++)

{

scanf("%d",&rating);

F=F+rating;

}

CAF=0.65+0.01\*F; FP=UFP\*CAF;

printf("\nAdjusted Function Point = %f",FP); printf("\nOR FP = %d",fround(FP));

getch(); }

**PROGRAM NO. 10**

**Objective**:- Draw an use case diagram for amizone.

### Result:-

Student

Faculty

Registration

Admission

Fees

APW

Attendance

Examination

Documents

Time Table

Marks

Leave

Salary

**PROGRAM NO. 11**

**Objective**:- Draw an use case diagram for bank application.

### Result:-

User

Bank Manager

Login

Open account

Close Account

Loan

interest

Fixed Deposit

Deposit

Withdrawal

Feedback

Account maintainance

Documentation

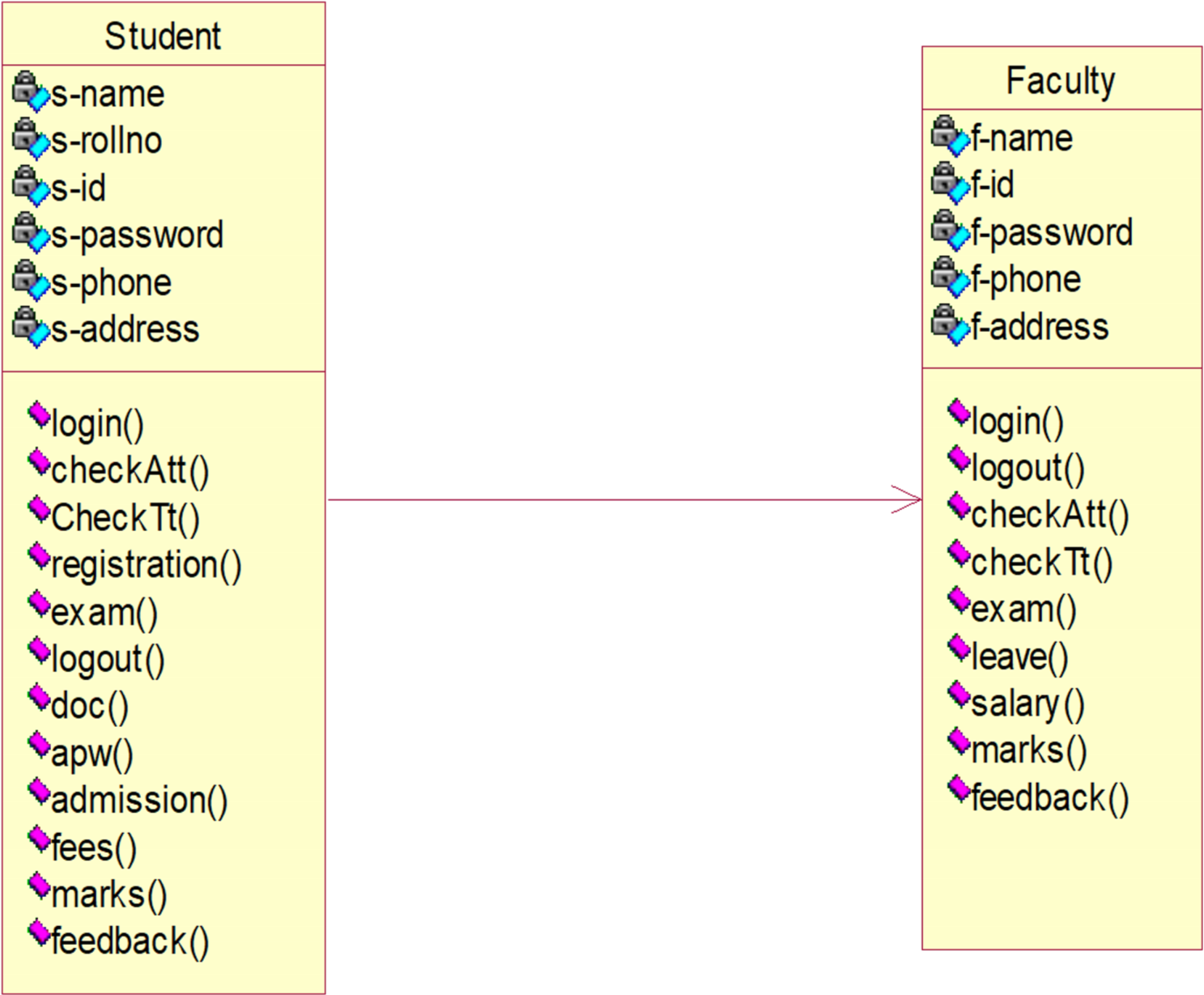
Balance

Switch account

**PROGRAM NO. 12**

**Objective-**Draw a class diagram for amizone.

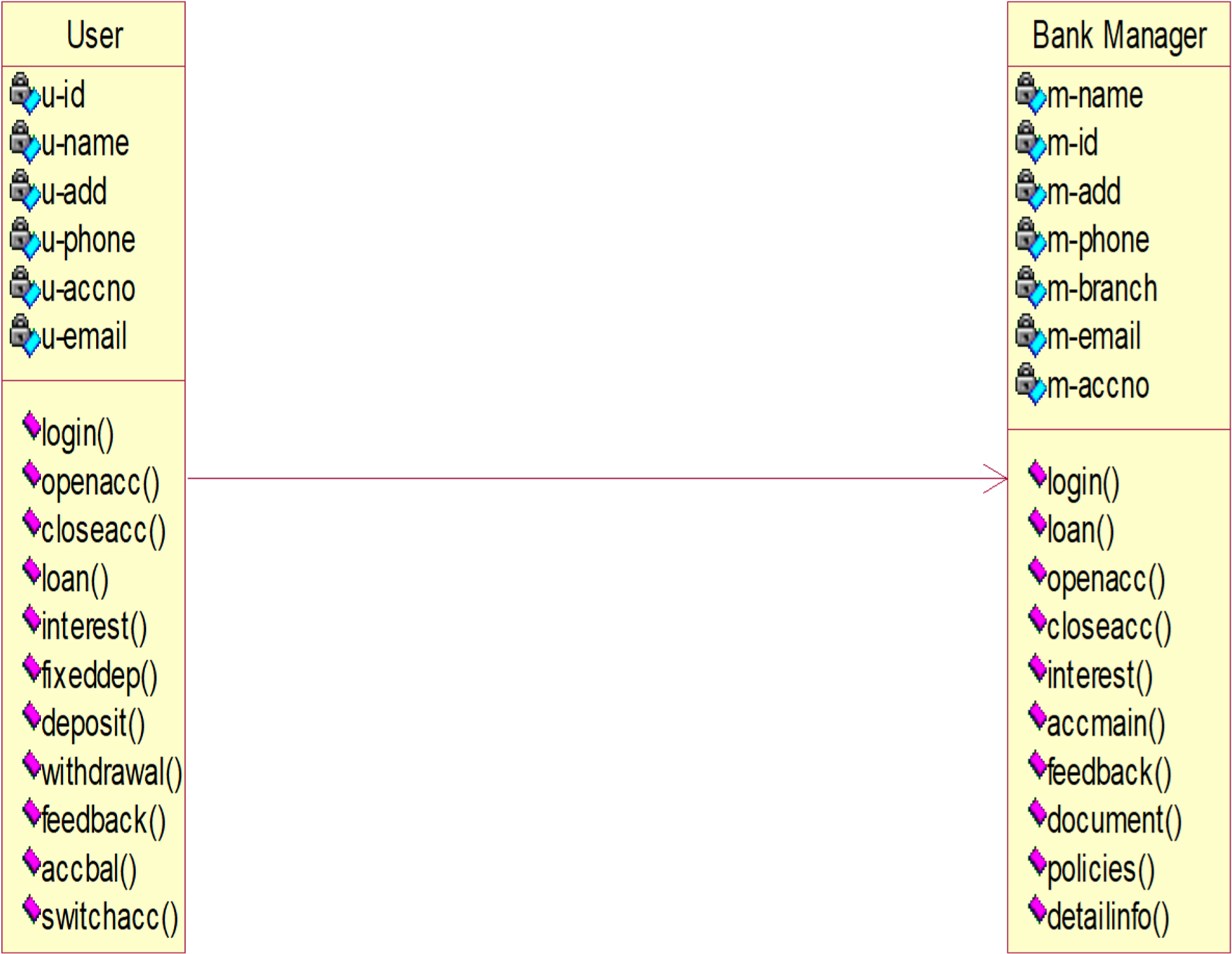
### Result:-



**PROGRAM NO. 13**

**Objective:** Draw a class diagram for bank application.

### Result:-



**PROGRAM NO. 14**

**Objective:** Draw an object diagram for amizone.

### Result:

|  |
| --- |
| T: Teacher |

A

1:

A

ttendance

Number=12

A

2

:

A

ttendance

Number=32

Number=40

A

3

:

A

ttendance

C

1:

Check

U

2:

