Assignment -1

1) Develop SRS.

(a) Search Book Availability in Library.

R-1 : Search Book Availability.

Description : This function determines which book the user want, and from which genre the book belongs to. It checks whether the genre or book is available or not. If available it will return the book as output. Otherwise it generates an error.

R.1.1 : select genre.

Input : select genre abc / def / ghi

Output : Prompt to select book.

R.1.2 : Select book

Input : select book from any of the following. mno / pqr / stu or anyother can be written.

Output : The required book availability will be given.

Processing : The required book availability will be printed if present, otherwise an error message will be displayed.

(b) Withdraw cash from ATM

R 1 : Withdraw cash.

Description : The function first determines the type of account that the user has and the account number from which the user wishes to withdraw the cash. It checks the balance to determine whether the requested amount is

available in the account. If enough balance is available in the account, it outputs required cash.
otherwise it generates an error message.

R.1.1 : Select withdraw amount option.

Input : "Withdraw amount" option selected.

Output : User prompted to enter the account type.

R.1.2 : Select account type.

Input : "~~withdraw~~ User selects option from any of the following savings / checking / deposit

Output : Prompt to enter amount.

R.1.3 : Get required amount.

Input : Amt to be withdrawn in integer values

Output : The requested cash and printed transaction statement.

(2) Develop the functional and non-functional requirement for the personal library software.

Functional requirement:

The librarian does the following function -

(i) Add Article — New article must be add in database.

(ii) Update Article — Any change in article must be updated.

(iii) Enquiry members — view all the details of current members.

(iv) Checkout article — To issue any article must be checked out.

(v) Check in article — After requiring any article, system will enter by checking.

(vi) Enquiry waiting for approval.

(vii) Reverse article — This is to reverse any book application which is in waiting.

→ member does the following function :-

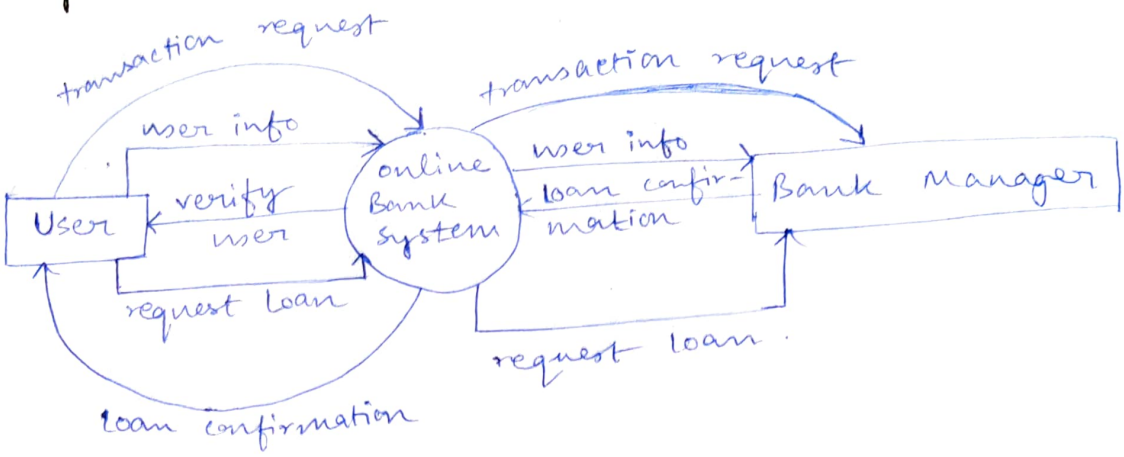
(i) Authetication — users must be authenticated before accessing system.

(ii) search article — User can search any article.

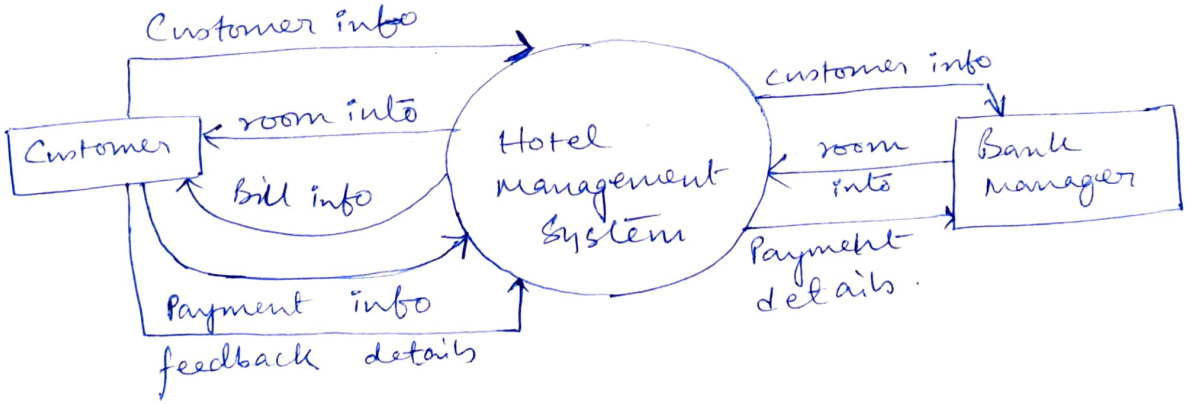
(iii) Request article — After succesful searching make the book marked.

(iv) Check account – The use case is used to check the account details.

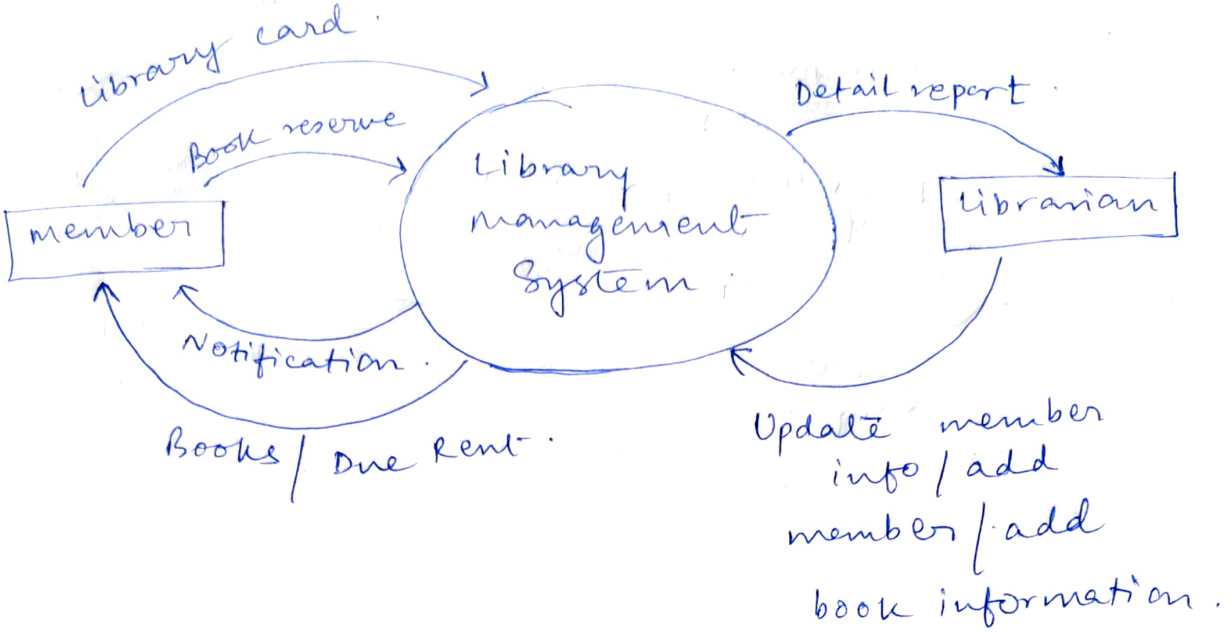
(3) Develop Data Flow Diagram for Online Banking System —



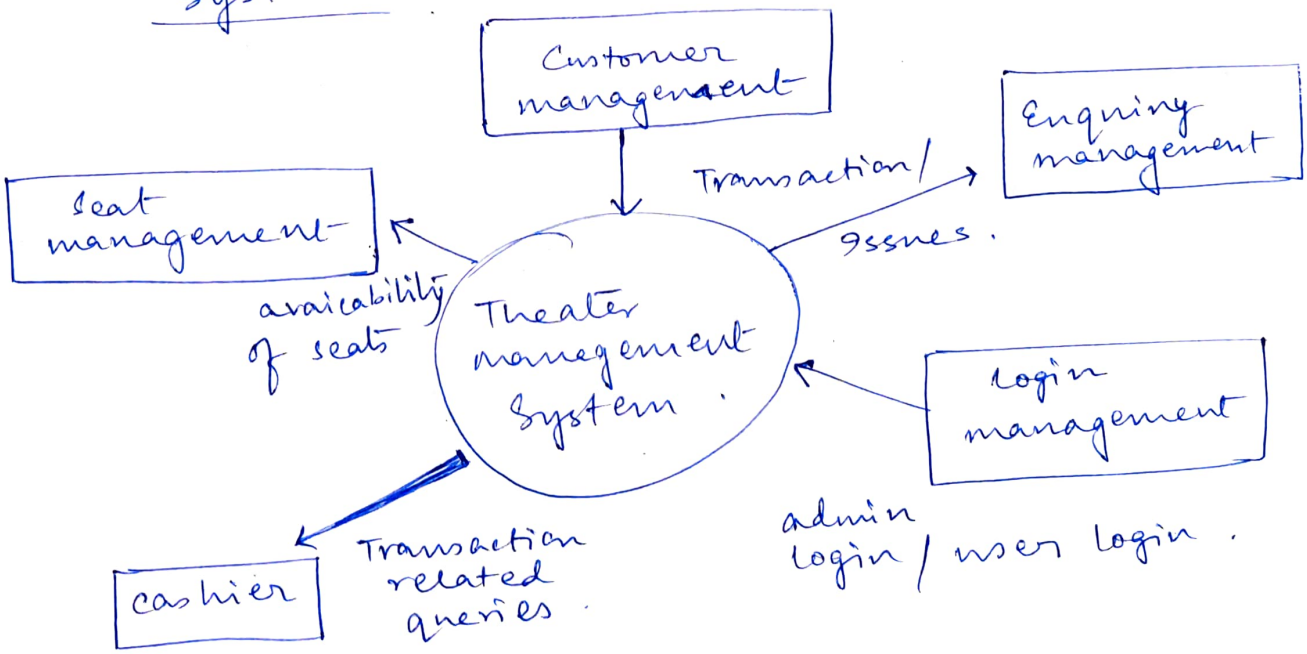
(5) Develop DFD for Hotel Management System.



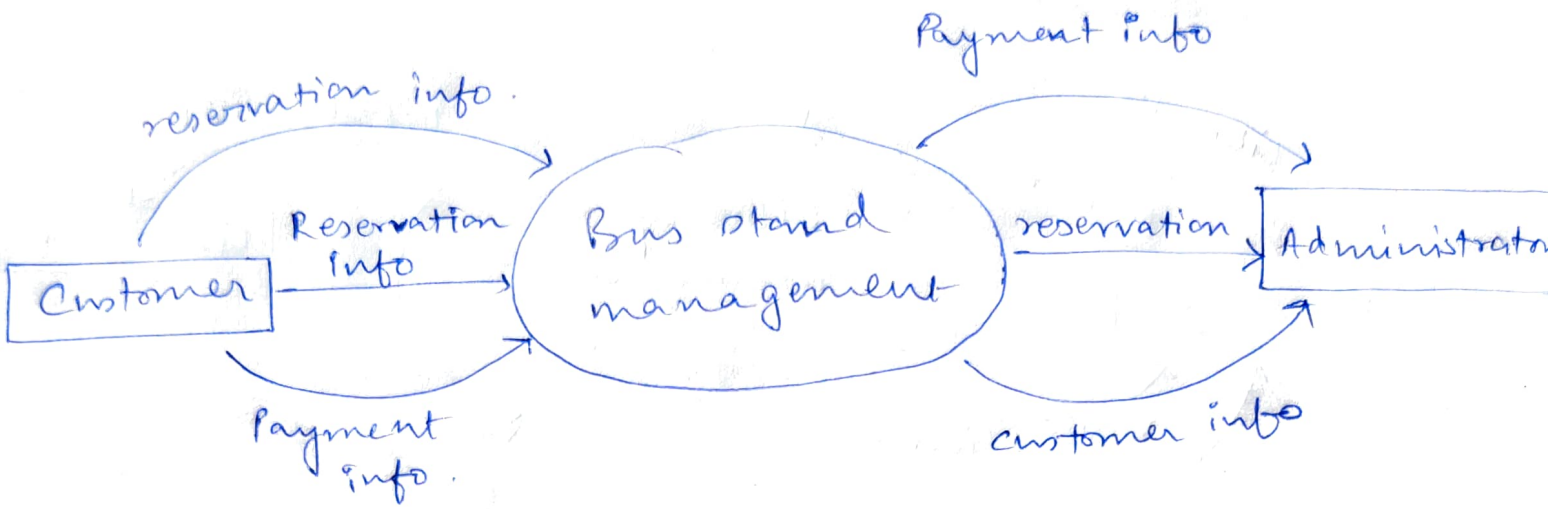
(4) Develop DFD for Library management System (Level 0).



(6) Develop DFD for Theater management system



(7) Develop DFD for Bus stand management System (Level - 0).



ASSIGNMENT : 2

① Develop data flow diagram for Online Banking System.

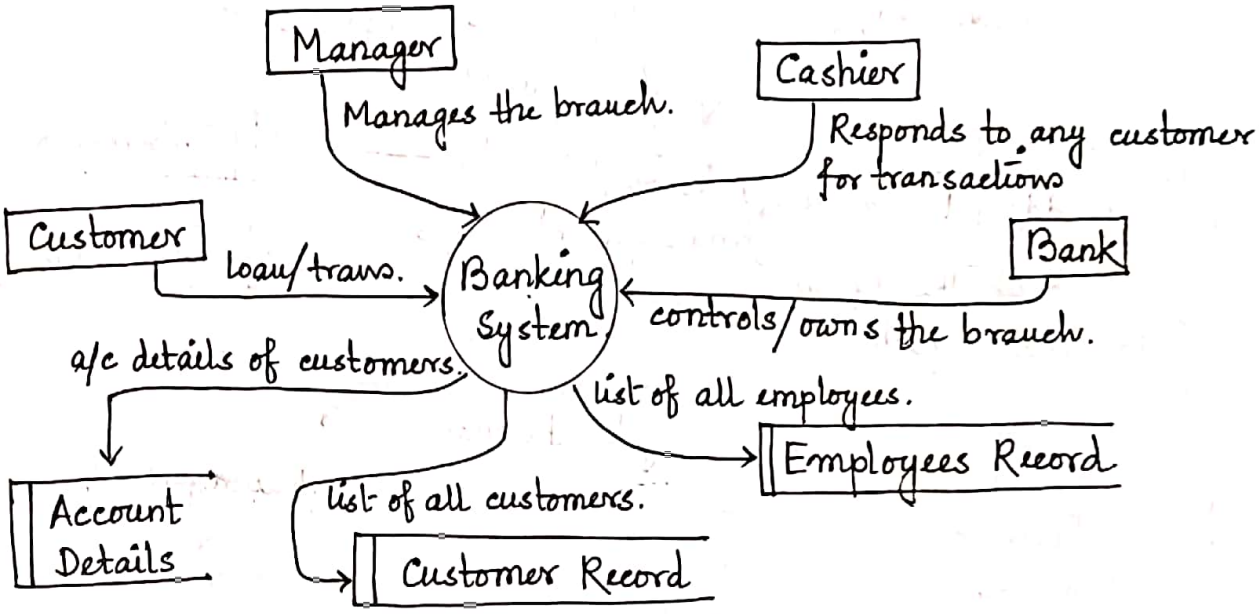


Fig 1 : Level-0 DFD for Online Banking System.

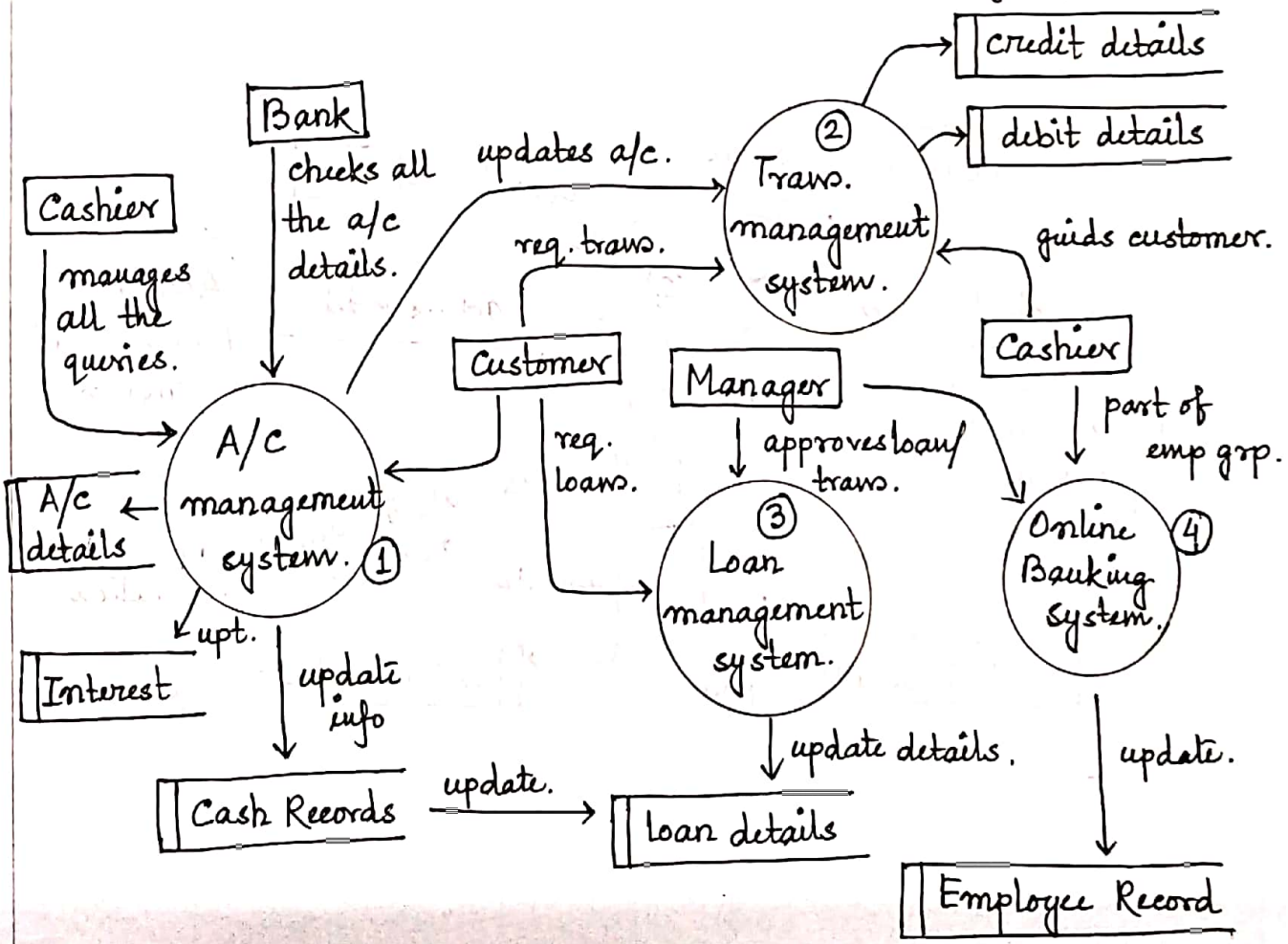
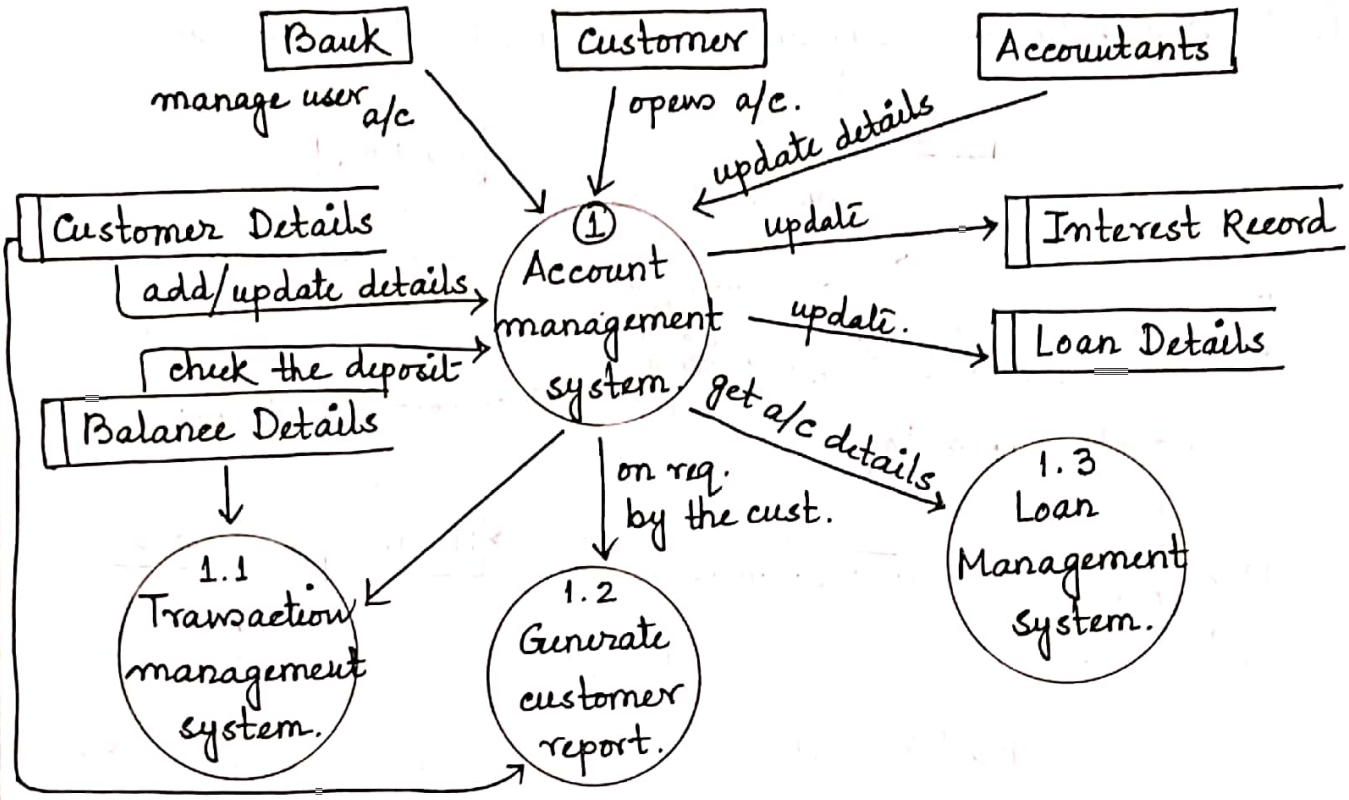
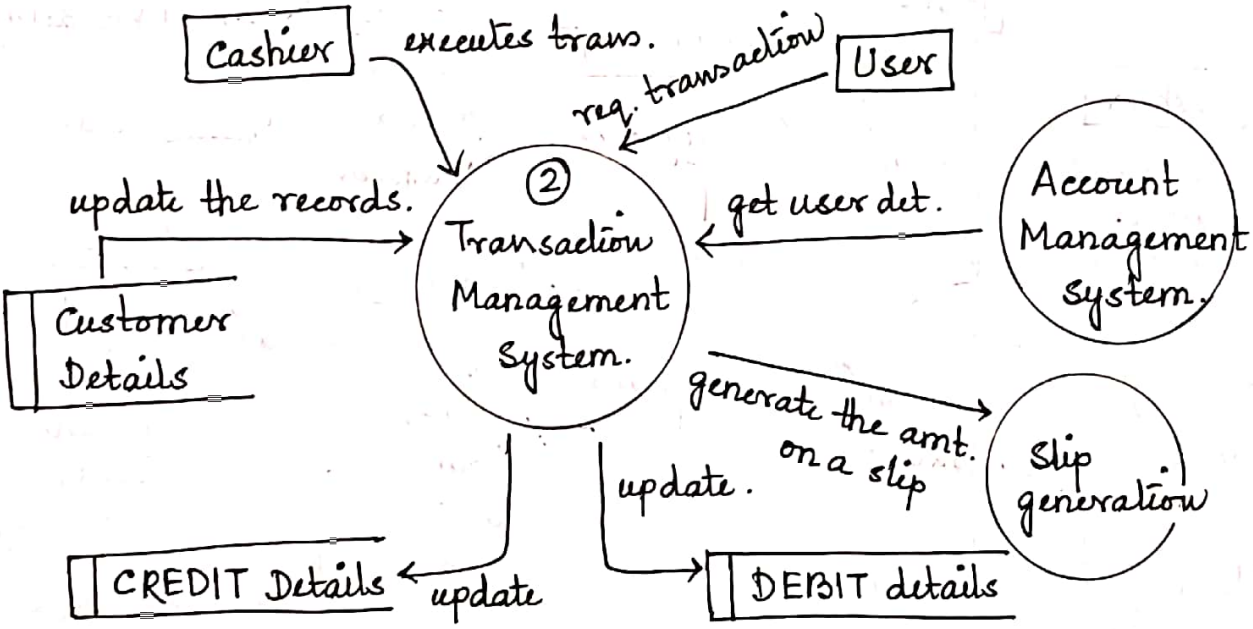
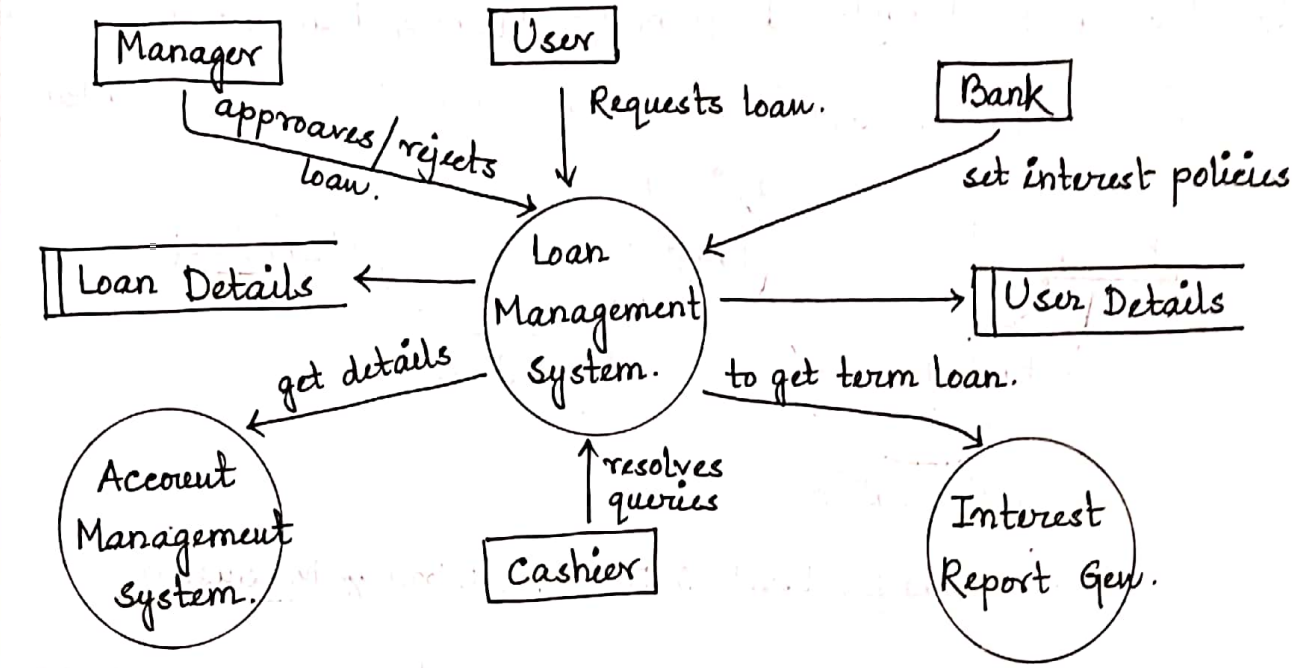


