



**PES University, Bengaluru**

(Established under Karnataka Act 16 of 2013)

**Department of Computer Science & Engineering**  
**Session: Jan - May 2022**

**UE19CS353 – Object Oriented Analysis and Design with Java**  
**Theory ISA (Mini Project)**

Report on

**ONLINE BANKING SYSTEM**

**By:**

**RAJESHWARI R – PES1UG19CS375**

**RUCHITA V R – PES1UG19CS397**

**PRAKRUTI P – PES1UG19CS338**

**6<sup>th</sup> Semester F**

## **1. Project Description**

Banking system is a software aimed to automate all the functioning of a bank both at the customer's and bank authorities' side.

- The banking system offers the customers solutions like e-payment (paying electricity bill, recharging, etc), e-shopping (paying for purchases done on e-commerce) and e-banking (transfer of funds to other accounts). It allows the account holders to conduct financial transactions via the internet.
- The customer doesn't need to visit the bank in order to complete most of their basic banking transactions. They only need a device, an internet connection and a bank account in the bank to carry out all the transactions.

- If the customer does not already have an account in the bank, they can also create one using the online banking system. On creating an account, they can login whenever they want to check for the details or carry out any transactions.
- On creating an account, the customer is provided with the credentials in order to activate the netbanking feature. They can login using their ID and password and if they fail to login after 2 attempts then the account will be deactivated temporarily for 24 hours.
- Customer can set and update his profile information like email id, mobile number using this system. They can also change his/ her password.
- Admin can create additional bank administrators and control user access with permissions.
- Customers can view the transaction summary for the required period (week or month or year). They can also transfer money to other accounts and use net-banking credentials for e-payment.
- Online banking system also provides some features for the customers like applying for the ATM card, loan, and creating the fixed deposits.
- Customers can visit the bank's website and apply for an ATM (debit/ Credit) Card in the personal banking/ retail banking section.
- Once a customer has submitted the details and the required documents (either aadhar card or pan card or passport or driving licence or voter id card), they need to wait for 2-3 days before receiving the card.
- Along with the card, customers will also receive a set of confidential information, like Personal Identification Number or PIN. This PIN is to be used for all transaction and payment purposes. Once you have the card and the PIN, you can transact using your card anytime you want.
- The customer can check the account status or transfer the amount to anyone using the receiver's account number and some other bank details. They can also view their transaction summary of all the credits and debits they have made so far.
- Banking system is designed to facilitate bank authorities to perform all the functions with speed and accuracy. Once a customer completes the procedure required to open an account using the online banking system, he/she approaches the local bank branch with all the relevant documents.
- Bank authorities use the banking system to activate his/her account. Customers are also given the net banking credentials. The Bank authorities can update the customer balance when withdrawals and deposits are made in the local branch. System is also enabled with features to process cheques and demand drafts.
- Bank manager is responsible for managing the team of product experts, tellers, and other bank executives to provide outstanding customer service and also managing resources and staff, developing and attaining sales goals, and delivering customer service.
- For those who do not have online banking must go to their respective bank branch for all the transactions. When a customer goes to the bank to withdraw cash, the bank employee verifies the account and instructs the cashier to dispose of the cash. Once the cash is disposed of, the customer's balance is updated.