Update

V3:View

Number=20

Number=30

Number=76

**PROGRAM NO. 15**

**Objective**:- Draw an object diagram for bank application.

### Result:-

|  |
| --- |
| C: Customer |

Ac1: Account

Number=12

Ac2:

Account

Number=32

Number=40

Ac3: Account

S: Saving Account

Cn: Current Account

N: New Account

Number=20

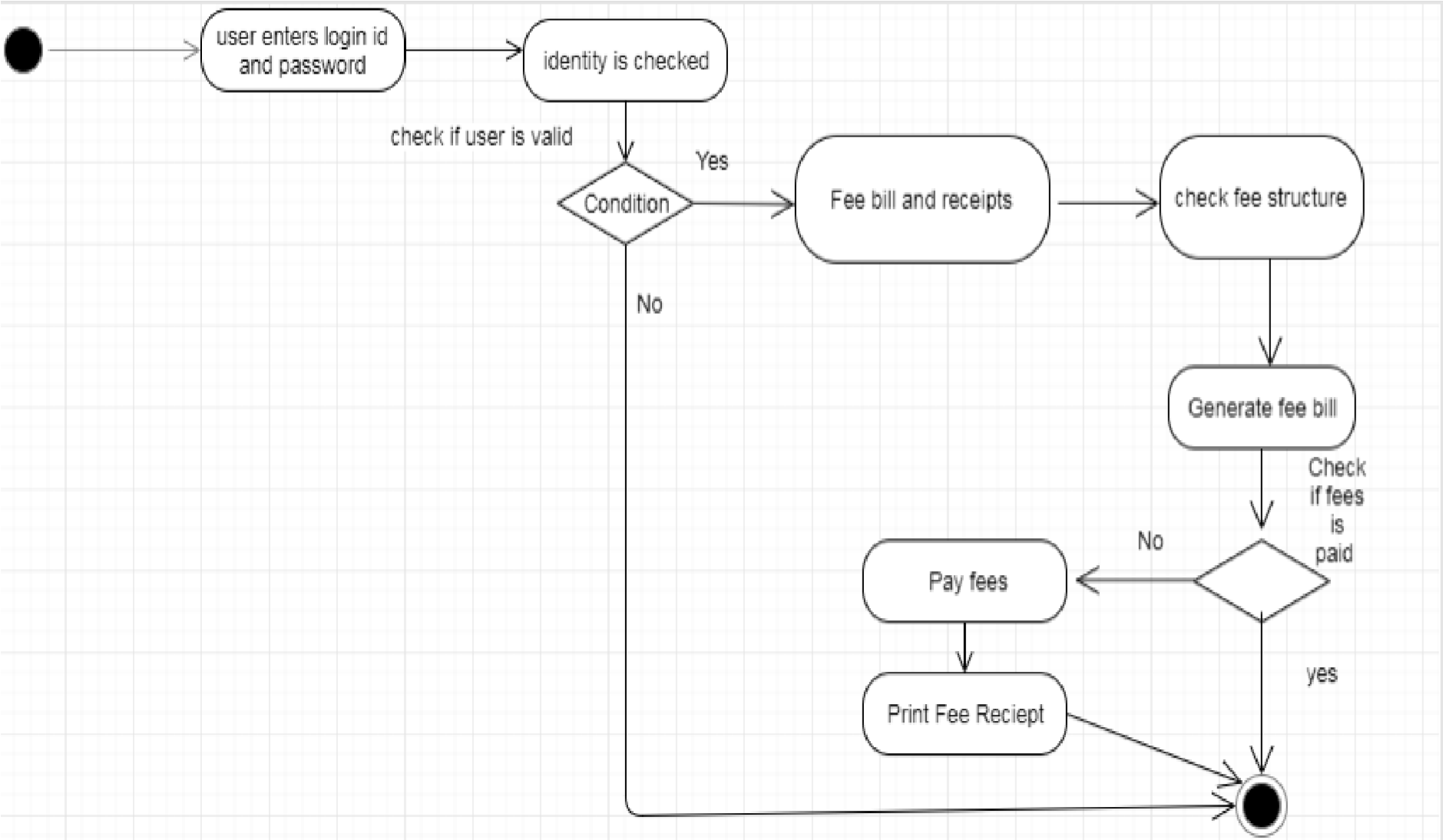
Number=30

Number=76

**PROGRAM NO. 16**

**Objective:-** Draw an activity diagram for amizone**.**

### Result:-



**PROGRAM NO. 17**

**Objective**:- Draw an activity diagram for bank application. **Result:-**