Fig 2 : Level-1 DFD for Online Banking System.







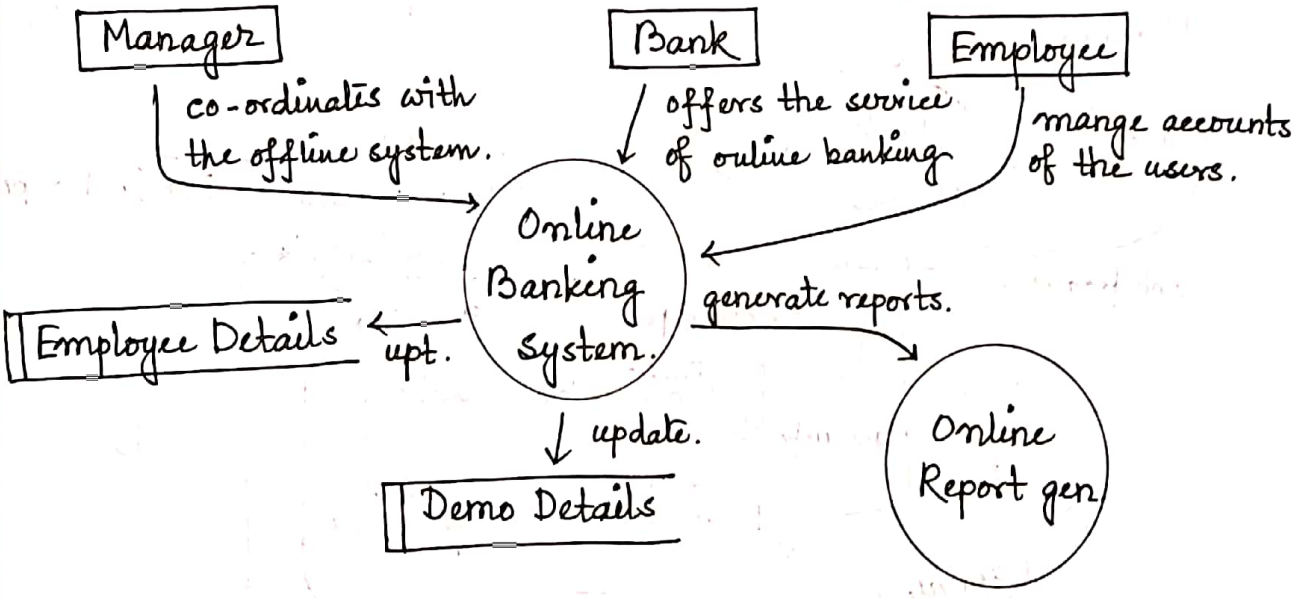


Fig 3 : Level-2 DFD of Online Banking System.

② Develop the dataflow diagram for Library Management System.

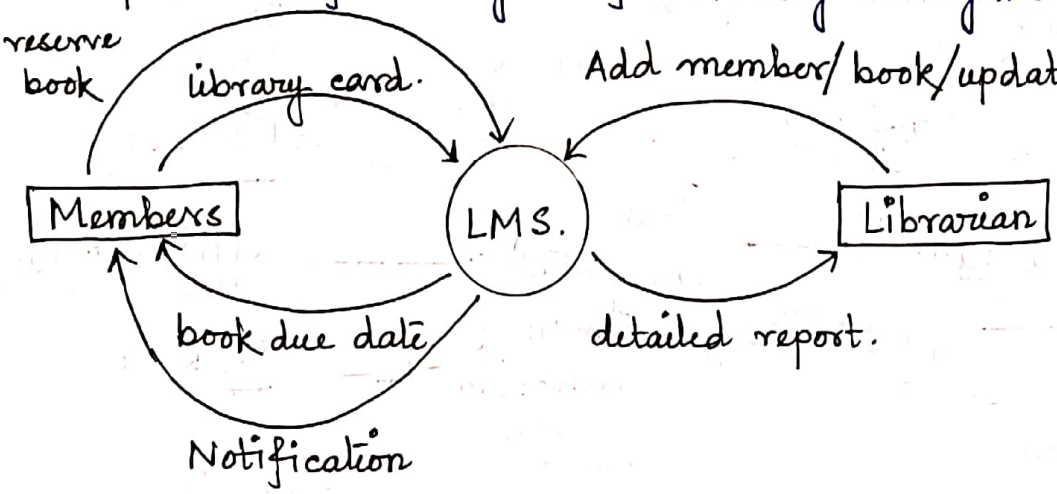


Fig 1 : Level-0 DFD for Library M. System.

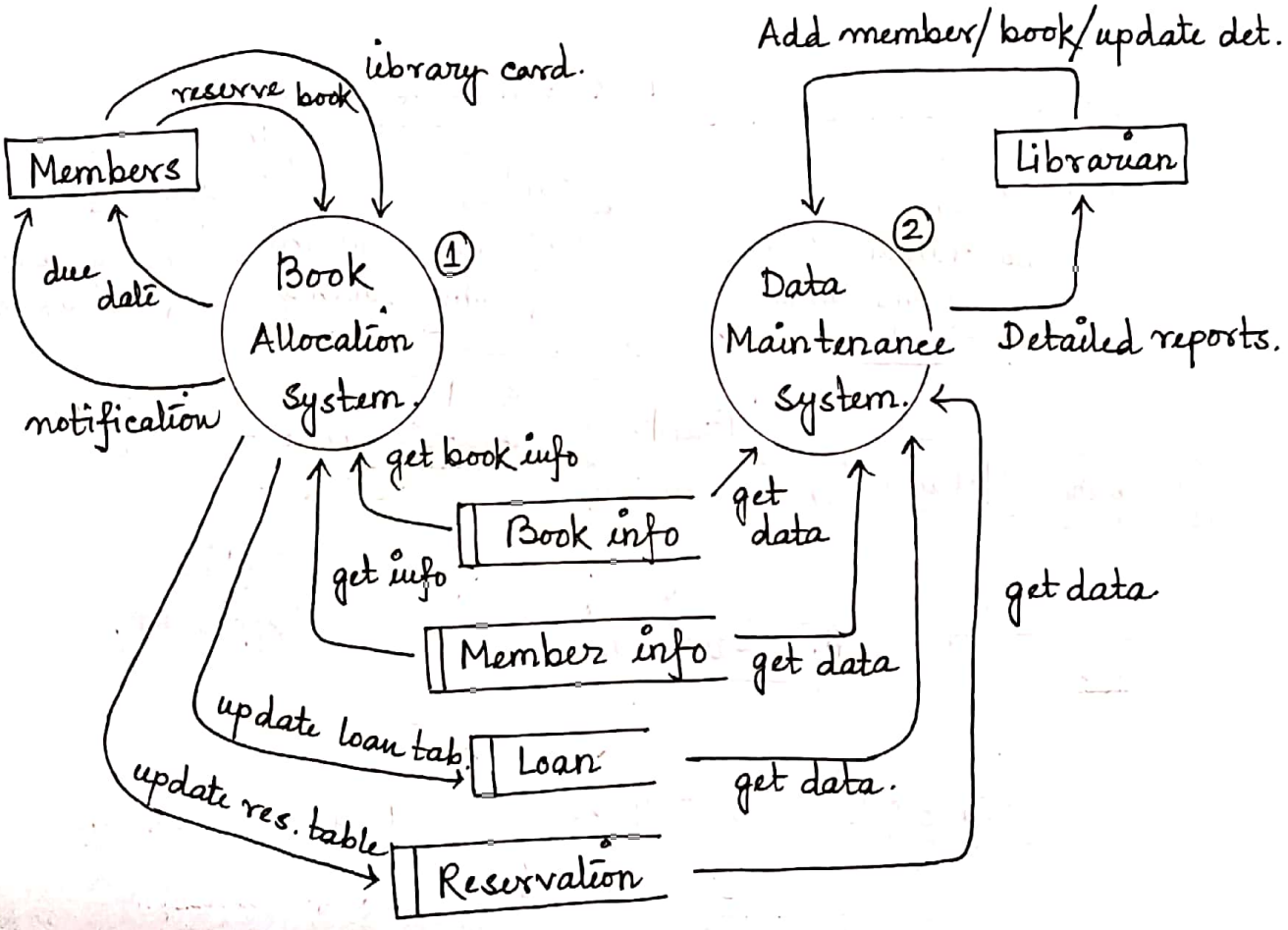
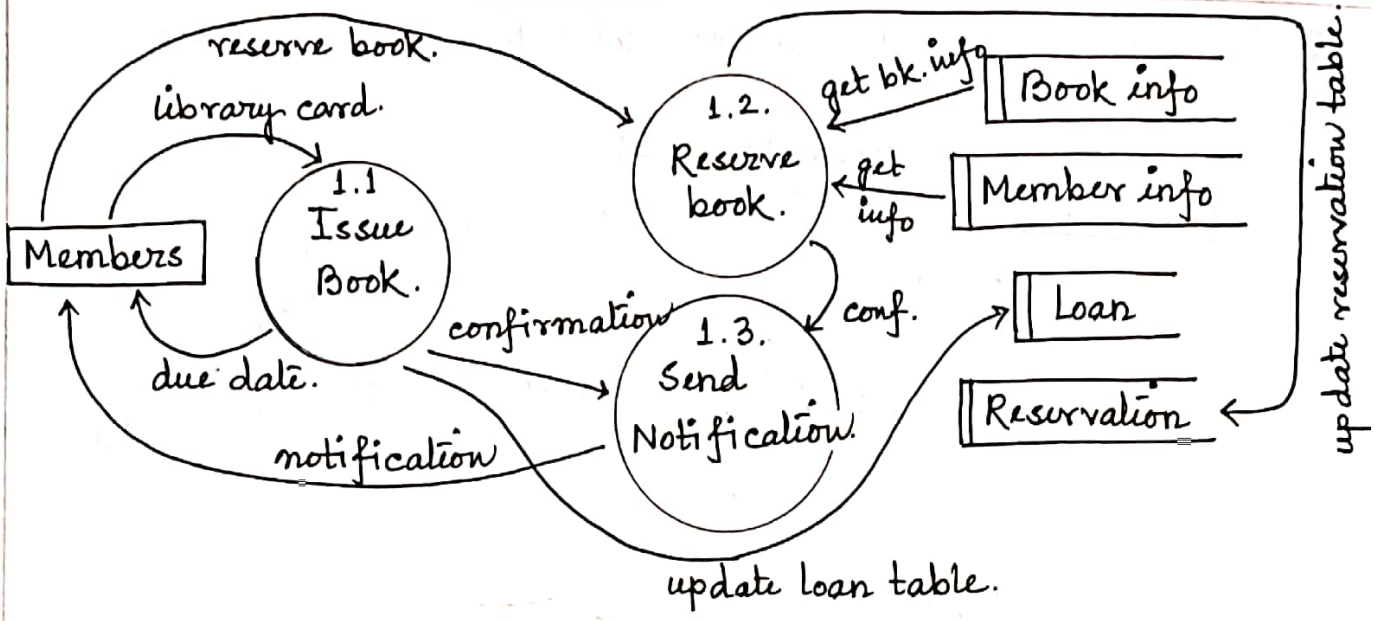


Fig 2 : Level-1 DFD for Library M. System.



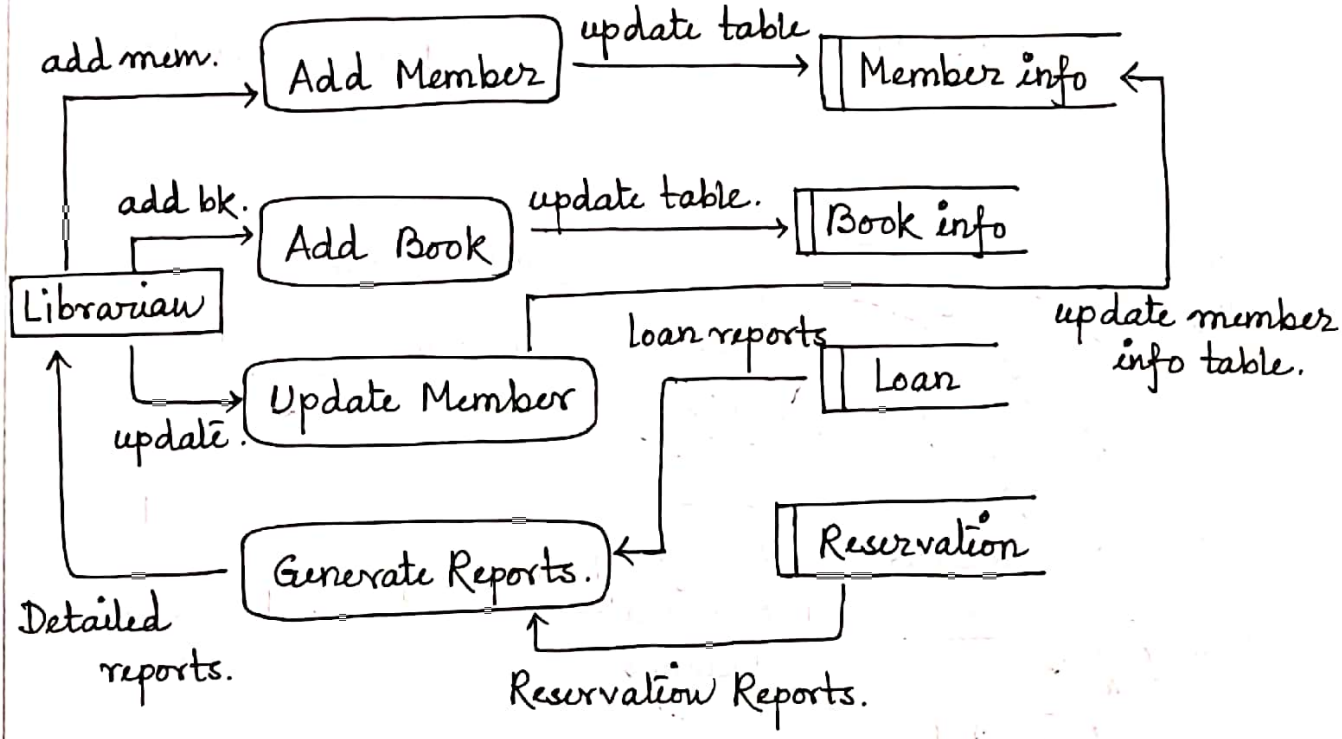


Fig 3: Level-2 DFD for Library Management System.

(3) Develop Dataflow diagram for Hotel Management System.

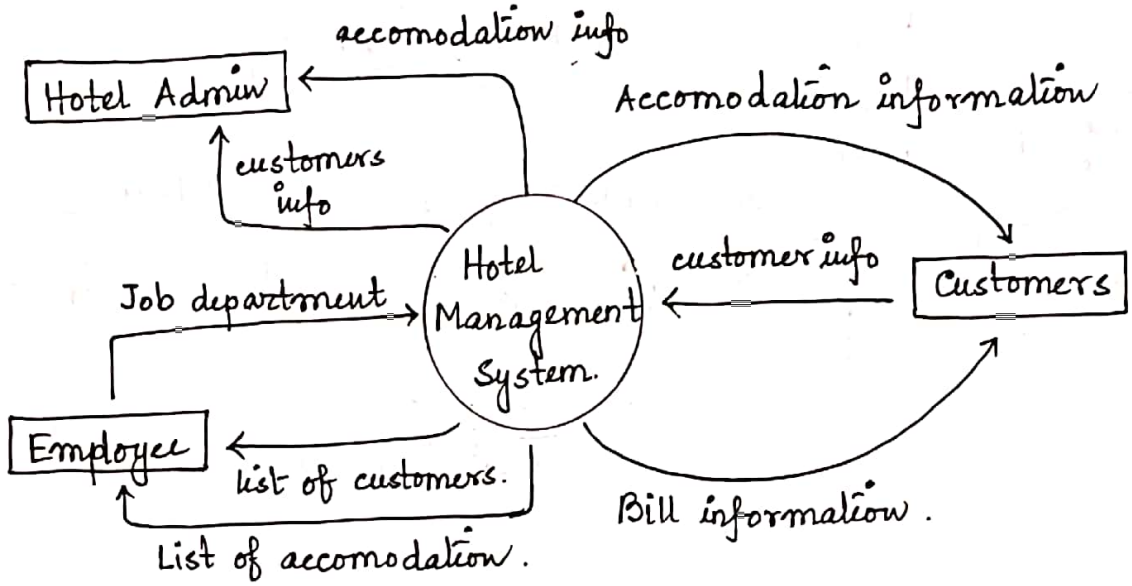


Fig 1: Level-0 DFD for Hotel Management System.

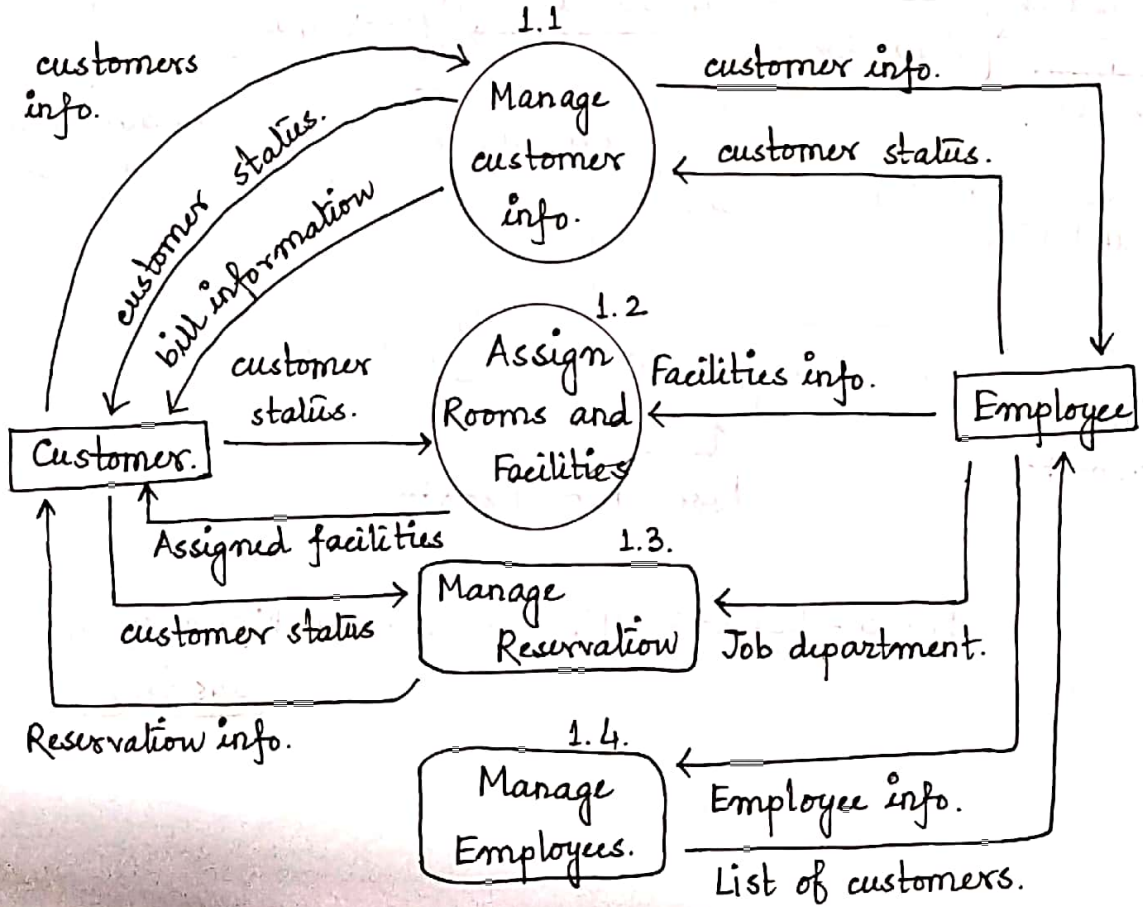


Fig 2: Level-1 DFD for Hotel Management System.