### **a. Link to Github repository**

<https://github.com/Ruchitavr/OOAJ-Project-Online-banking-system-.git>

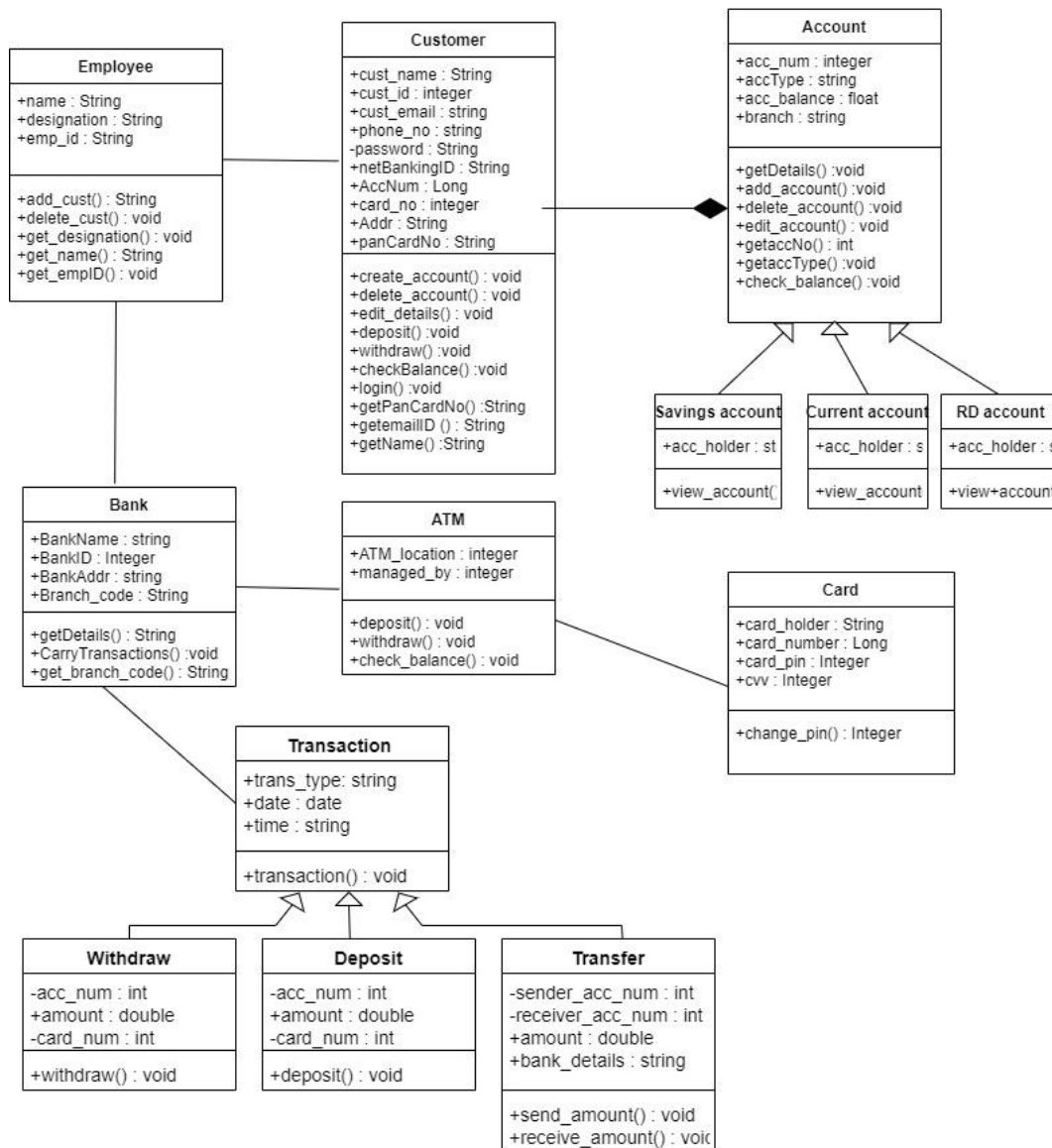
## 2. Analysis and Design Models

USE CASE DIAGRAM:

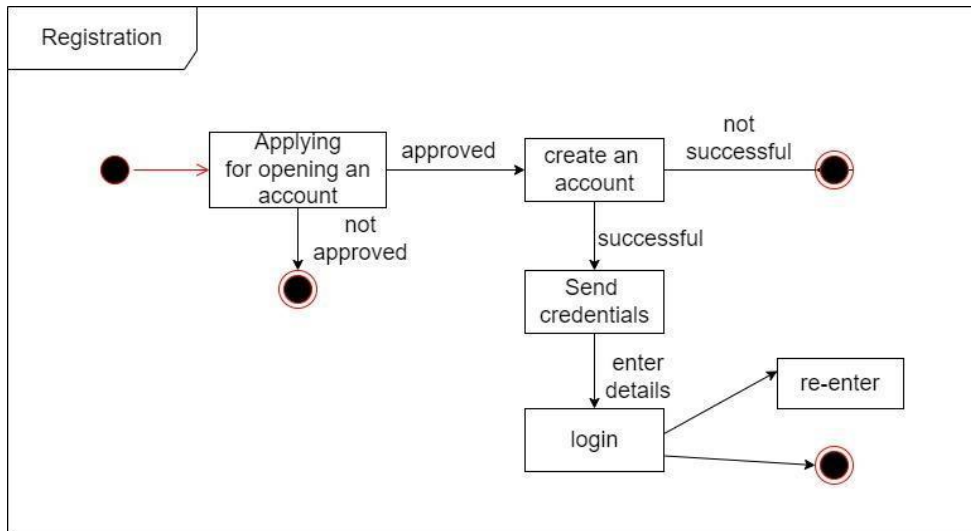


CLASS DIAGRAM:

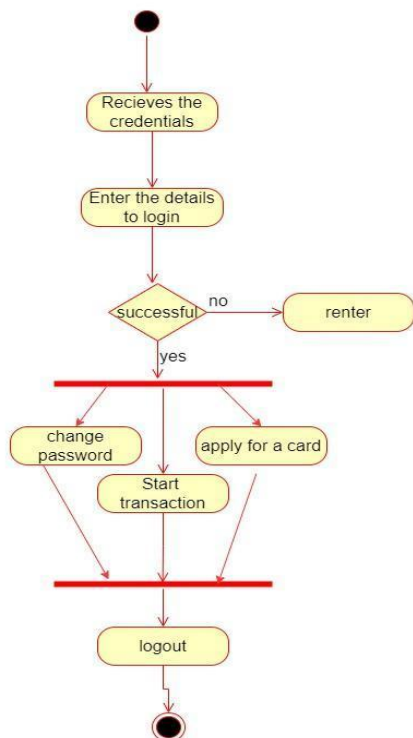
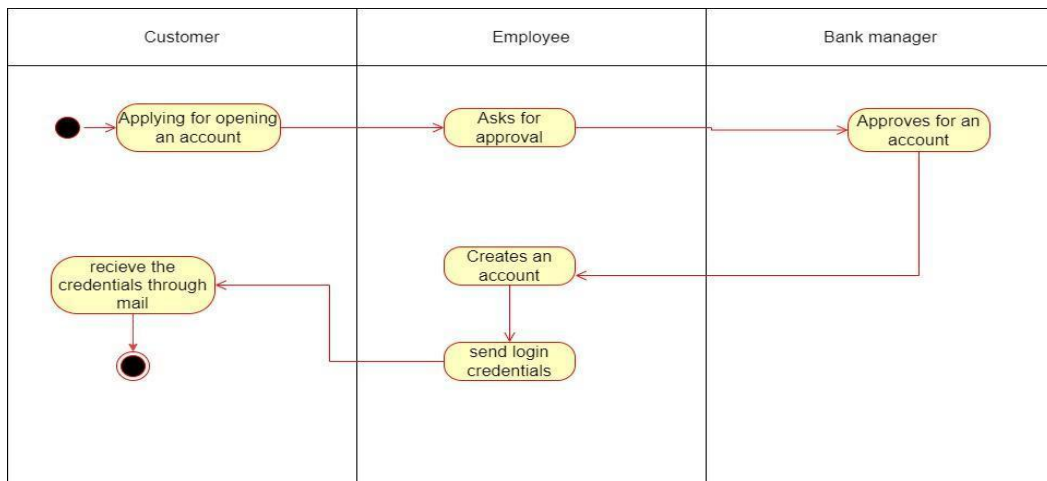
## Online Banking system