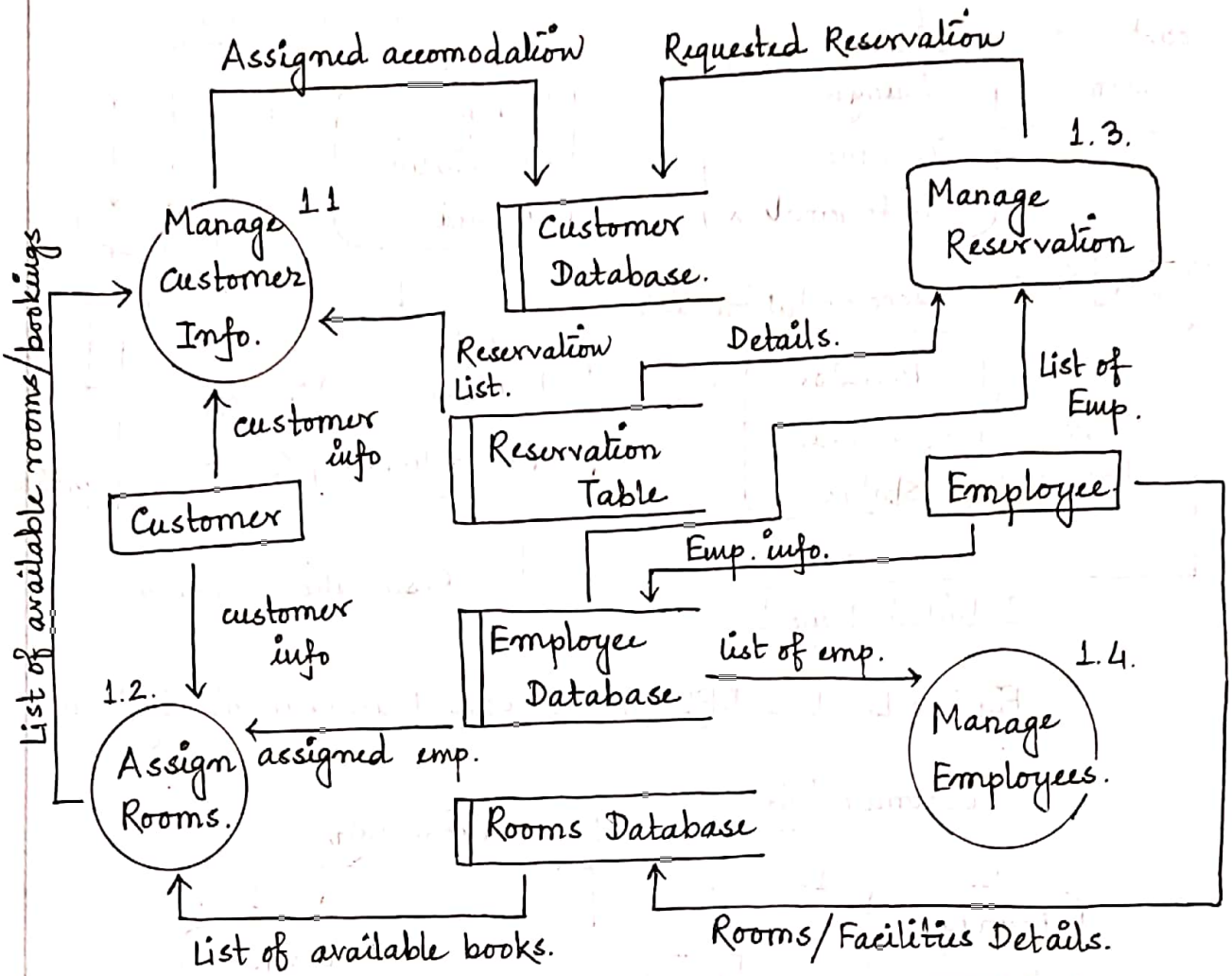


Fig 3 : Level-2 DFD for Hotel Management System.

④ Develop Dataflow diagram for (DFD) Airport Management System.

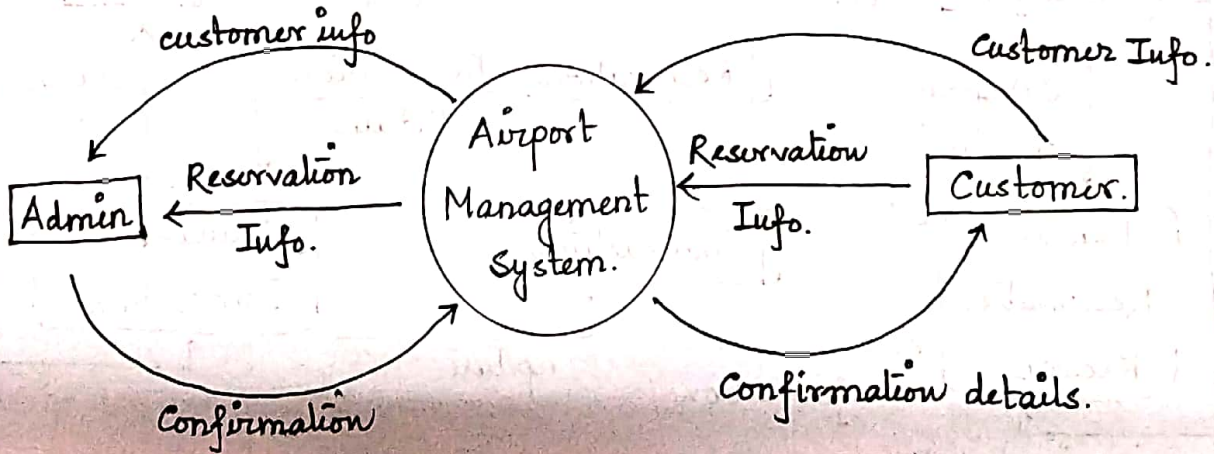


Fig 1 : Level-0 DFD for Airport Management System.

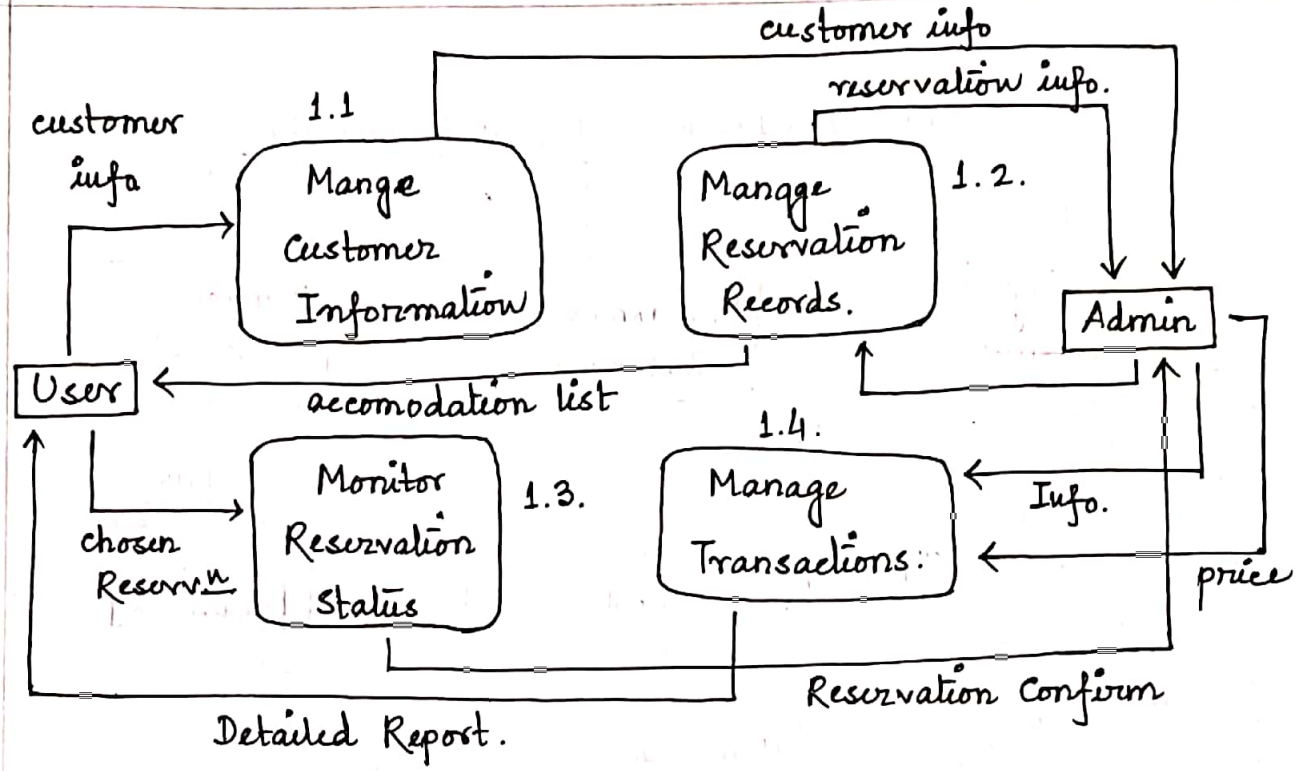


Fig 2: Level-1 DFD for Airport Management System.

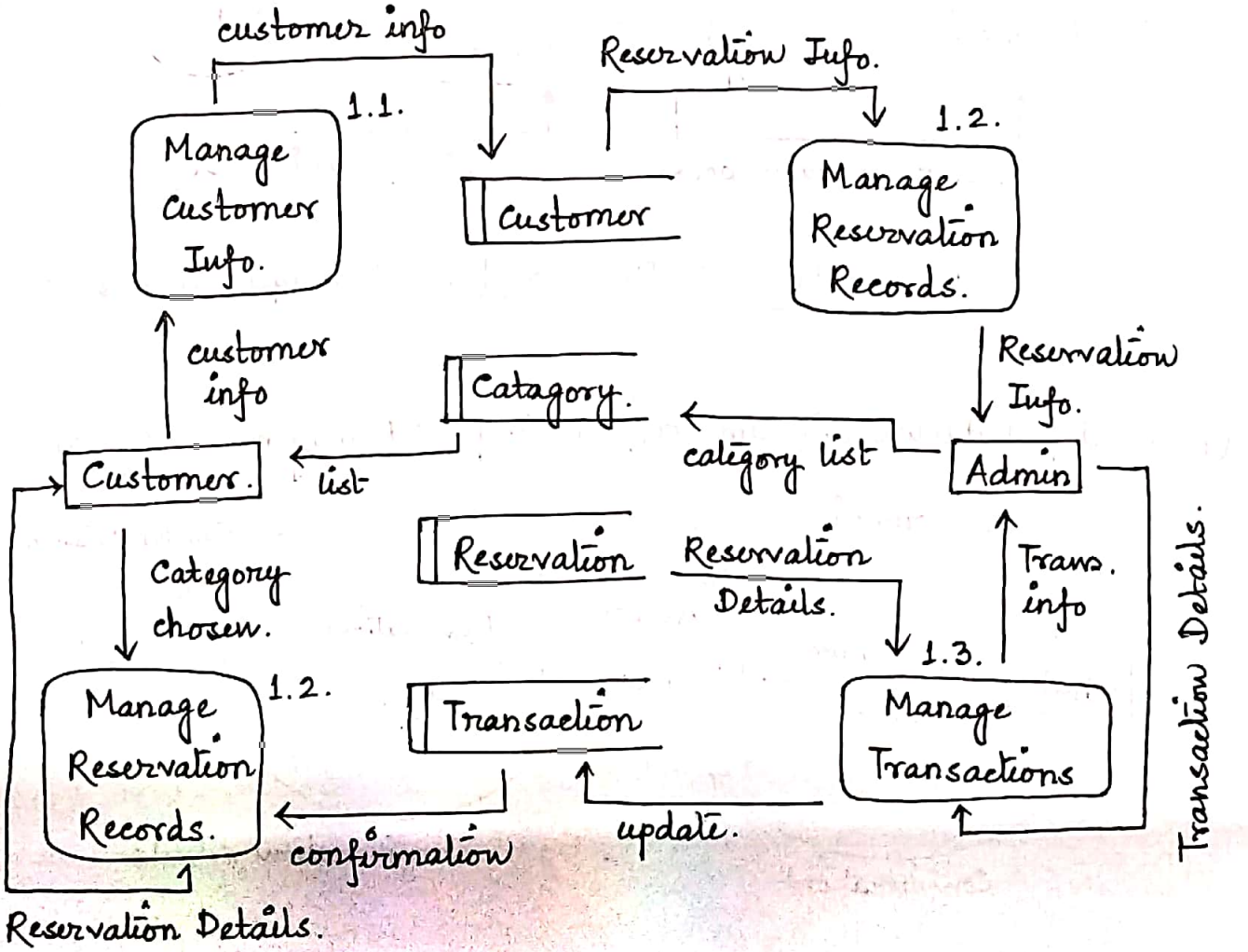


Fig 3: Level-2 DFD of Airport Management System.

⑤ Develop Case diagram for Library Management System.

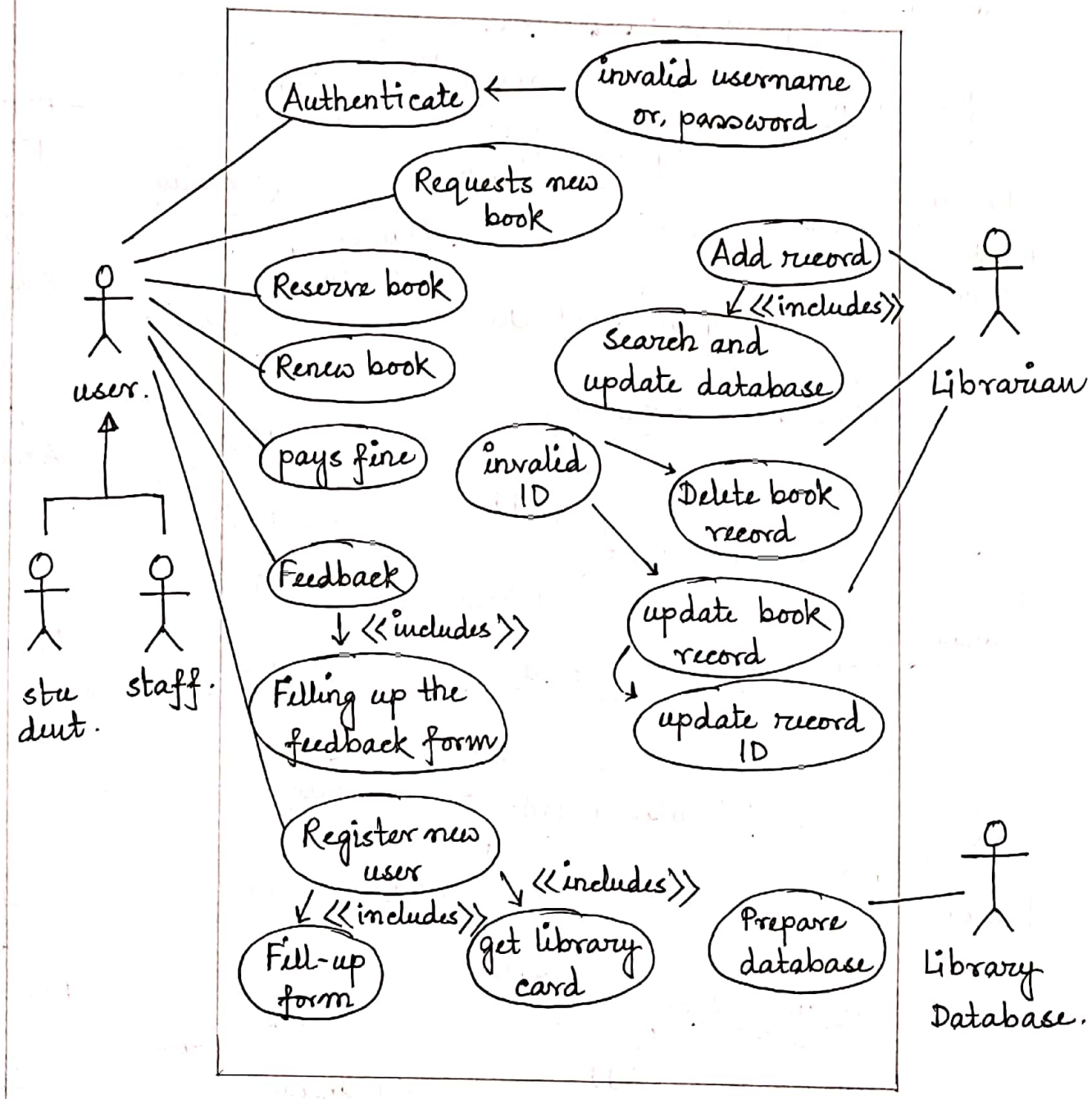


Fig : USE CASE diagram for Library Management System.

⑥ Develop Case diagram for Mall Management System.

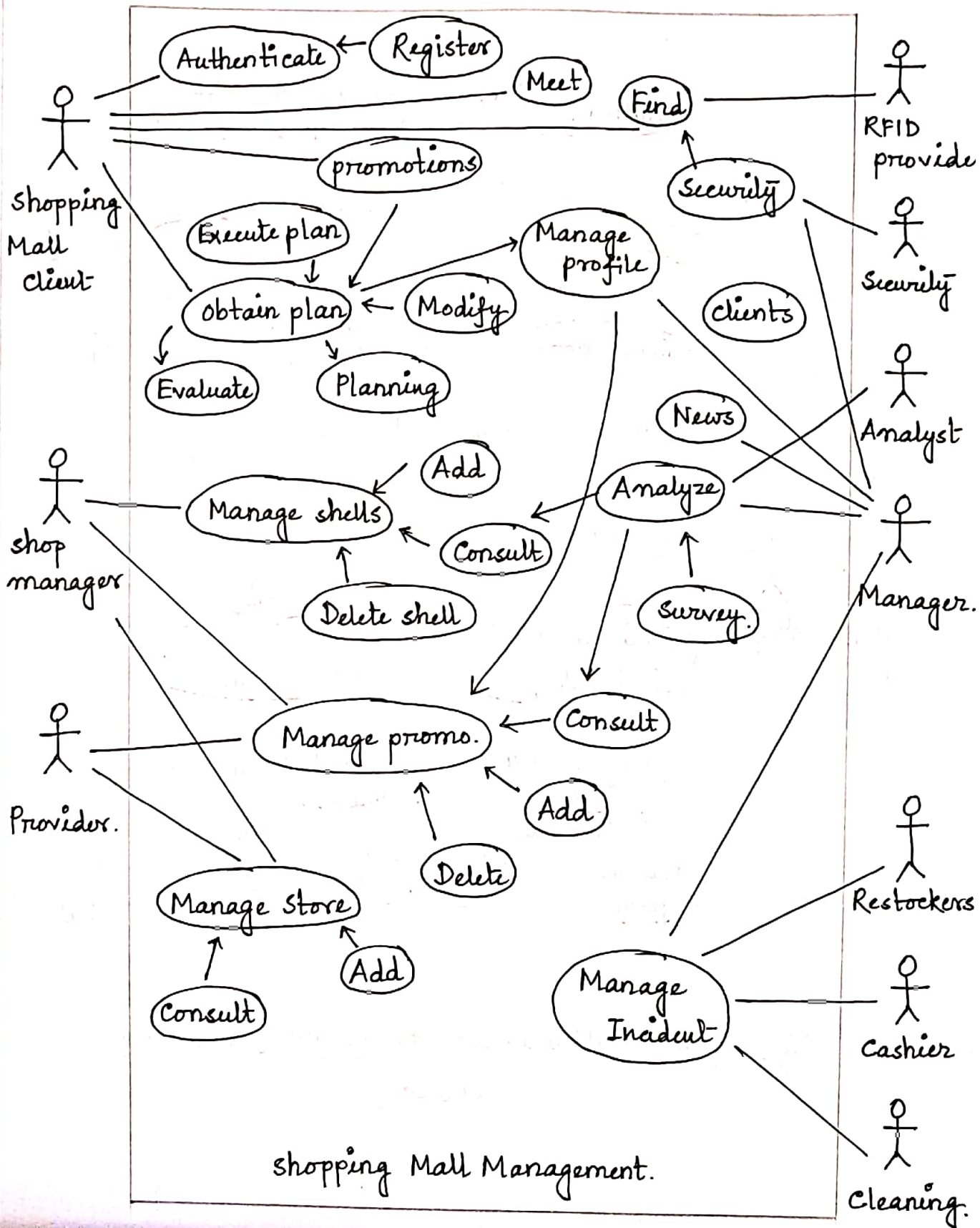


Fig : USE CASE diagram for Mall Management System.

⑦ Develop a case diagram for Hotel Management System.

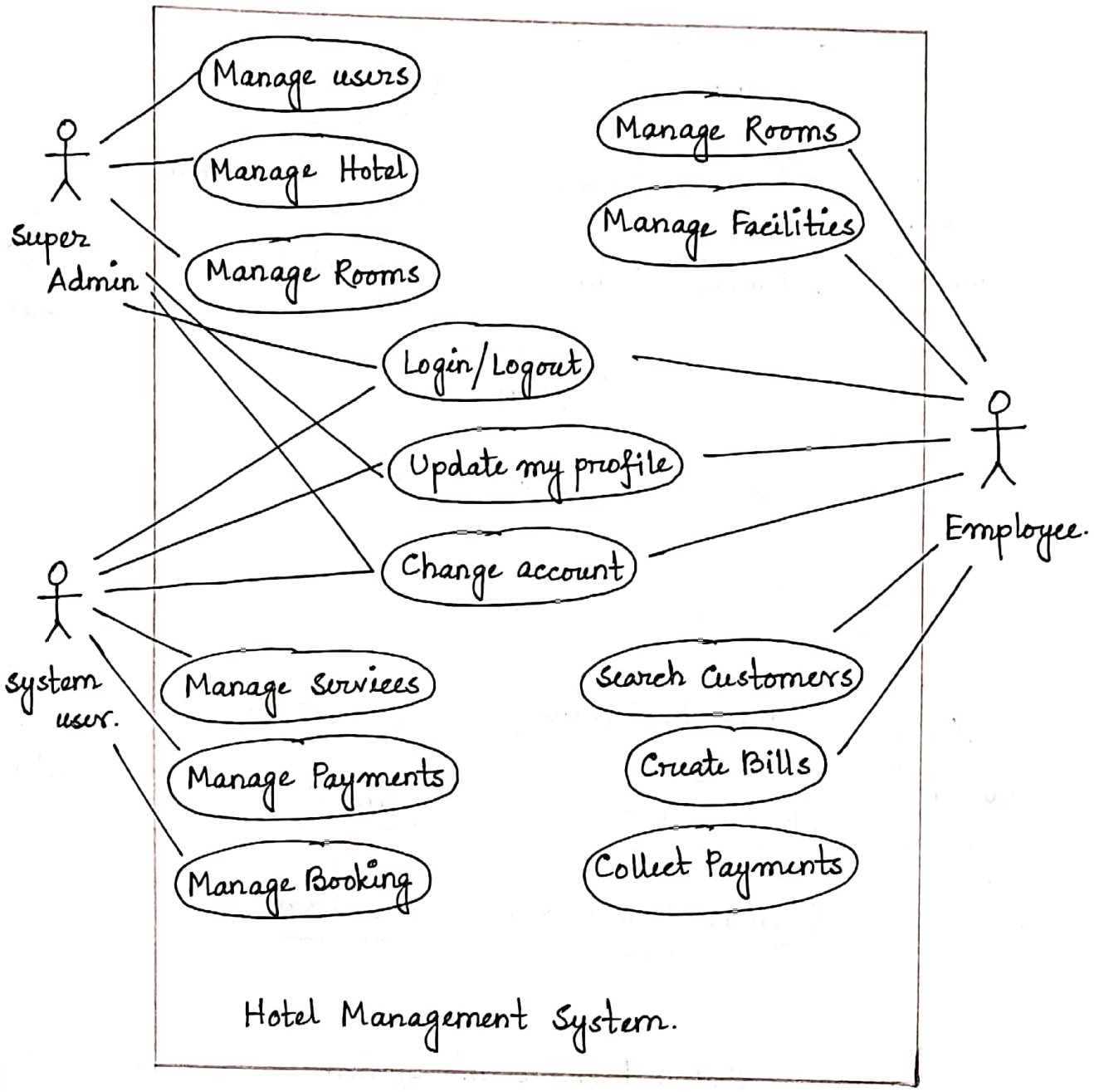


Fig : USE CASE diagram for Hotel Management System

(8) Design the case diagram for College Management System.

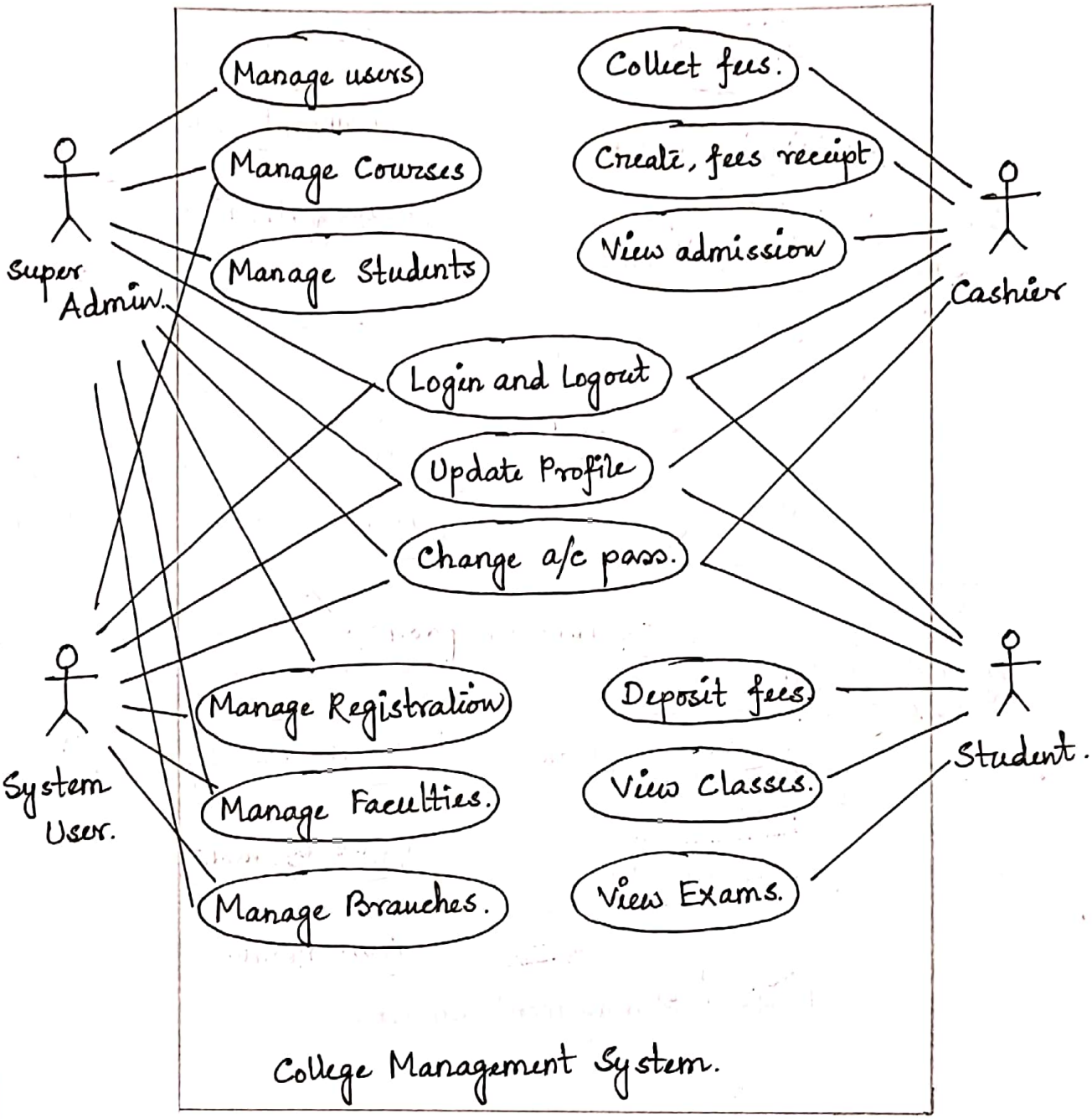


Fig : USE CASE diagram for College Management System.

(9) Develop use case diagram for School Management System.

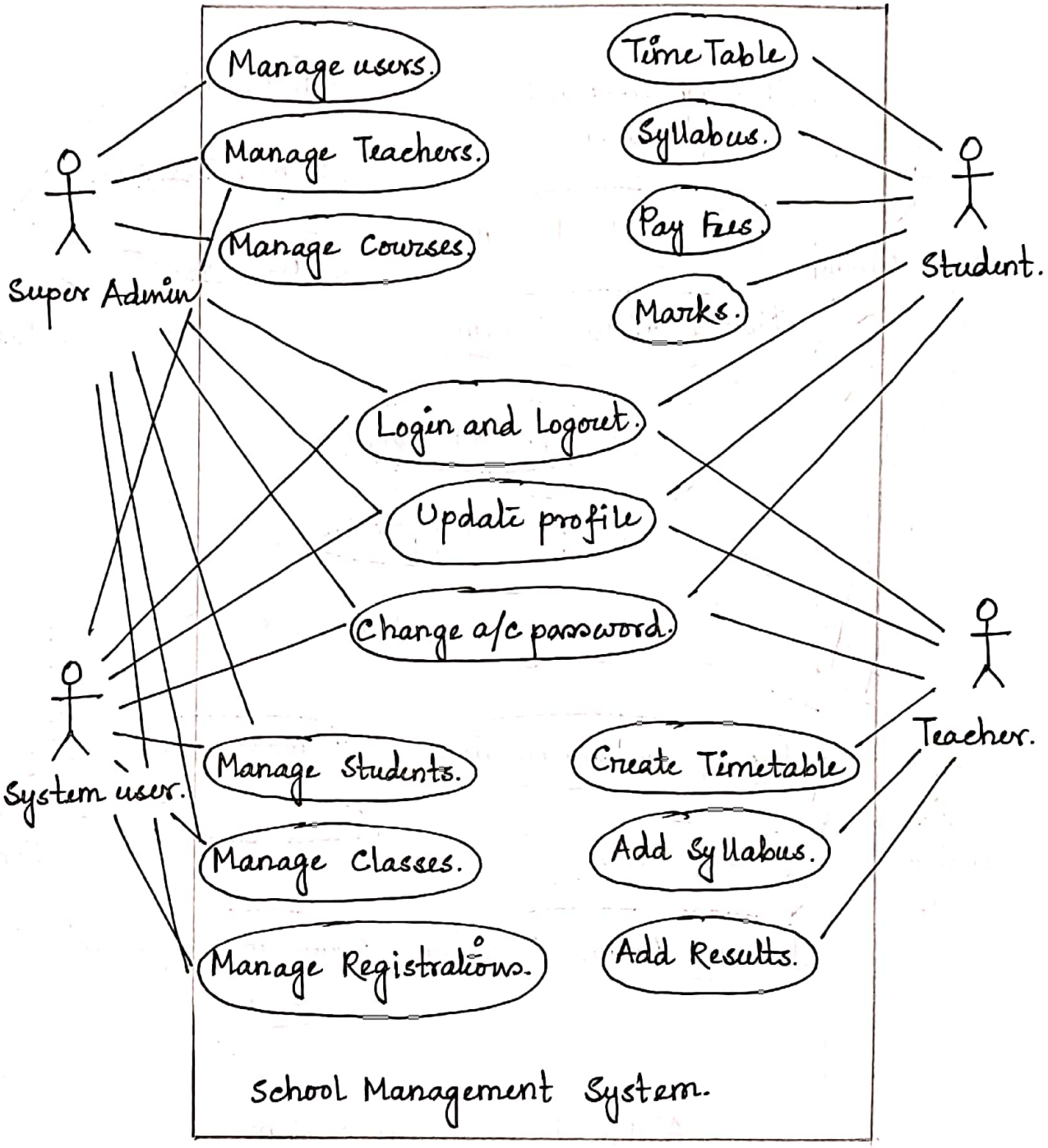


Fig : USE CASE diagram for School Management System.

(10) Develop the case diagram for Airline Management System.

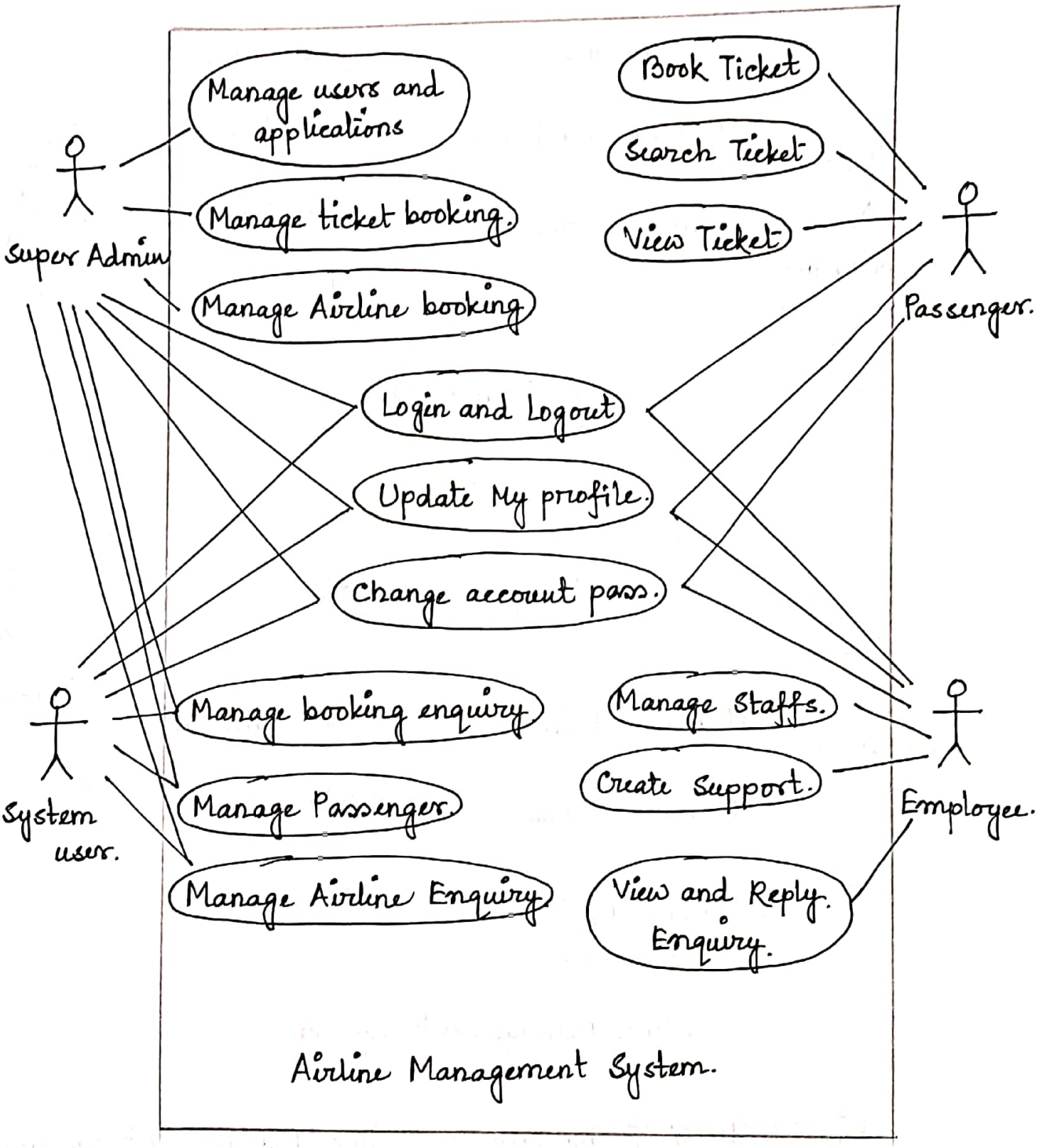


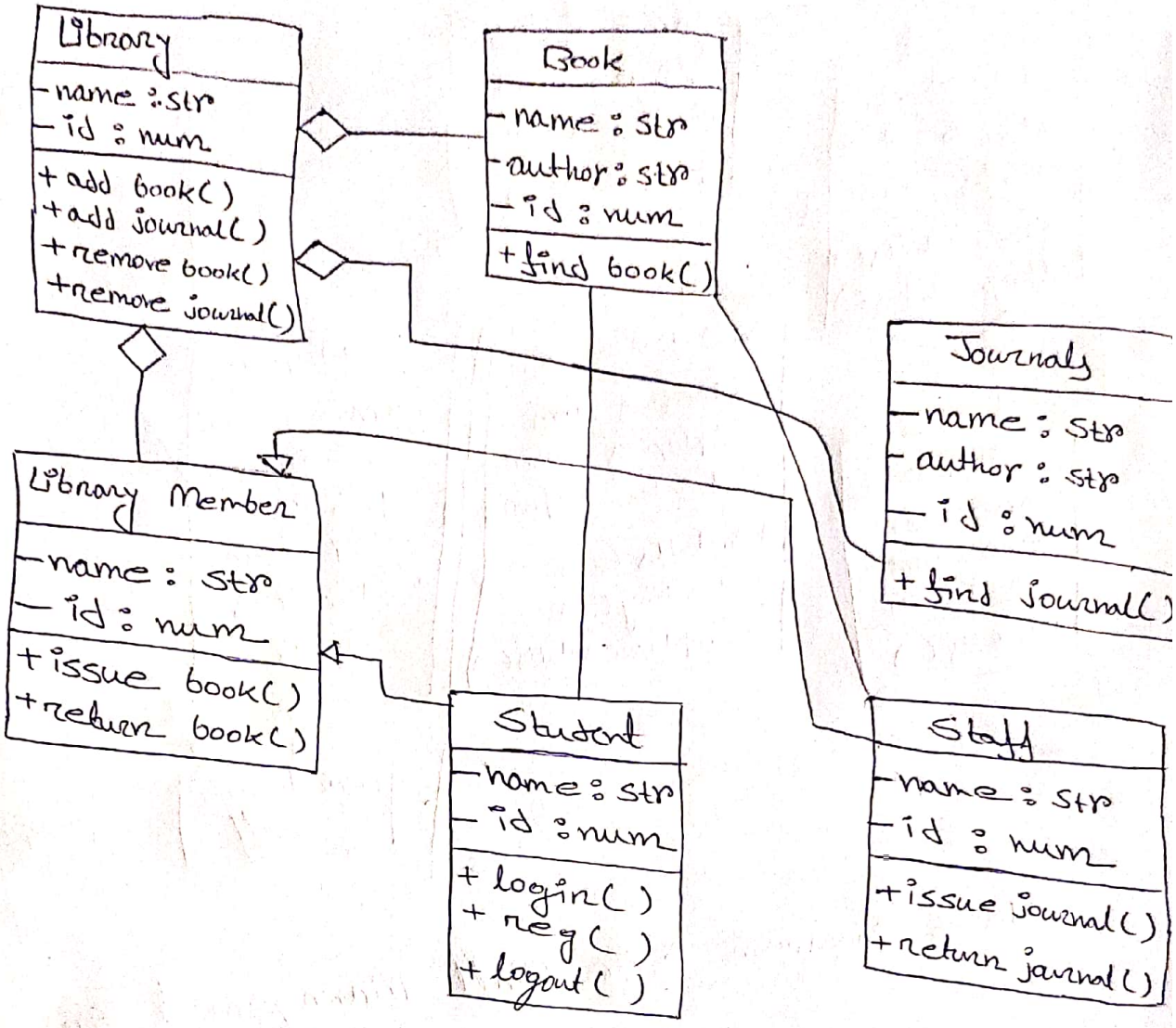
Fig : USE CASE diagram for Airline Management System.

RB

Assignment-3

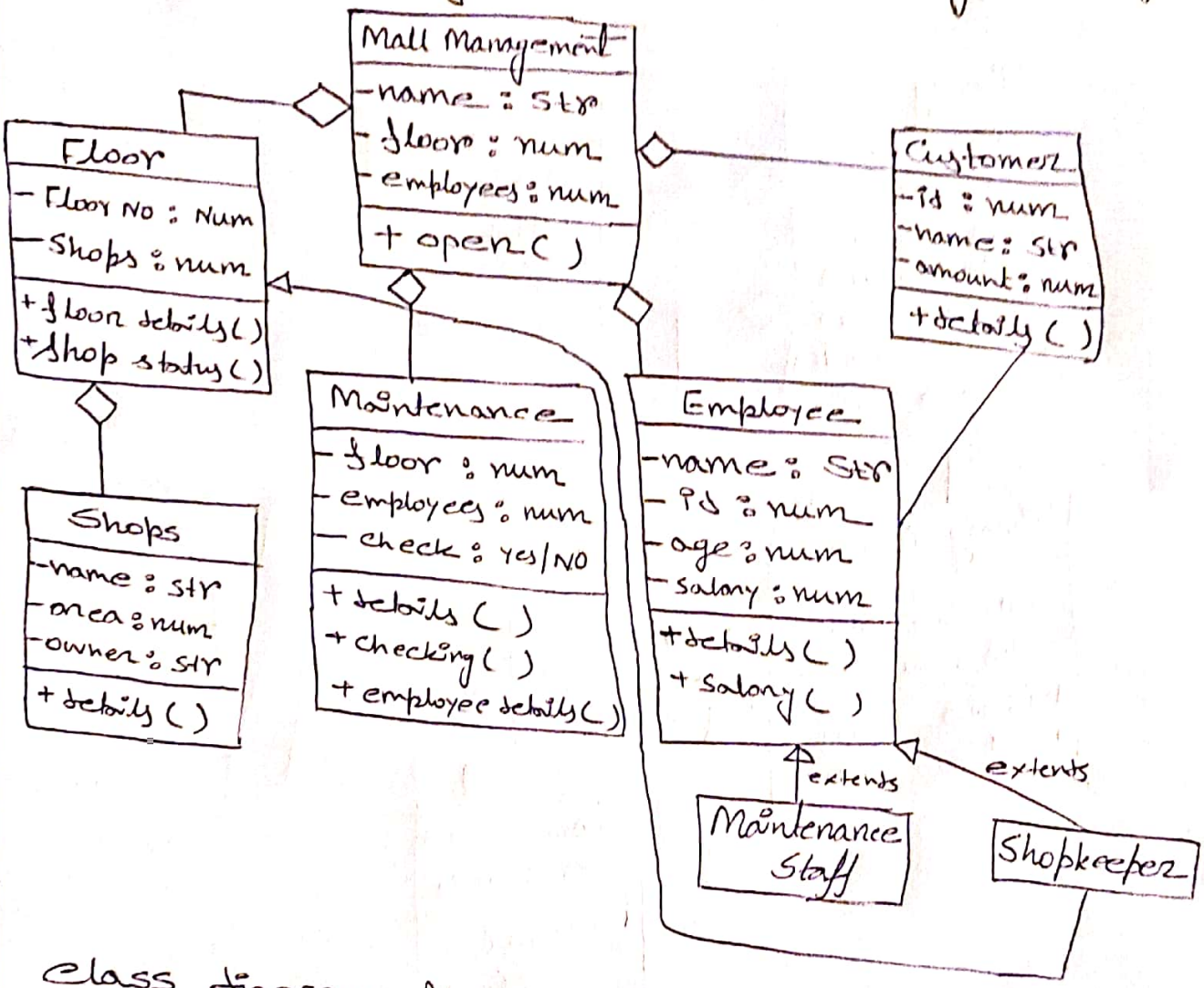
08/03/22

1. Develop a class diagram for Library Management System :



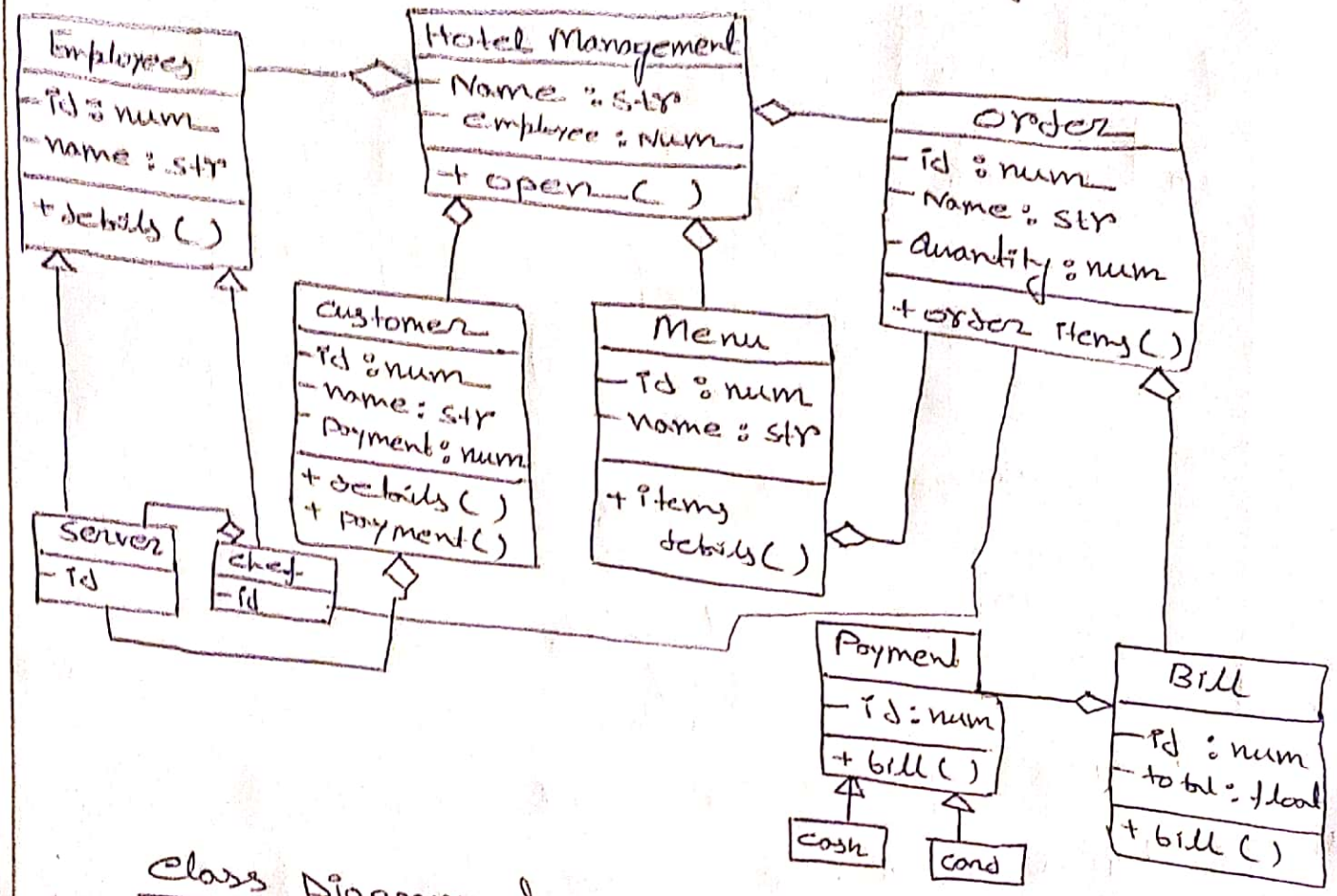
Class Diagram for Library Management System

2. Develop class Diagram for Mall Management System.



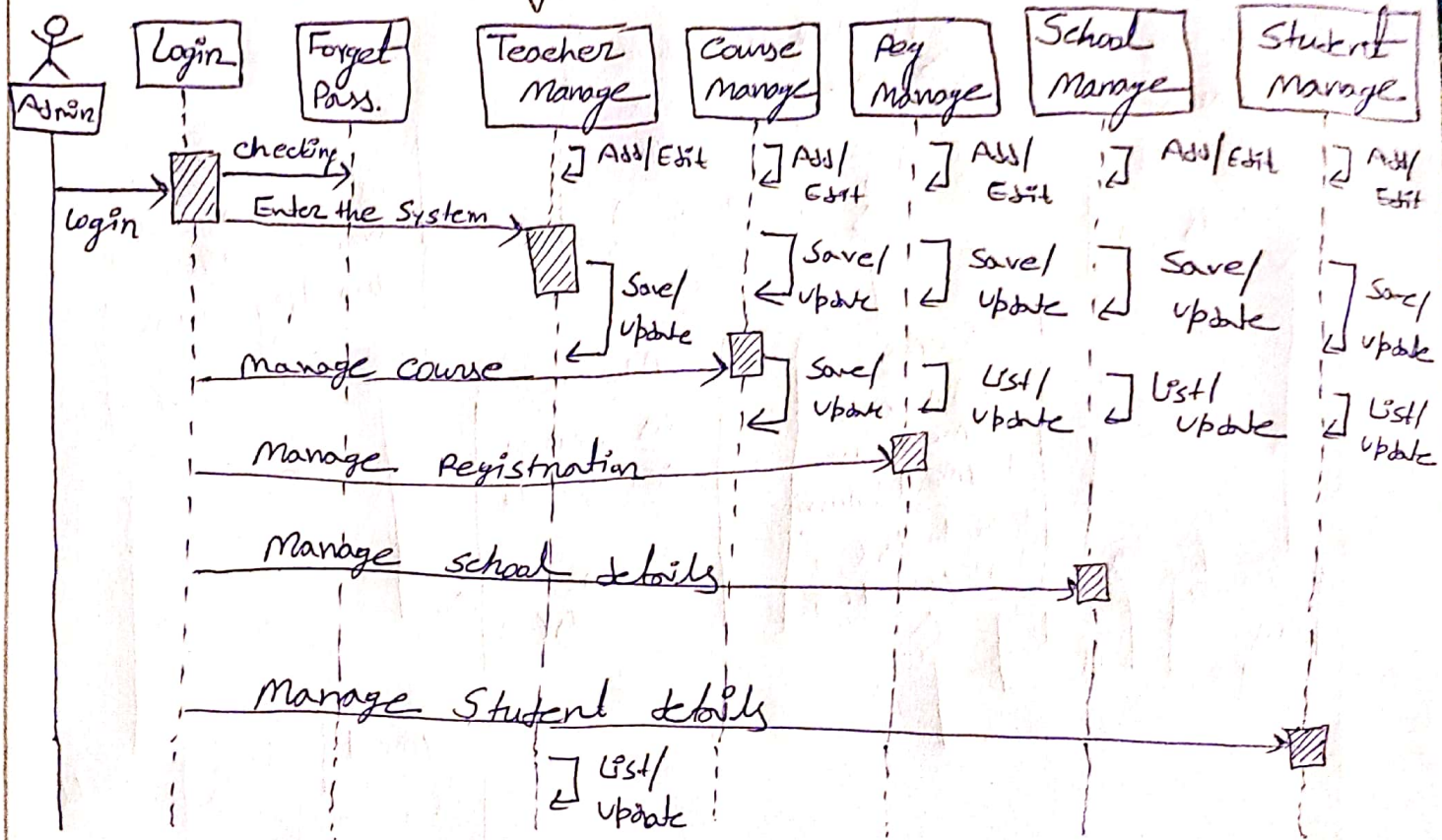
Class diagram for Mall Management System

3. Develop class diagram for Hotel Management System



Class Diagram for Hotel Management System

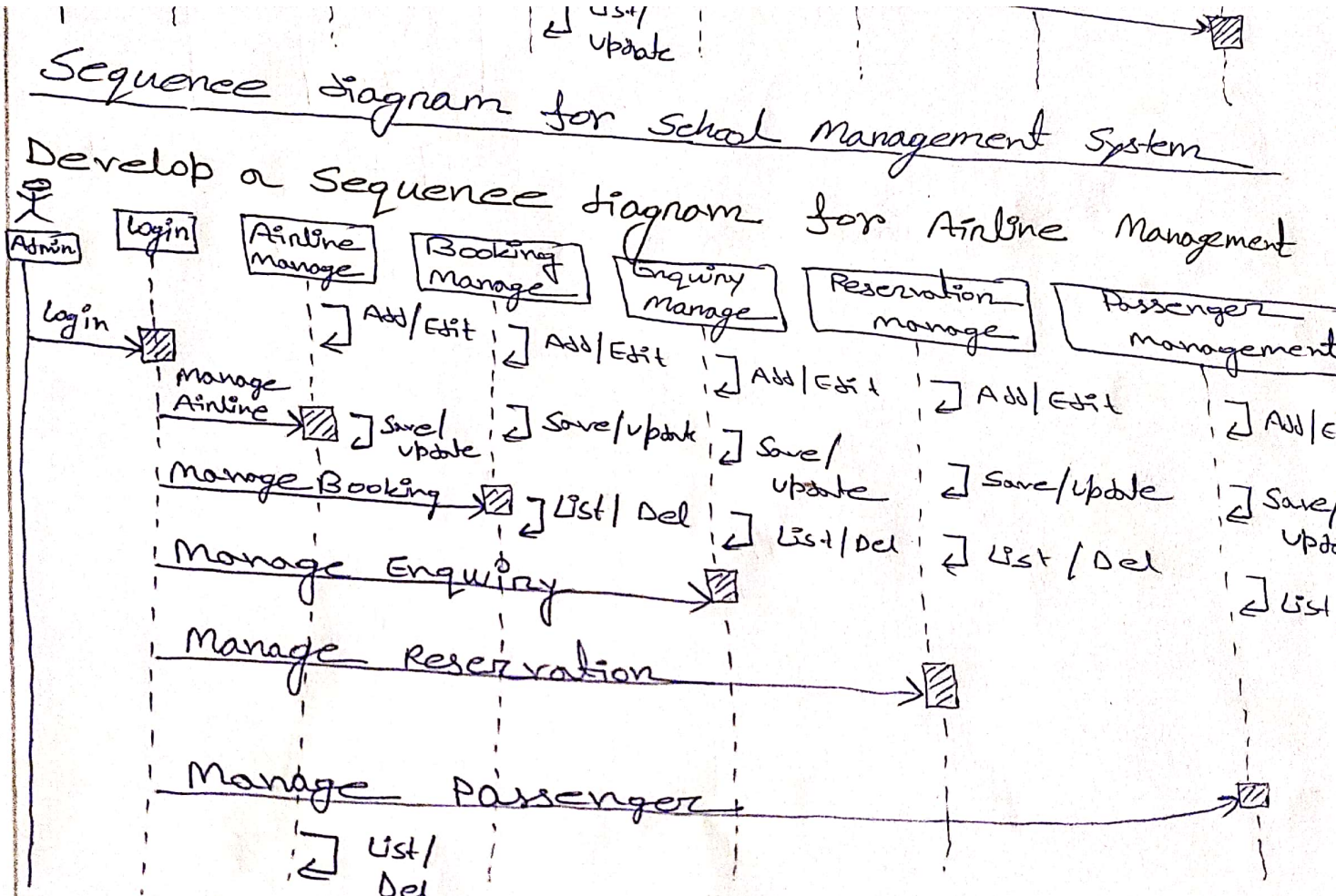
5. Develop Sequence diagram for School Management System



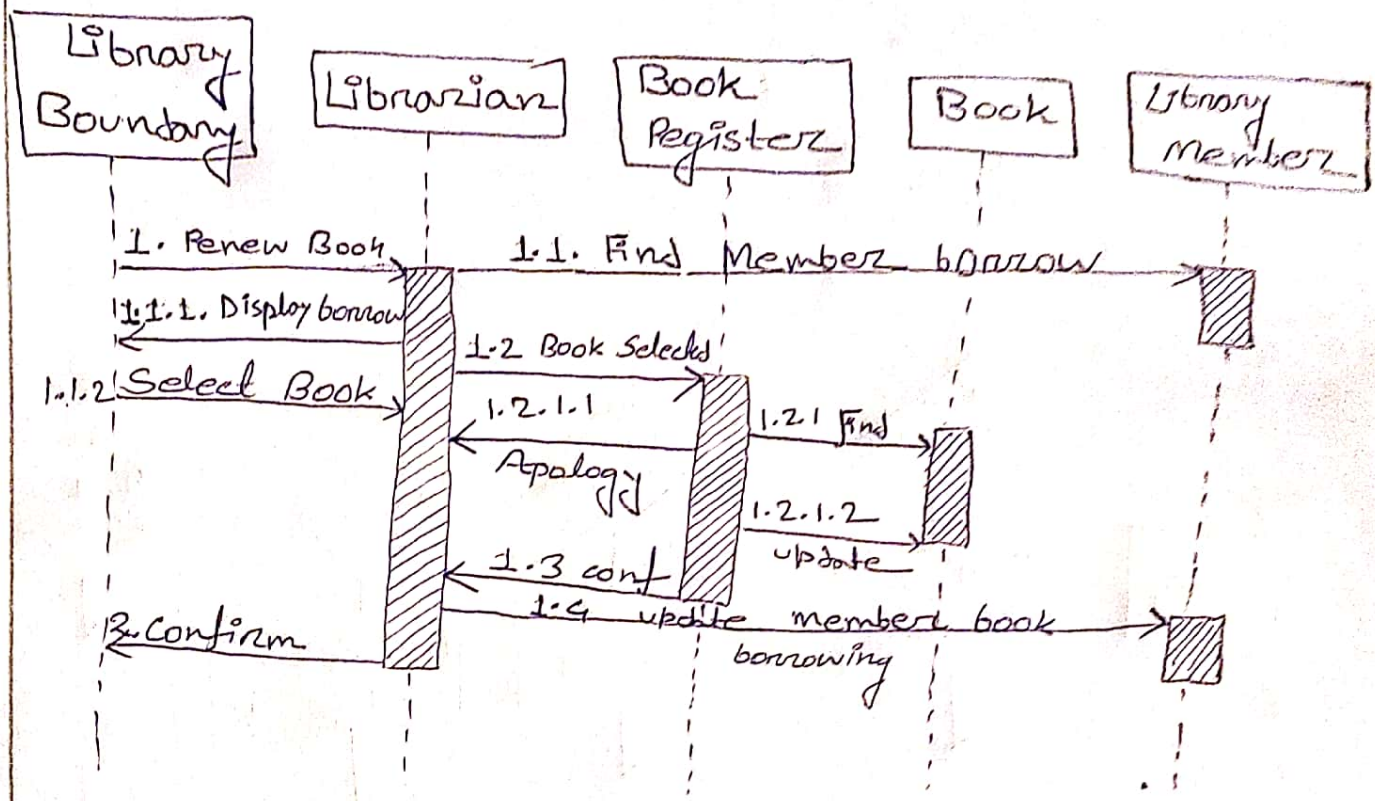
Sequence diagram for School Management System

5. Develop a Sequence diagram for Airline Management

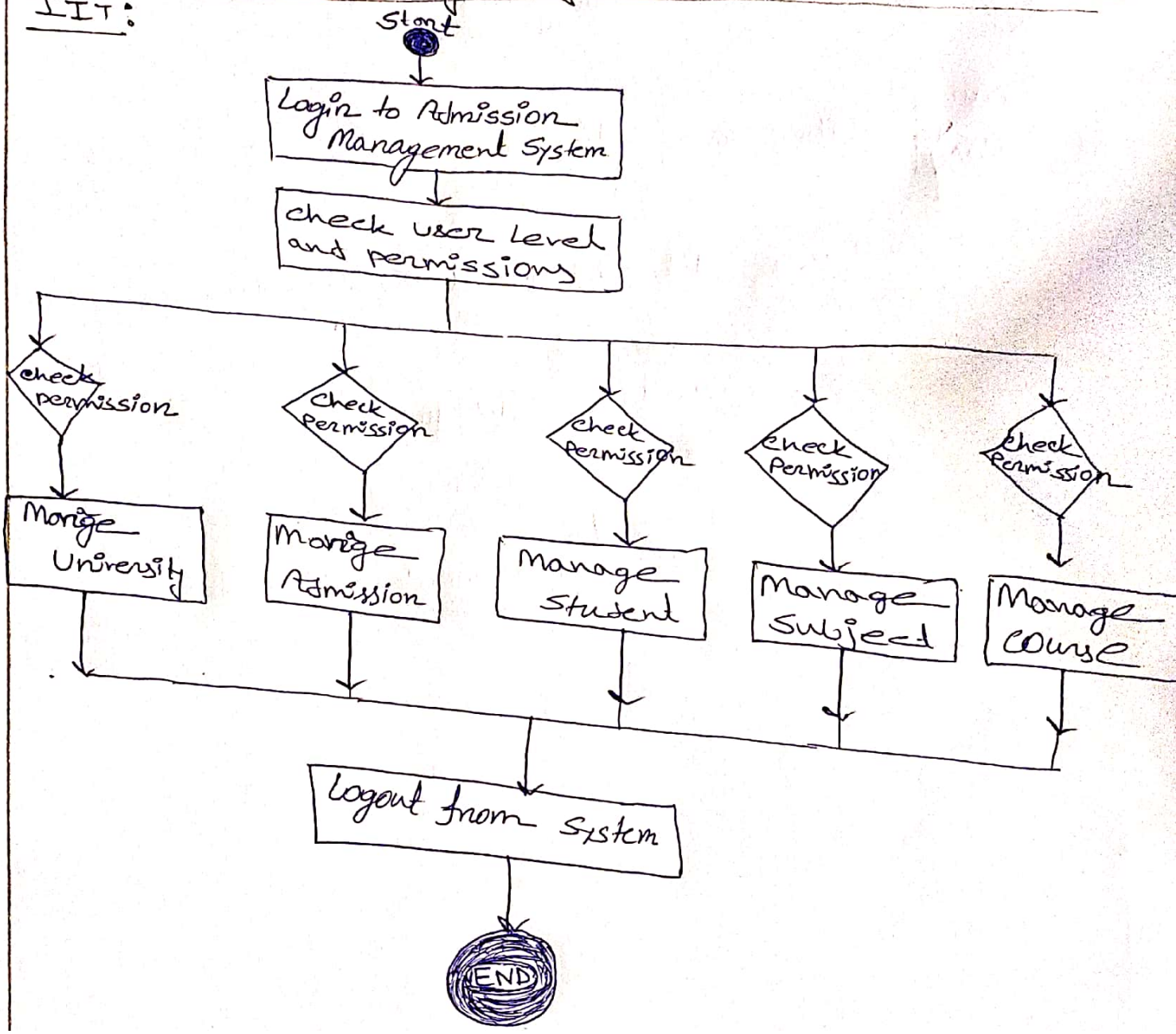
Sequence diagram for School Management System



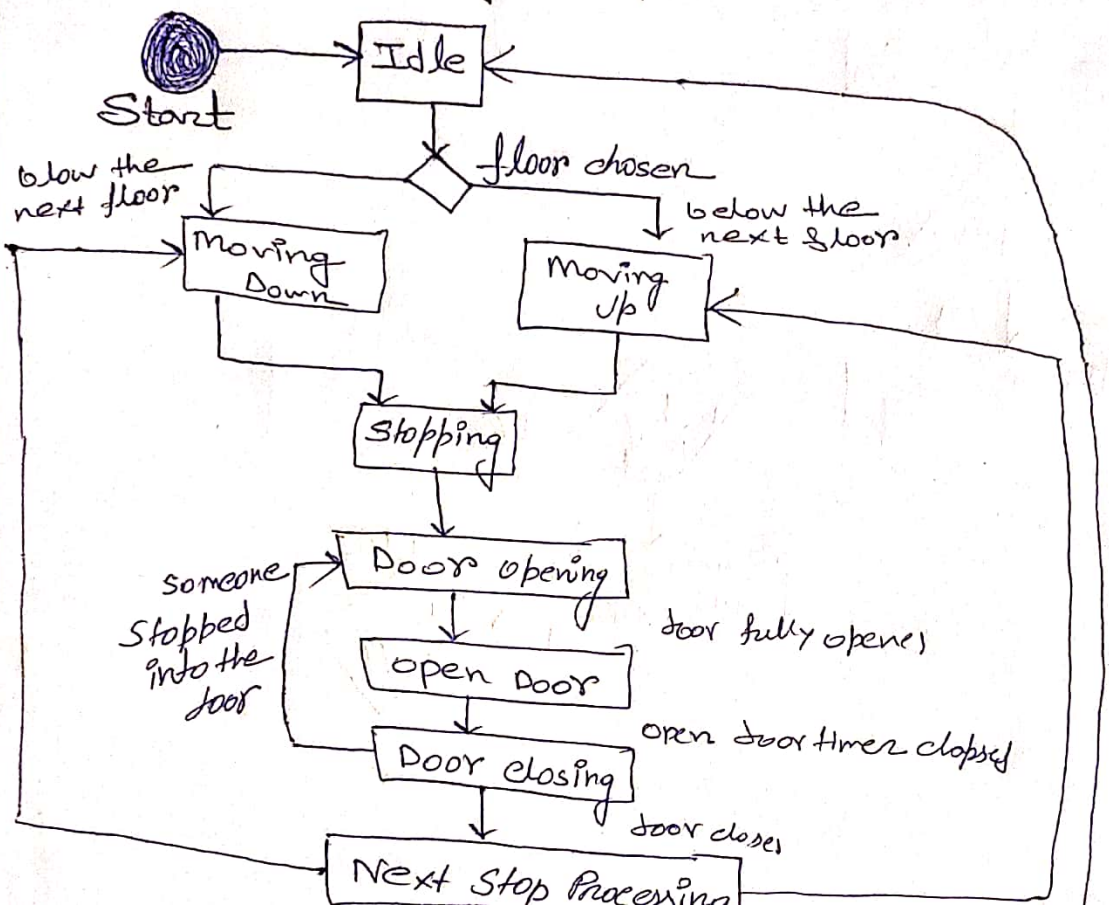
7. Develop Sequence diagram for Book Renew use case



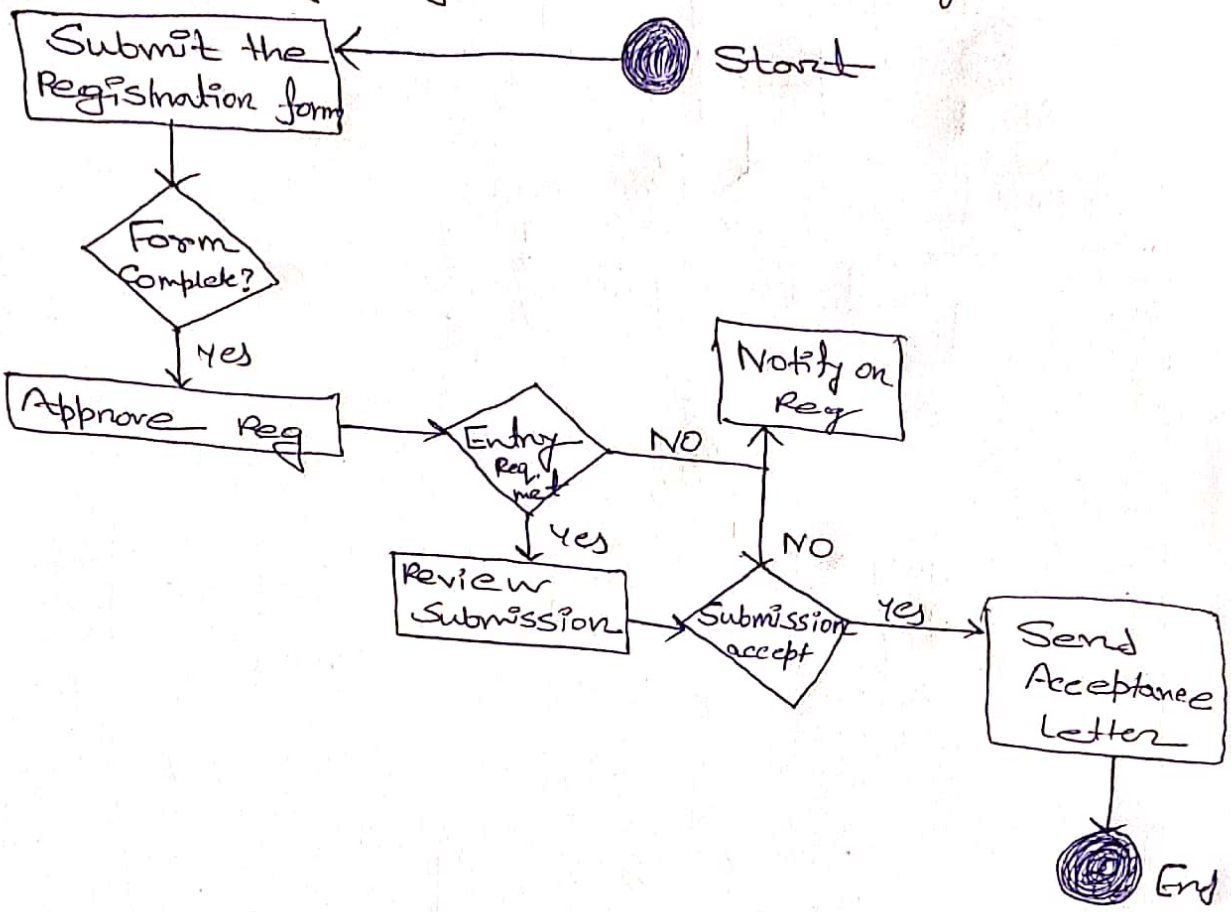
8. Develop an Activity Diagram for Admission procedure at IIT:



9. Develop an activity diagram for Elevator System

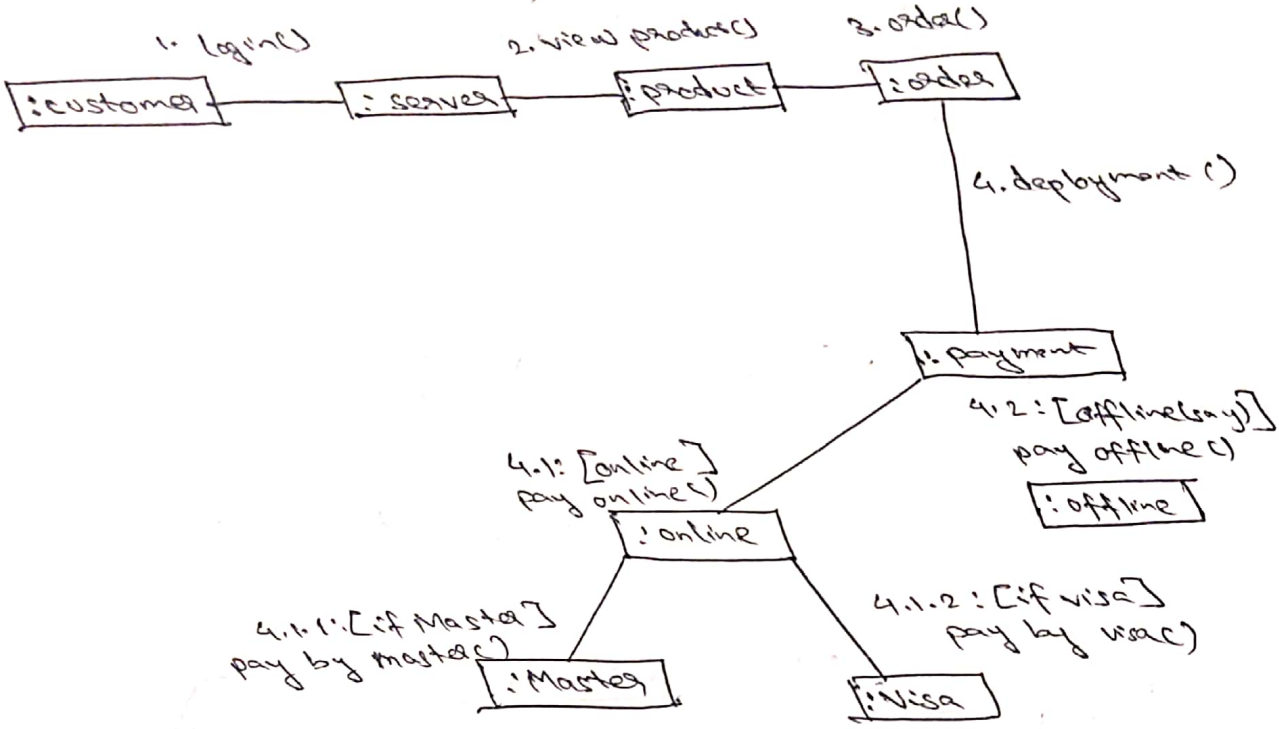


10. Develop an activity diagram for Student Registration System.

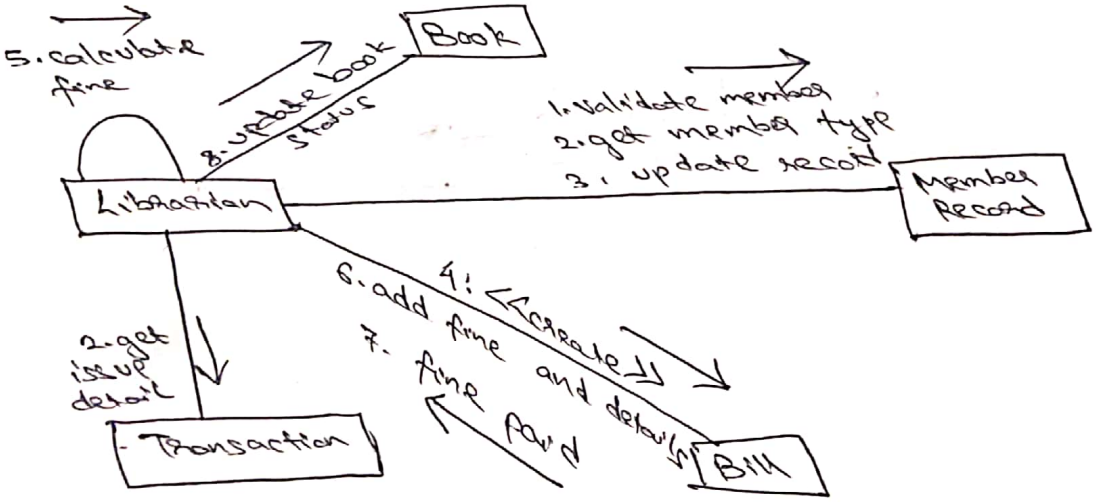


Assignment-4

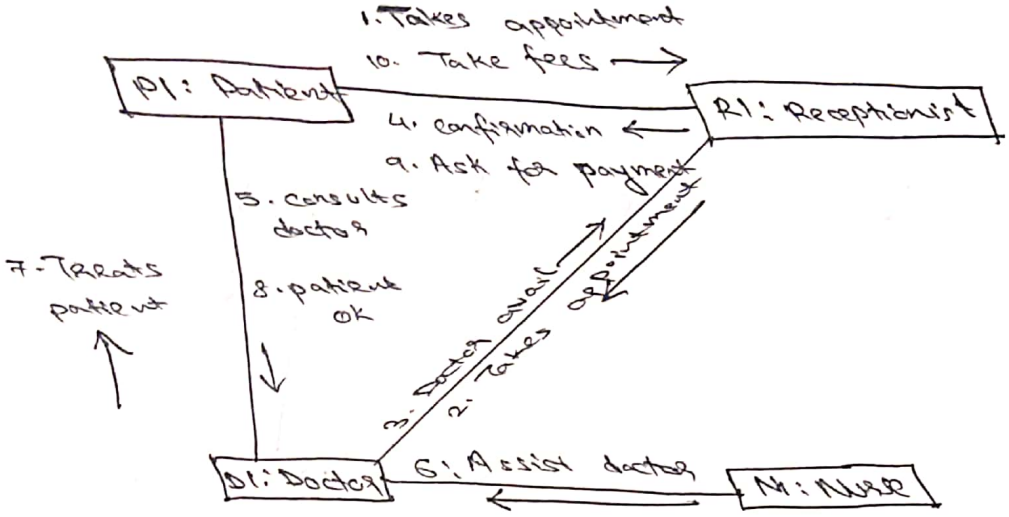
1) Develop a collaboration diagram for Online Payment System.



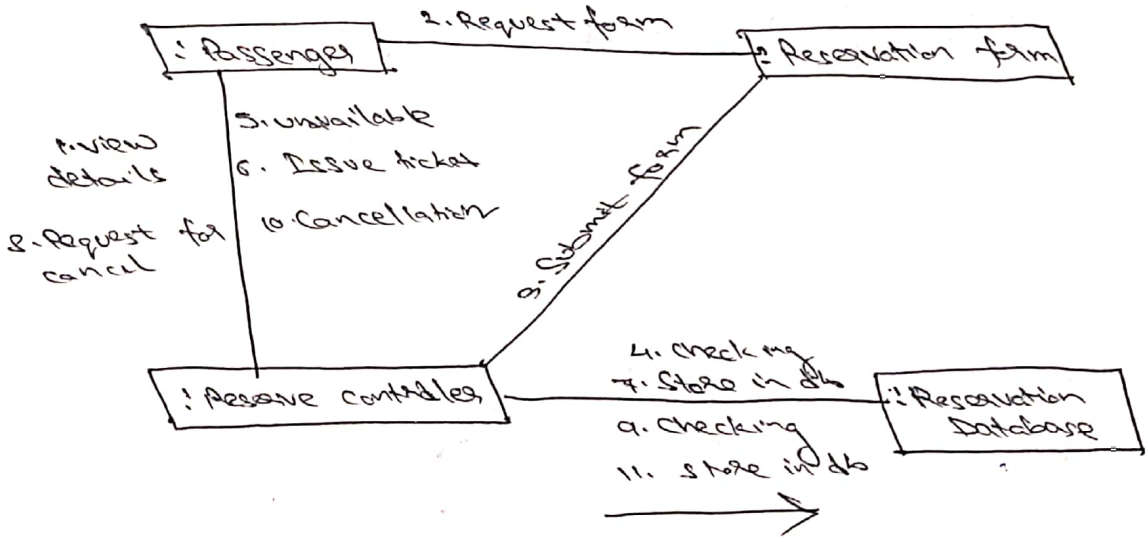
2) Develop a collaboration diagram for Library Management system.



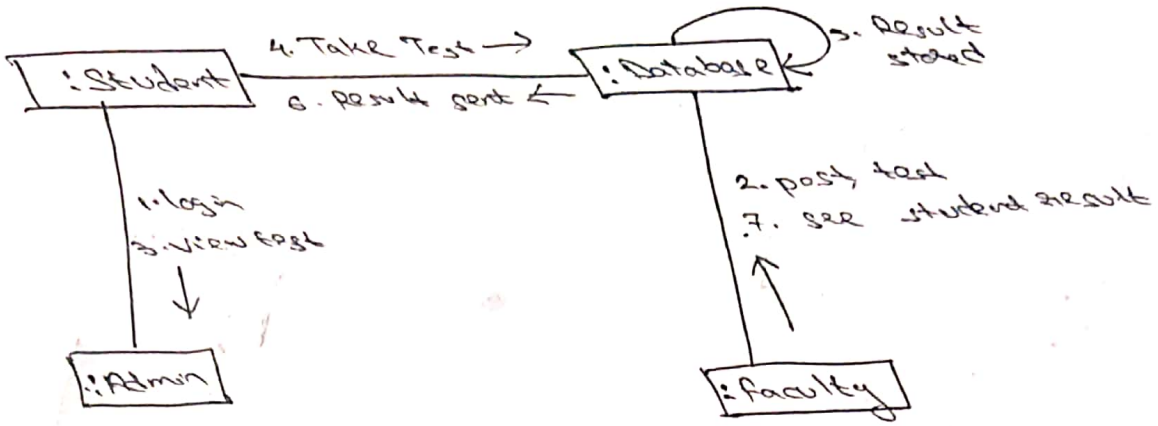
3) Develop a collaboration diagram for Hospital Management System.



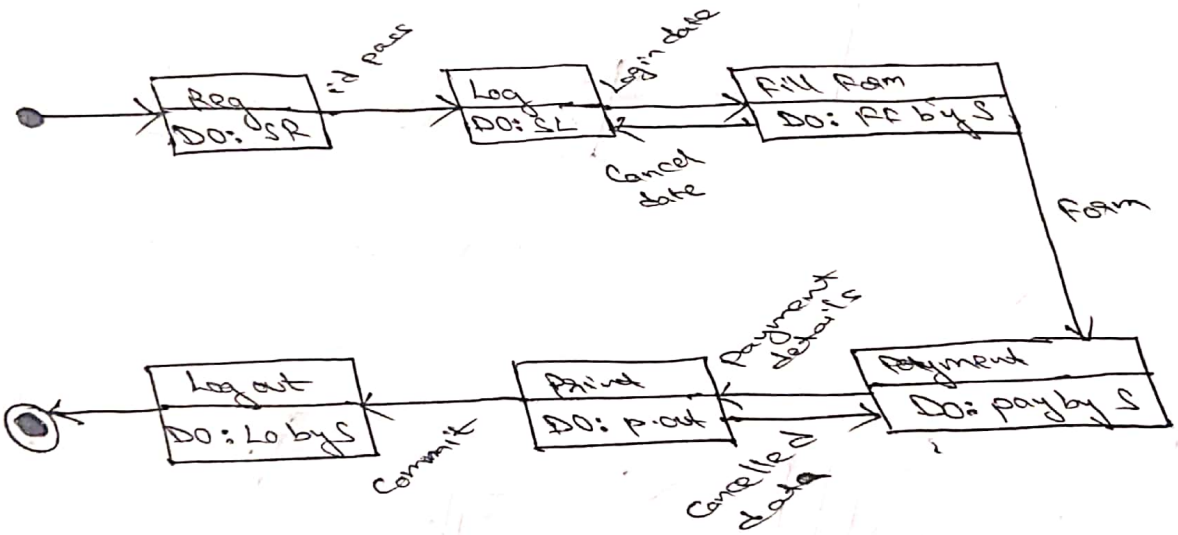
4) Develop a collaboration diagram for Airline Reservation System.



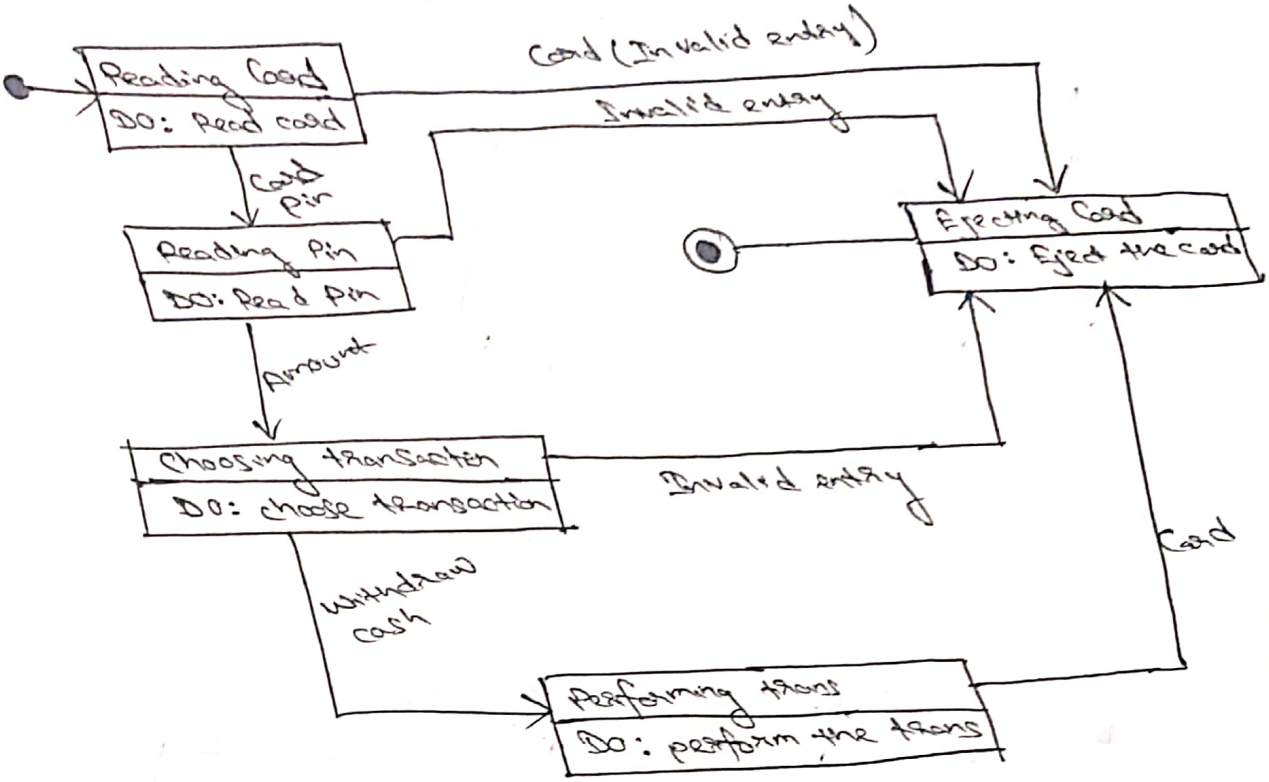
5) Develop a collaboration diagram for Online Exam System.



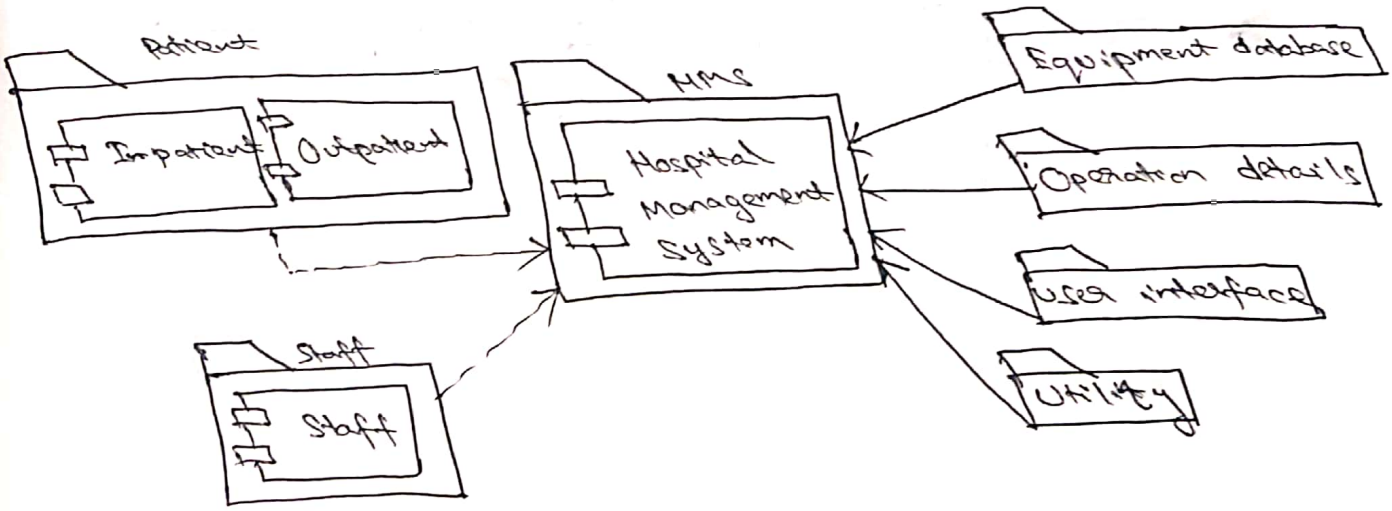
6) Develop a state chart diagram for university form fill up-



7) Develop a state chart diagram for ATM Management System.

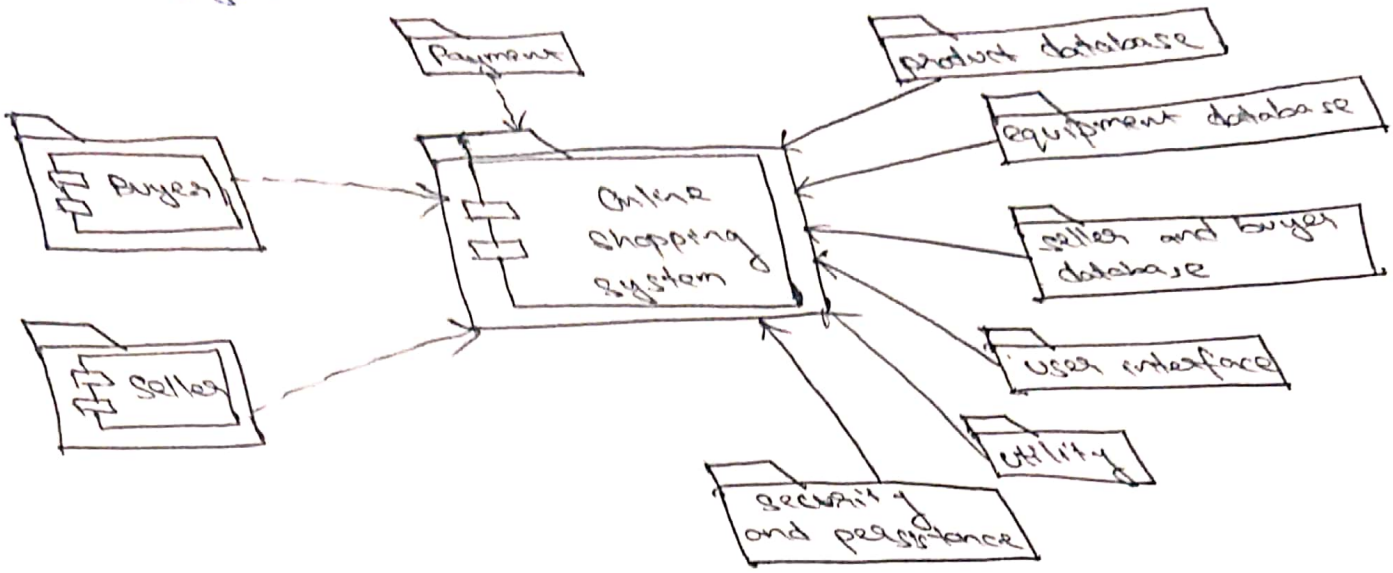


8) Develop component diagram for Hospital Management System.

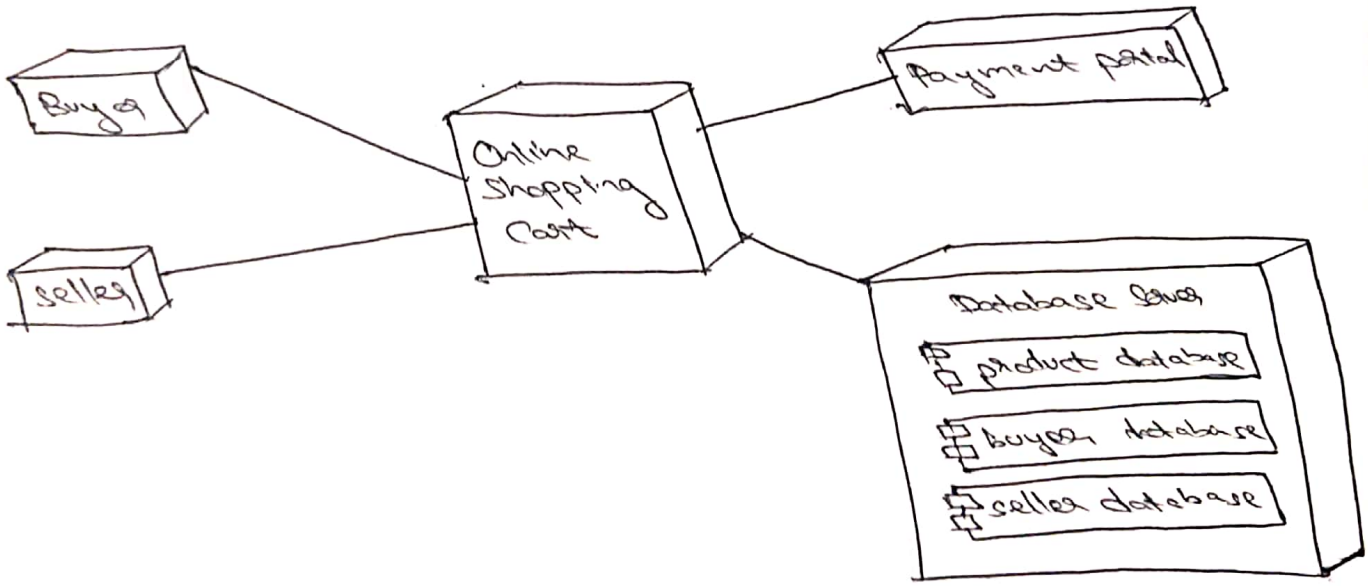


Pg 6

9) Develop a component diagram for online shopping system.

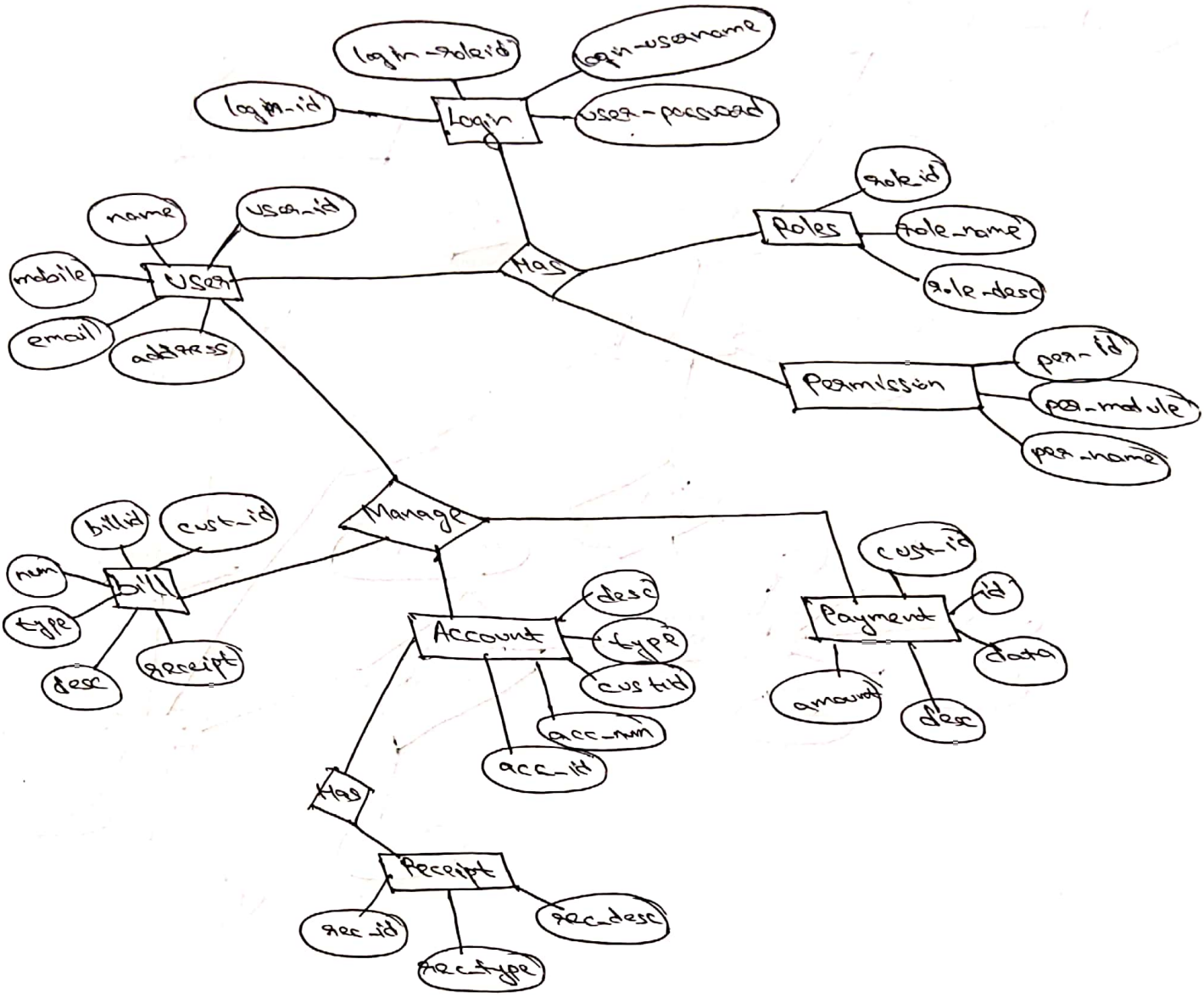


10) Develop deployment diagram for online shopping system.

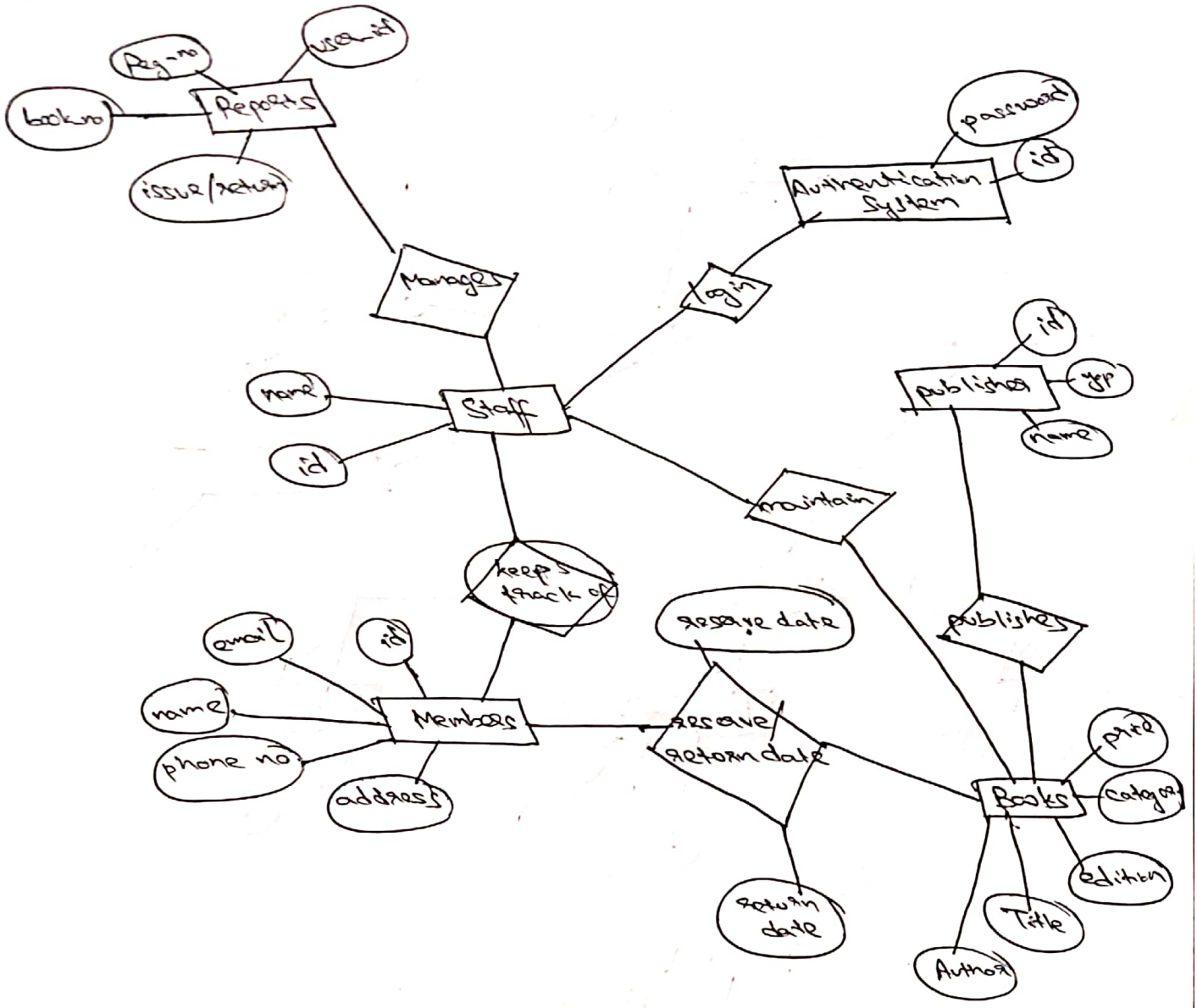


Assignment-5

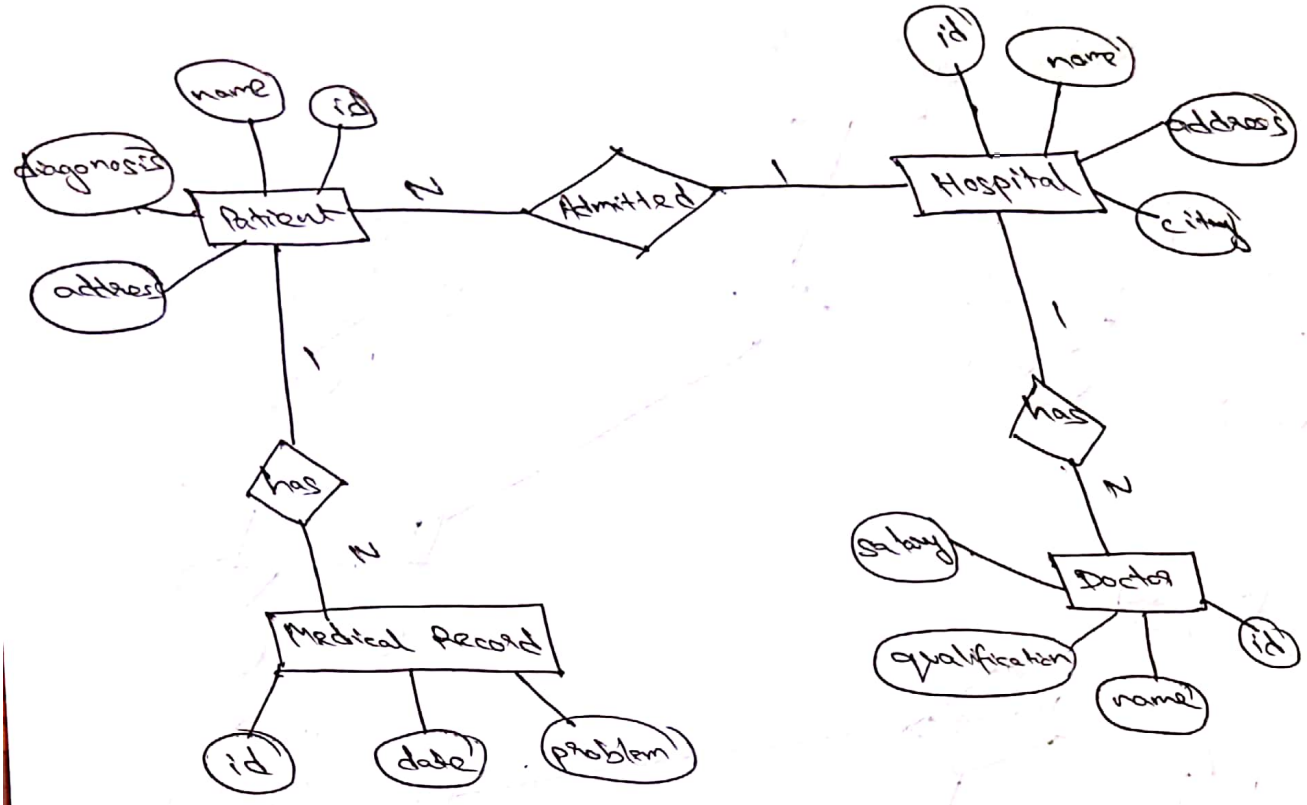
1) Develop ER diagram for Online Payment System.



* Develop ER diagram for Library Management System.

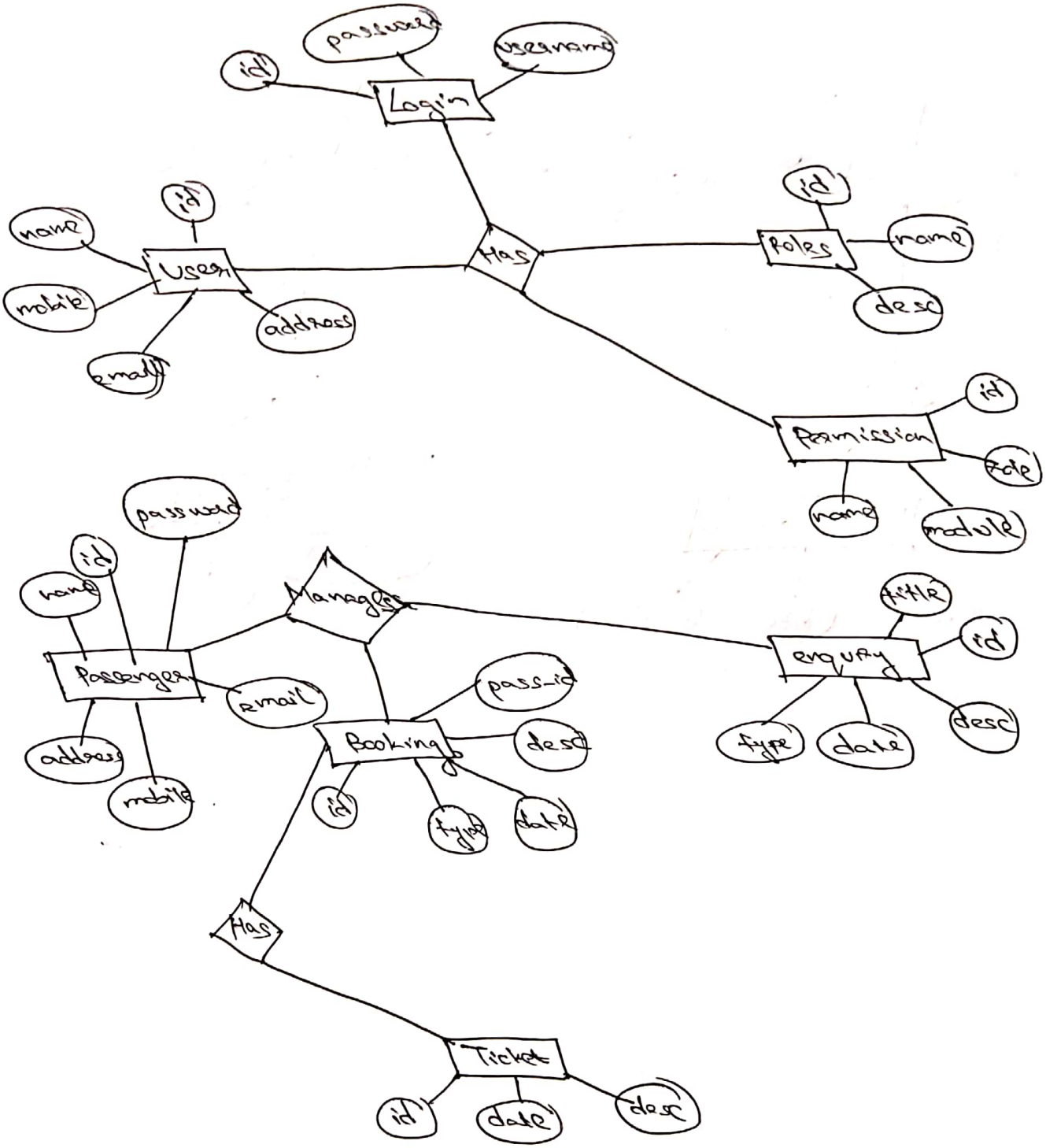


3) Develop ER diagram for Hospital Management System.

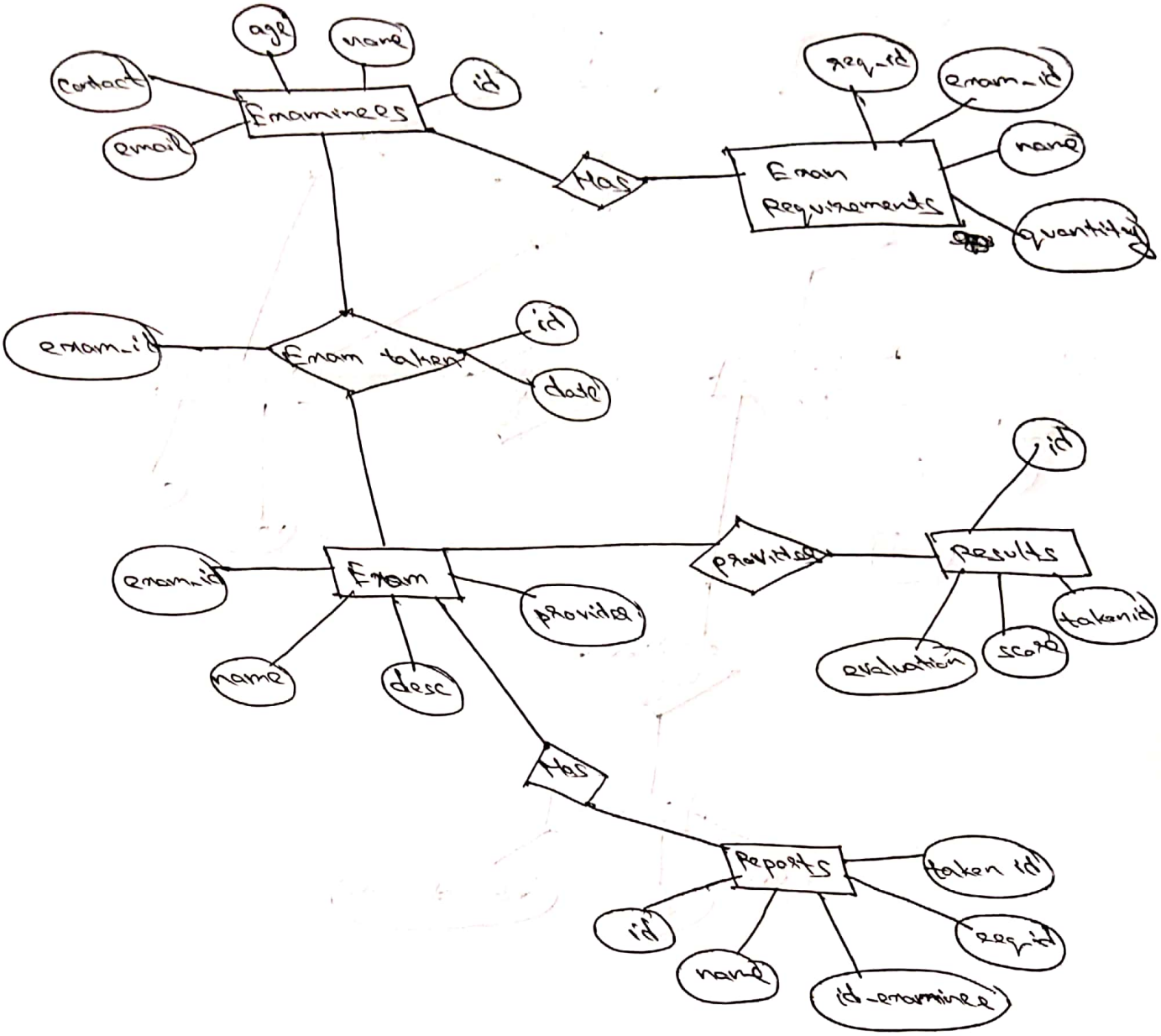


Q.8

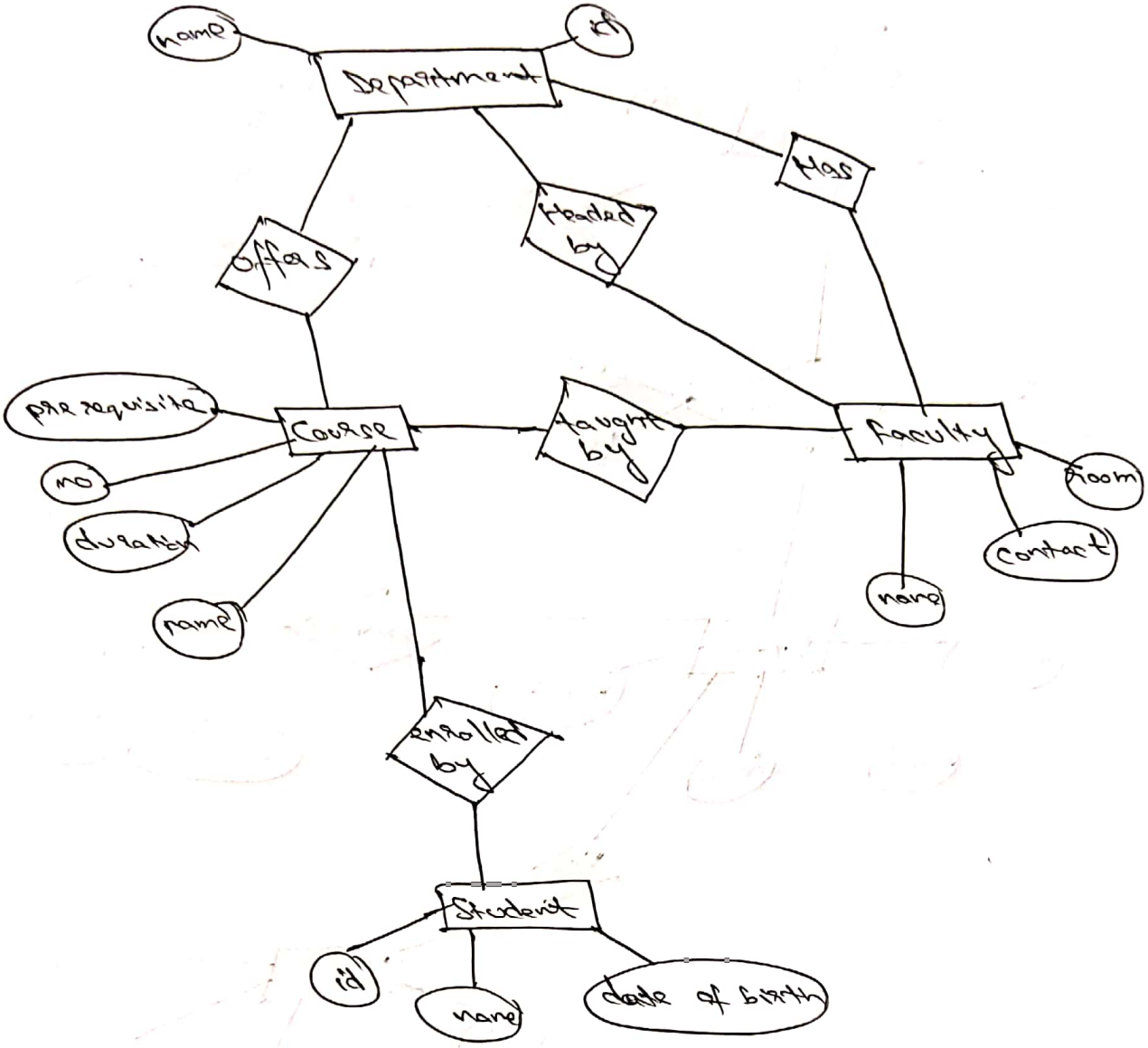
Develop ER diagram for Airline Reservation System.



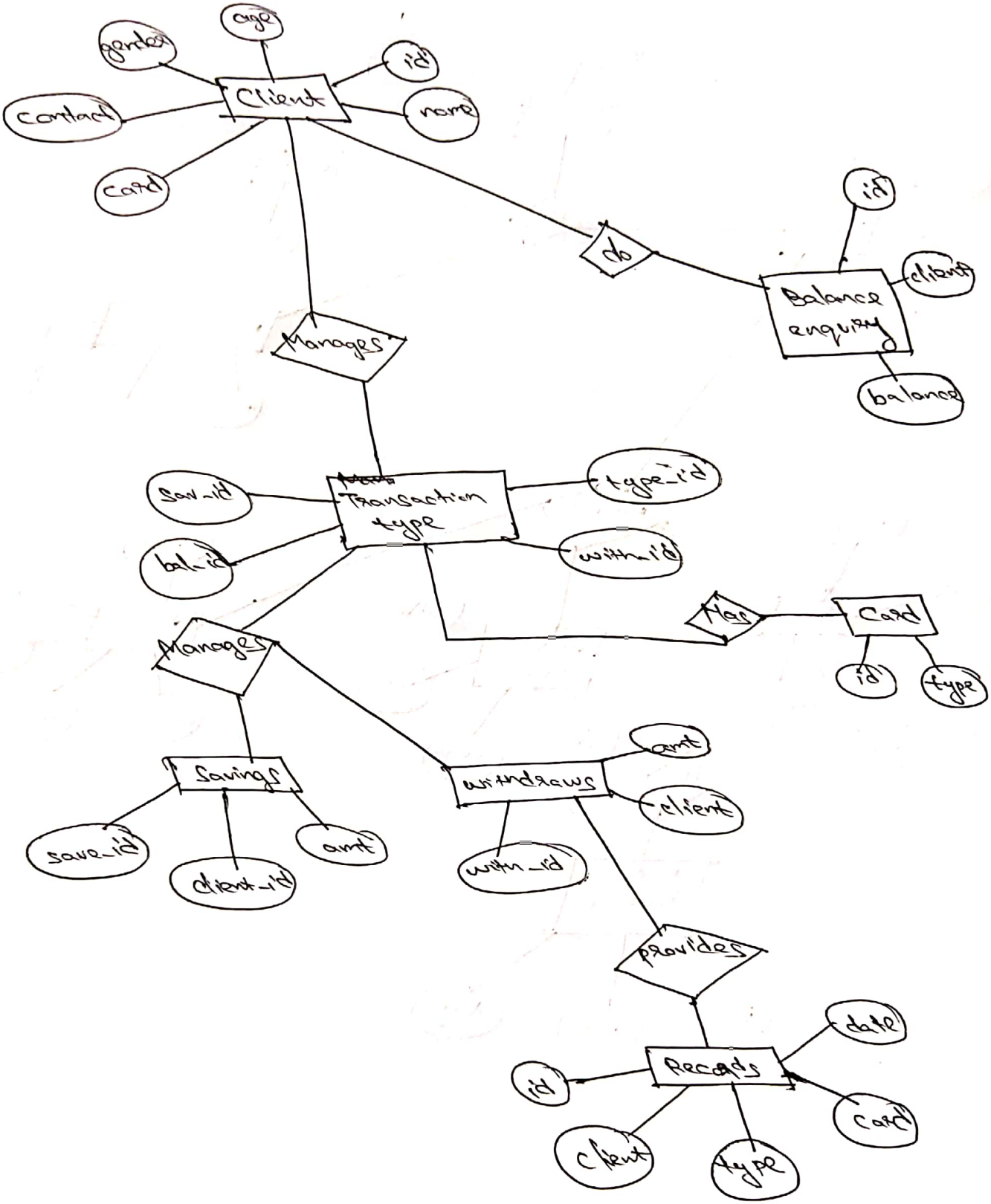
5) Develop ER diagram for Online Examination system.



6) Develop ER diagram for University form Fill up system.

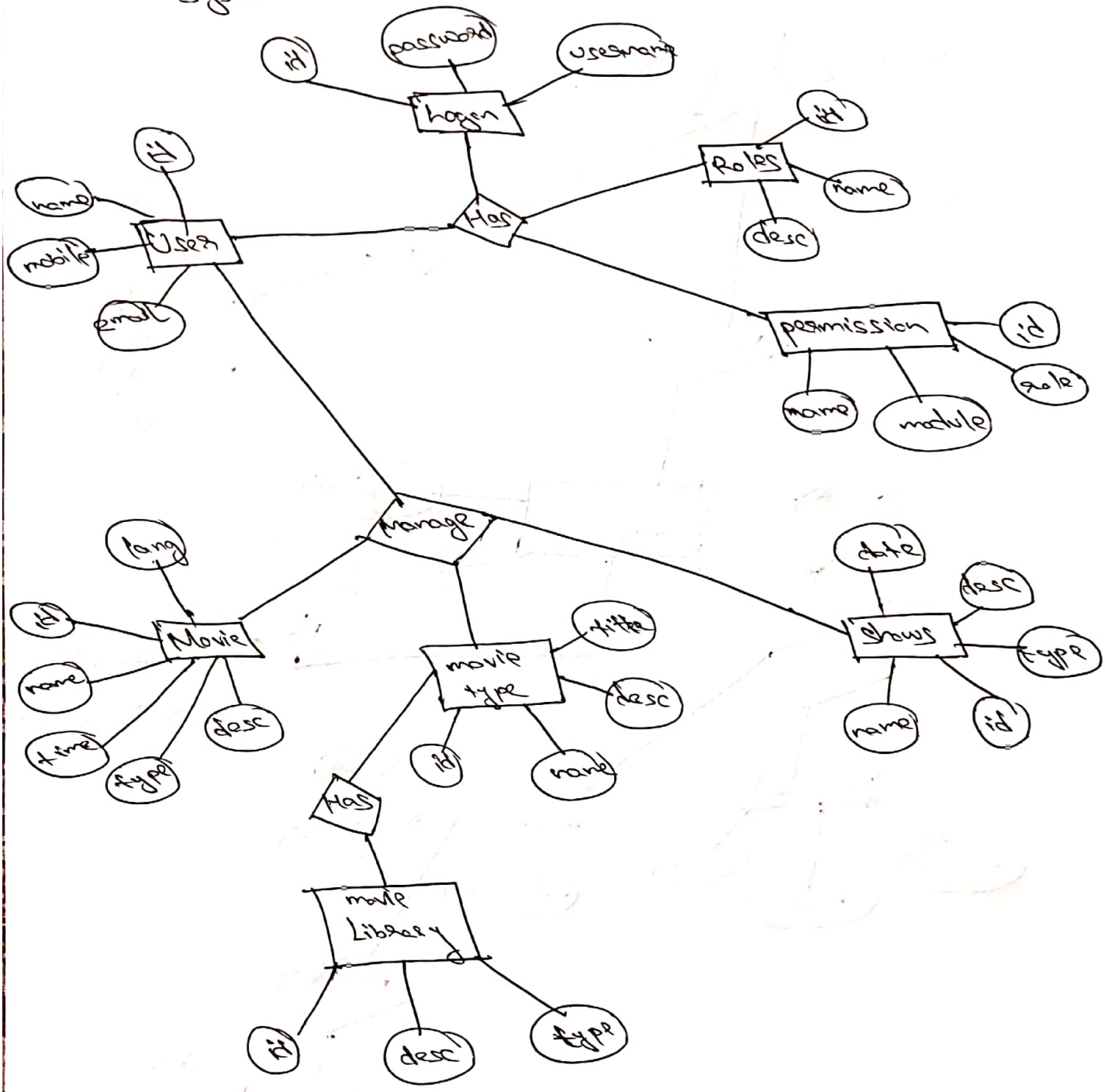


Develop ER diagram for ATM Management System.

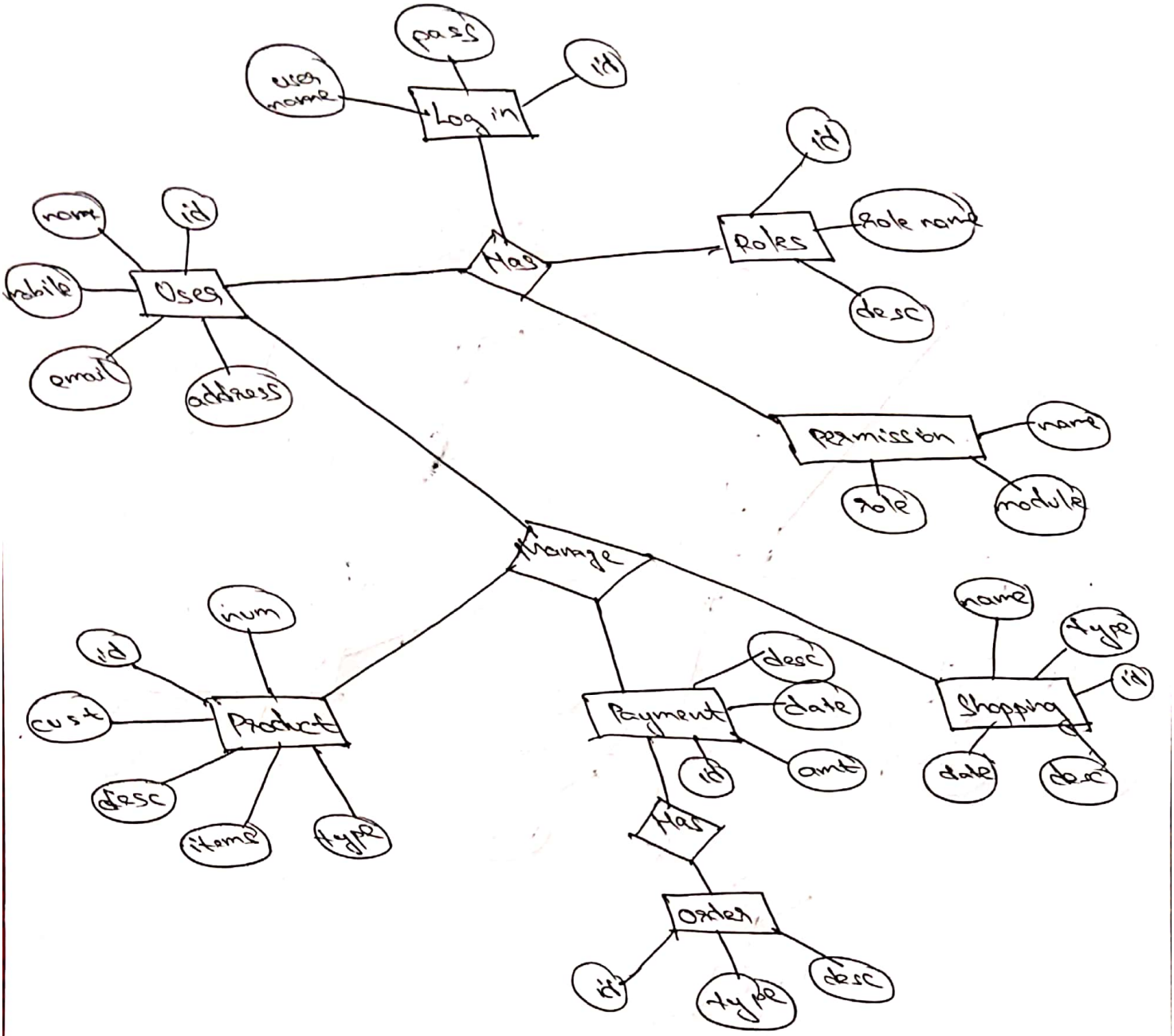


Pg ③

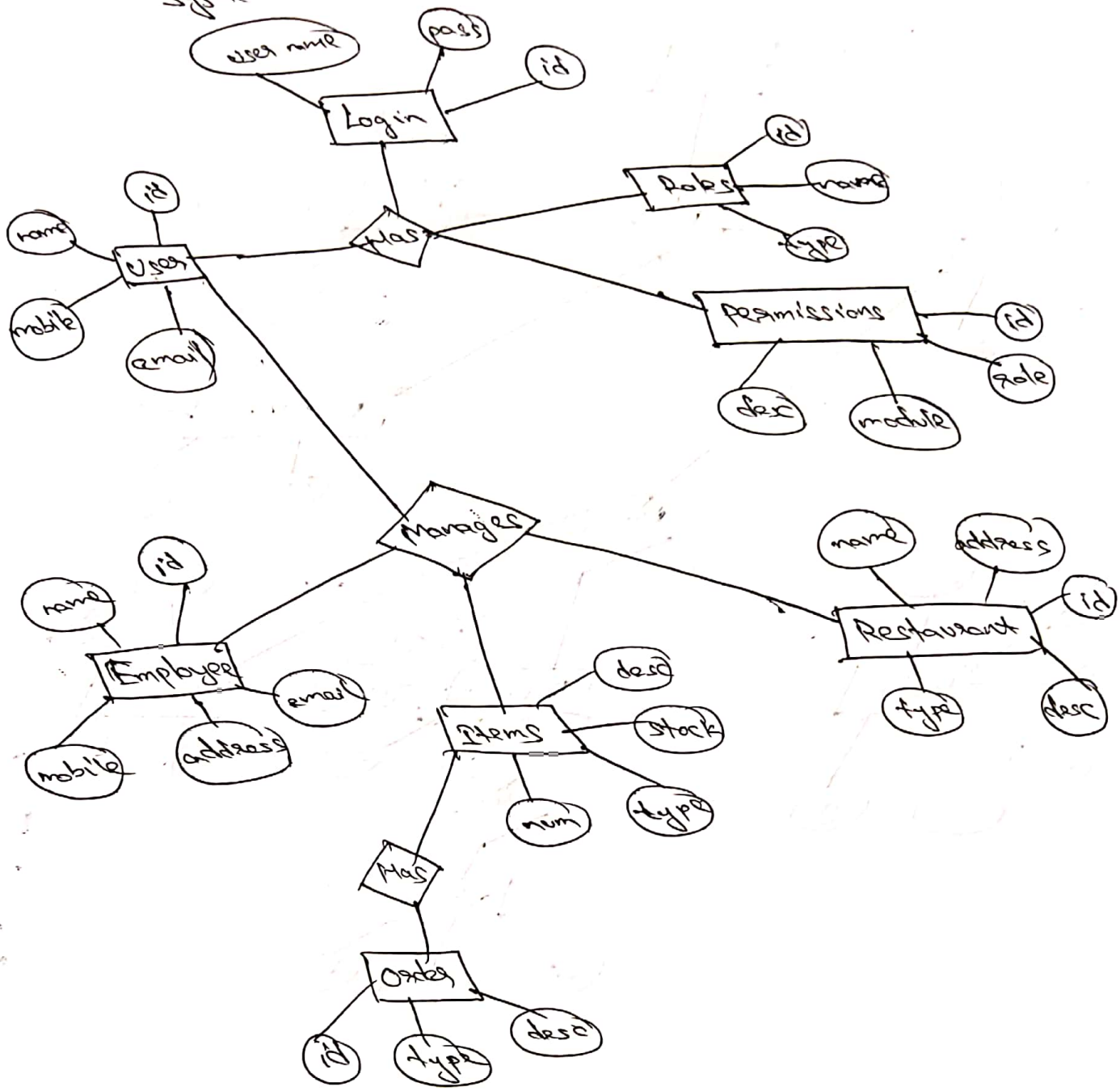
Q) Develop an ER diagram for Movie Ashis System.



Q Develop ER diagram for Online Shopping system.



6) Develop ER Diagram for Restaurant Management System.



Assignment - 6

1. Develop a JDBC application that connects to Oracle DBMS.

→
```
import java.sql.*;
class CA {
public static void main(String args[]) throws
ClassNotFoundException, SQLException
{
Class.forName("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection("jdbc:
oracle:thin:@localhost:1521:XE","system","oracle");
// SERVER, USERNAME, PASSWORD
System.out.println("connection established");
}
}
```

Output

Connection established.

2. Develop a JDBC application to store two account's information (acc no, account holder name and balance) into the database.

→ Code

```
import java.sql.*;
class q2 {
  public static void main (String args[]) throws
ClassNotFoundException, SQLException {
Class.forName ("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection ("jdbc:
oracle:thin:@localhost:1521:xE", "system", "oracle");
Statement st = c.createStatement ();
int n = st.executeUpdate ("insert into Account
values (1111, 'S', 50000)");
n = st.executeUpdate ("insert into Account
values (1112, 'K', 1500)");
st.close ();
c.close ();
if (n > 0) {
    System.out.println ("Inserted Successfully");
}
else {
System.out.println ("Not Inserted ... Try again!");
}
st.close ();
c.close ();
}
}
```

Output

Inserted Successfully.

3. Develop a JDBC application that increases the balance of all account by Rupees 500.

→ Code

```
Import java.sql.*;
class q3 {
 public static void main (String args[ ]) throws
 ClassNotfoundException, SQLException {
Class.forName ("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection ("jdbc:
oracle:thin:@localhost:1521:xE", "system",
"oracle");
Statement st = c.createStatement ( );
int n = st.executeUpdate ("UPDATE Account SET
BALANCE = BALANCE + 500");
if (n>1)
{ system.out.println ("Updated successfully");
}
 st.close ( );
 c.close ( );
 }
}
```

Output

Updated successfully.

4. Develop a JDBC application that delete all the account whose balance is less than 10 000.

→ Code

```
import java.sql.*;
class q4 {
public static void main (String args[]) throws
ClassNotFoundException, SQLException {
Class.forName ("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection ("jdbc:
oracle:thin:@localhost:1521:XE","system",
"oracle");
int n = st.executeUpdate ("DELETE FROM ACCOUNT
WHERE BALANCE<10000");
System.out.println ("Database updated!!!");
st.close ();
c.close ();
}
}
```

Output

Database updated!!!

5. Develop a JDBC application that prompt the end-user to enter account detail and stores the same into the data-base.

→ Code

```
import java.util.scanner;
class q5 {
public static void main (string args[ ]) throws
Class Not found Exception, SQLException {
Scanner s = new Scanner ( System.in );
System.out.println (" enter the A/c holder name ");
string nm = s.next ( );
system.out.println (" enter the account number");
int acno = s.nextInt ( );
system.out.println (" Enter the balance RS");
float bal = s.nextfloat ( );
Class.forname (" oracle.jdbc.driver.oracle Driver");
Connection c = DriverManager.getConnection (" jdbc:
oracle:thin:@localhost:1521:xE","system","oracle");
statement st = c.create statement ( );
int n = st.executeUpdate (" Insert into Account
values (" + acno + "," + nm + "," + bal + ")" );
system.out.println (n +" account stored successfully");
  st.close ( );
  c.close ( );
 }
}
```

Assignment - 7

1) Develop a JDBC application that prompts the end-user to enter the A/c no. & displays the account details. If account does not exist with that account number then display error message.

```
import java.sql.*;
import java.util.Scanner;
class Account
{
public static void main(String args[]) throws
ClassNotfound Exception, SQL Exception
{
Scanner s = new Scanner(System.in);
System.out.println("Enter the A/c number:");
  int accno = s.nextInt();

Class.forName("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection("jdbc:oracle:thin@
localhost:1521:XE", "system", "oracle");

Statement st = c.createStatement();
ResultSet rs = st.executeQuery("SELECT * from
      Account where AccNo=" +accno);

  if (rs.next()) {
String name = rs.getString(2);
  int balance = rs.getInt(3);
System.out.println("Account holder's name:" +name);
  System.out.println("Account Balance:Rs." + balance);

}
else {
```

```
System.out.println ("Not a Valid information");
}
rs.close();
st. close();
c .close();
}
}
```

8) Develop a JDBC Application that receives name and balance of all the accounts whose balance is more than Rs 5000.

```
import java.sql.*;
import java.util.Scanner;
class Account
{
public static void main (String args[]) throws
ClassNotFoundException, SQLException {
Class.forName ("oracle.jdbc.OracleDriver");
Connection c= DriverManager. getConnection ("jdbc:oracle:
thin:@localhost:1521:XE", "system", "oracle");
Statement st = c.createStatement();
ResultSet rs=st.executeQuery ("SELECT * from Account
                               where balance >5000");
while (rs.next ()){
int accountNo = rs.getInt(1);
String name = rs.getString(2);
int balance =rs.getInt(3);
System.out.println ("\nAccount Number:" + accountNo);
System.out.println ("Account holder's name: " + name);
System.out.println ("Account Balance: Rs" + balance);
System.out.println ("\n");
```

```
rs.close();
st.close();
c.close();
}
}
```

2) Develop a JDBC application in which two accounts information is stored into the database using prepared statement.

```
import java.sql.*;
import java.util.Scanner;
class Account {
public static void main(String args[]) throws
ClassNotFoundException, SQLException {
Class.forName("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection("jdbc:oracle:
thin:@localhost:1521:XE", "system", "oracle");

PreparedStatement ps = c.prepareStatement("Insert
Into Account Values (?,?,?)");

ps.setInt(1,1009);
ps.setString(2,"SURESH");
ps.setInt(3,9800);
n= ps.executeUpdate();

System.out.println("\nInformation Stored.");
ps.close();
c.close();

}}
```

Pg-④

4) Develop a JDBC code to implement the use case of balance ~~query~~ enquiry.

```
import java.sql.*;
import java.util.Scanner;
class Account{
  public static void main(String args[]) throws
    ClassNotFoundException, SQLException {
    Scanner s = new Scanner(System.in);
    System.out.println("Enter the A/c number:");
    int accno = s.nextInt();
    Class.forName("oracle.jdbc.OracleDriver");
    Connection c = DriverManager.getConnection("jdbc:oracle:
thin@~~oracle~~localhost:1521:XE", "system", "oracle");
    Statement st = c.createStatement();
    ResultSet rs = st.executeQuery("SELECT * from
      Account where ACCNO = "+accno);
    if(rs.next()){
    String name = rs.getString(2);
    int balance = rs.getInt(3);
    System.out.println("Account holder's name:"+name);
    System.out.println("Account Balance:Rs"+balance);
    }
    else{
    System.out.println("Not a valid information");
    }rs.close();
     st.close();
     c.close();
  }
}
```

Q) Develop a JDBC code to implement the use case of closing the account.

```
import java.sql.*;
import java.util.Scanner;
class Account {
public static void main(String args[]) throws ClassNotFound
Exception, SQLException {
Scanner s=new Scanner(System.in);
System.out.println("Enter the A/C no:");
int acno = s.nextInt();
Class.forName("oracle.jdbc.OracleDriver");
Connection c = DriverManager.getConnection("jdbc:
oracle:thin:@localhost:1521:XE", "system", "oracle");
Statement st = c.createStatement();
ResultSet rs = st.executeQuery("SELECT * from
Account where AccNo = "+acno);
if(rs.next()) {
String name = rs.getString(2);
int balance = rs.getInt(3);
System.out.println("Account Holder's name:"+name);
System.out.println("Account Balance: Rs."+balance);
}
else {
System.out.println("Not a valid information");
}
System.out.println("Do you want to terminate
your account? [Y/N]\n");
char check = s.next().charAt(0);
```

```
if (check == 'Y') {
    int n = st.executeUpdate ("Delete from Account where
                               Acc no = " + accno);
    System.out.println("Your account has been terminated");
}
else
{
System.out.println("Thank you");
}
rs.close();
st.close();
c.close();
}
}
```

Assignment-8

1) Develop a JDBC Application in which fund transfers user case is implemented and data is safeguard using transaction management.

→ Code

```
import java.sql.*;
class q1 {
    Public static void main(String[] args)
        throws ClassNotFoundException, SQLException
    {
String QUERY = "Select * from Account where
                acc_no = 1111";
String QUERY1 = "select * from Account where
                acc_no = 1112";
Boolean autoCommit;
String update_query = "update Account set
                balance = 41000 where
                acc_no = 1111";
String update_query1 = "update Account set
                balance = 42000 where acc_no
                = 1112";
Class.forName(className: "oracle.jdbc.OracleDriver")
Connection c = DriverManager.getConnection(url: "jdbc:
                Oracle:thin:@localhost:1521:xe",
```

User: "system", Password: "oracle");

Statement st = c.createStatement();

ResultSet rs1 = null;

rs1 = st.executeQuery(QUERY);

System.out.println(x: "Getting the data from Account table ...");

displayData(rs1);

System.out.println(x: "Setting the AutoCommit value as FALSE");

c.setAutoCommit(autoCommit: false);

autoCommit = c.getAutoCommit(autoCommit: false);

System.out.println("AutoCommit value of the Connection = " + autocommit);

Statement st1 = c.createStatement();

System.out.println(x: "Executing update query to update salary of acc_no = 1111");

System.out.println("Update Query is" + update_query);

int return_rows = st1.executeUpdate(update_query);

System.out.println(x: "Update the data but didn't commit");

Connection conn1 = DriverManager.getConnection(url: "jdbc:oracle:thin:@localhost:1521:XE", user: "system", password: "oracle");

```
User: "system", password: "oracle");
System.out.println(x: "opening new connection");
System.out.println(x: "acc_no = 1111 data");
Statement statement2 = conn1.createStatement();
ResultSet rs;
rs = statement2.executeQuery(QUERY);
displayData(rs);
System.out.println(x: "commit has been done");
c.commit();
Savepoint s1 = c.setSavepoint();
System.out.println(x: "Savepoint has been created");
System.out.println(x: "Displaying data of acc.num
                       = 1111");

System.out.println(x: "using the second connection");
rs = statement2.executeQuery(QUERY);
displayData(rs);
rs = st1.executeQuery(QUERY);
System.out.println(x: "Data of acc_no = 1112 but didn't
                       commit");
rs1 = st1.executeQuery(QUERY1);
displayData(rs1);
```

```
System.out.println("Rollback is done... So updated
                    data won't be reflected");

c.rollback(s1);
System.out.println("Data of acc_no = 1112 after
                    Rollback till the last savepoint");

rs1 = st1.executeQuery(QUERY1);
displayData(rs1);
}
Public static void displayData(ResultSet rs1) throws
                                SQLException {

    while(rs1.next()) {
        int acc_no = rs1.getInt(columnLabel: "acc_no");
        String Acc_holder_name = rs1.getString
                    (columnLabel: "Acc_holder_name");
        float balance = rs1.getFloat(columnLabel: "balance");
        System.out.println(acc_no + "," + Acc_holder_name
                    + "," + balance);

    }

}
```

Output

Getting the data from Account table...

1111, S, 41000.0

1111, S, 41000.0

1111, S, 41000.0

Setting the AutoCommit value as FALSE

AutoCommit value of the connection = false

Executing update query to update salary of acc_no = 1111

Update Query is update Account set balance = 41000

Where acc_no = 1111.

Updated the data but didn't commit

Opening new connection

acc_no = 1111 data

1111, S, 41000.0

1111, S, 41000.0

1111, S, 41000.0

Commit has been done

Savepoint has been created

Displaying data of acc_num = 1111.

Using the second connection

1111, S, 41000.0

1111, S, 41000.0

1111, S, 41000.0

Data of acc_no = 1112.

1112, K, 3000.0
1112, K, 3000.0
1112, K, 1500.0

Updating the salary of acc_no = 1112

Update query is update Account set balance = 42000 where acc_no = 1112

Data of acc_no = 1112 but didn't commit

1112, K, 42000.0
1112, K, 42000.0
1112, K, 42000.0

Rollback is done... So updated data won't be reflected.

Data of acc_no = 1112 after Rollback till the last savepoint.

1112, K, 3000.0
1112, K, 3000.0
1112, K, 1500.0.

9) Develop a JDBC Application that prompts the end-user to perform the following operations.

- Store accounts' information (acc no, accounts holders name and balance) into the database.
- Implement the use case of closing the account.
- Increases the balance of a particular account.
- Retrieve the data from the database based on the particular account numbers.

→ Code:-

```
import java.sql.*;
import java.util.*;

class q2{
    public static void main(String args[]) throws
ClassNotFoundException, SQLException {
    Scanner sc = new Scanner(System.in);
    System.out.println("1. Store Datain 2. Delete
Account In 3. Increase Balance In 4. Retrieve Datain");
```

```
int choice = sc.nextInt();
Connection c = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe",
"system", "oracle");
class.forName("oracle.jdbc.OracleDriver");
Switch (choice) {
    case 1:
        System.out.println("Enter Account Number");
    int acc_no = sc.nextInt();
    System.out.println("Enter Account-Holder Name");
    String acc_name = sc.next();
    System.out.println("Enter Balance Rs: ");
    float bal = sc.nextFloat();
Statement st = c.createStatement();
int n = st.executeUpdate("Insert into accounts
values (" + acc_no + ",'" + acc_name + "'," +
bal + ")");
System.out.println("Inserted succesfully");
    st.close();
    break;
```

Case 2:

```
System.out.println("Enter Account Number
TO Delete The Account: ");
int acc_no1 = sc.nextInt();
Statement st1 = c.createStatement();
int p = st1.executeUpdate("Delete from
accounts where acc_no =" + acc_no1);
System.out.println("Account Deleted
Successfully with acc_no: " + acc_no1);
st1.close();

break;
```

Case 3:

```
System.out.println("Enter Account Number to
Increase Balance: ");
int acc_no2 = sc.nextInt();
System.out.println("Enter Amount to
add: ");
float balance = sc.nextFloat();
Statement st3 = c.createStatement();
int q = st3.executeUpdate("Update account
set balance = balance +" balance + " where
acc_no =" + acc_no2);
```

```
System.out.println("Updated Successfully!!!!!!");
st3.close();
break;

case 4:
    System.out.println("Enter Account Number
    to view Details: ");
    int acc_no3 = sc.nextInt();
    Statement st4 = c.createStatement();
    ResultSet r = st4.executeQuery("select *
    from accounts where acc_no = "+acc_no3);
    int count = 0;
    while (r.next()) {
        String acc_no4 = r.getString("acc_no");
        String acc_name4 = r.getString("acc_name");
        String balance4 = r.getString("balance");
    System.out.println("\n Account Number: \n"
+acc_no4 + "\n Account Holder's Name! \n"
+acc_name4 + "\n Balance: \n" + balance4);
```

```
count++;
}
if (count==0){
    System.out.println("There is no account
with the account number: "+accno3);
}
System.out.println("u viewed succesfully!!!");
    stu.close();
    c.close();
    break;
  }
}
}
```

Output

1. Store Data
2. Delete Account
3. Increase Balance.
4. Retrieve Data.

1↵
Enter Account Number:
121
Enter Account Holder Name:
Samasree
Enter Balance Rs:
400000
Inserted Successfully.

3↵
Enter Account Number to Increase Balance:
121
Enter Amount to add:
500
Updated Successfully!!!!!!

4↵
Enter Account Number to view Details: 121

Account Number:
121

Account Holder's Name:
Saravanan

Balance:
400500

Viewd successfully!!!

2↵

Enter Account Number To Delete the Account:

121

Account Deleted Succesfully with ace_no: 121.

4↵

Enter Account Number To view Details:

121

There is no account with account number: 121