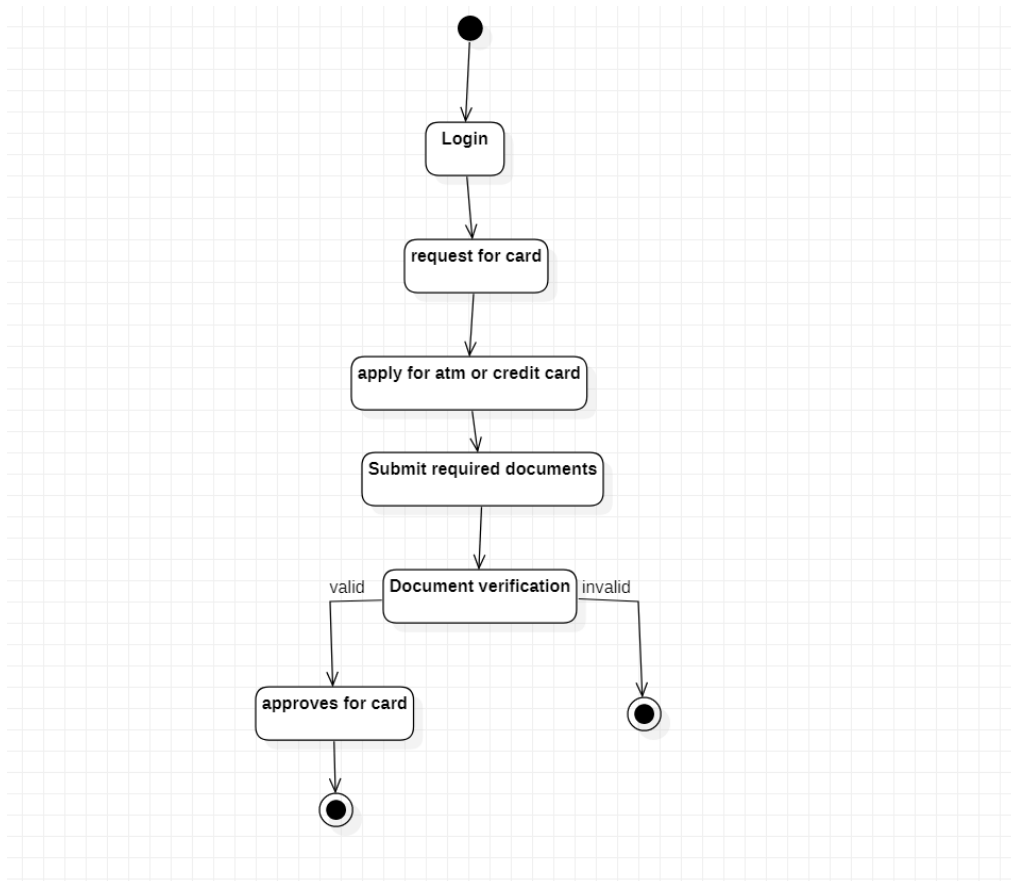
STATE DIAGRAM FOR REGISTRATION:



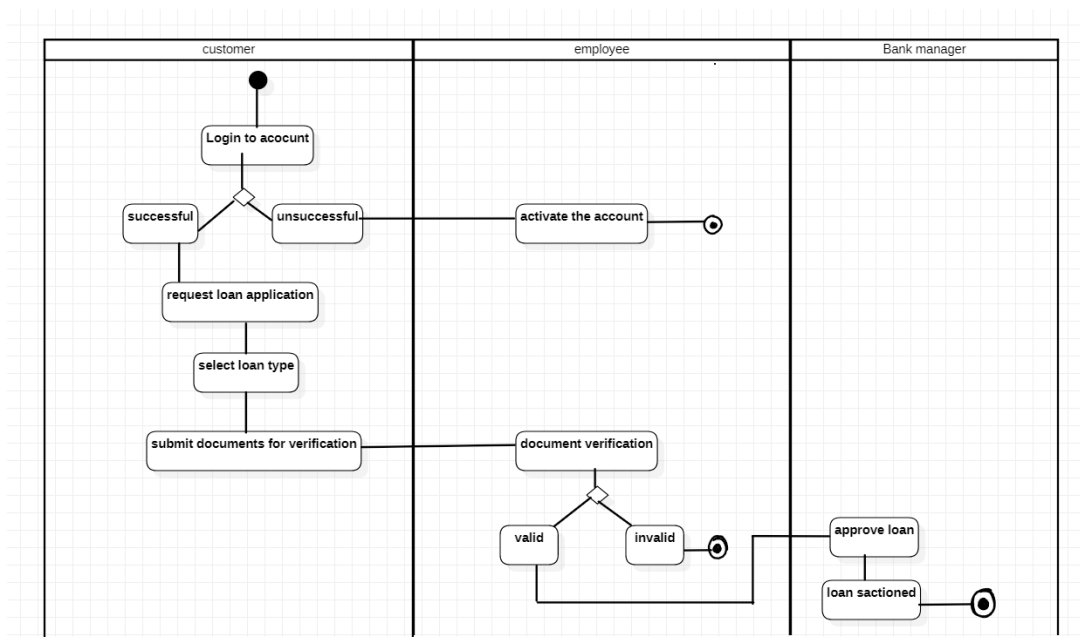
ACTIVITY DIAGRAM FOR REGISTRATION:



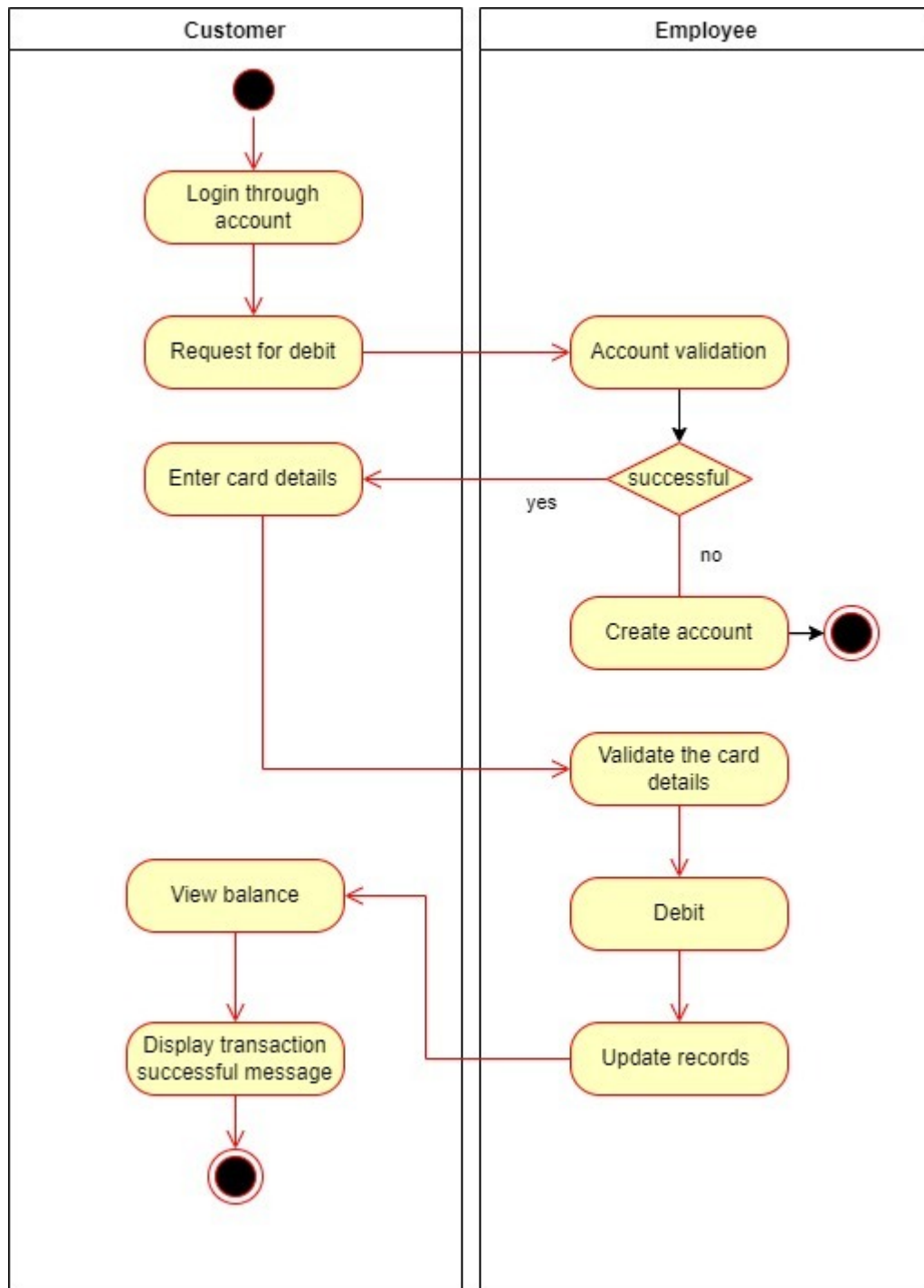
STATE DIAGRAM FOR APPLICATION OF ATM CARD:



## ACTIVITY DIAGRAM FOR ATM CARD APPLICATION:

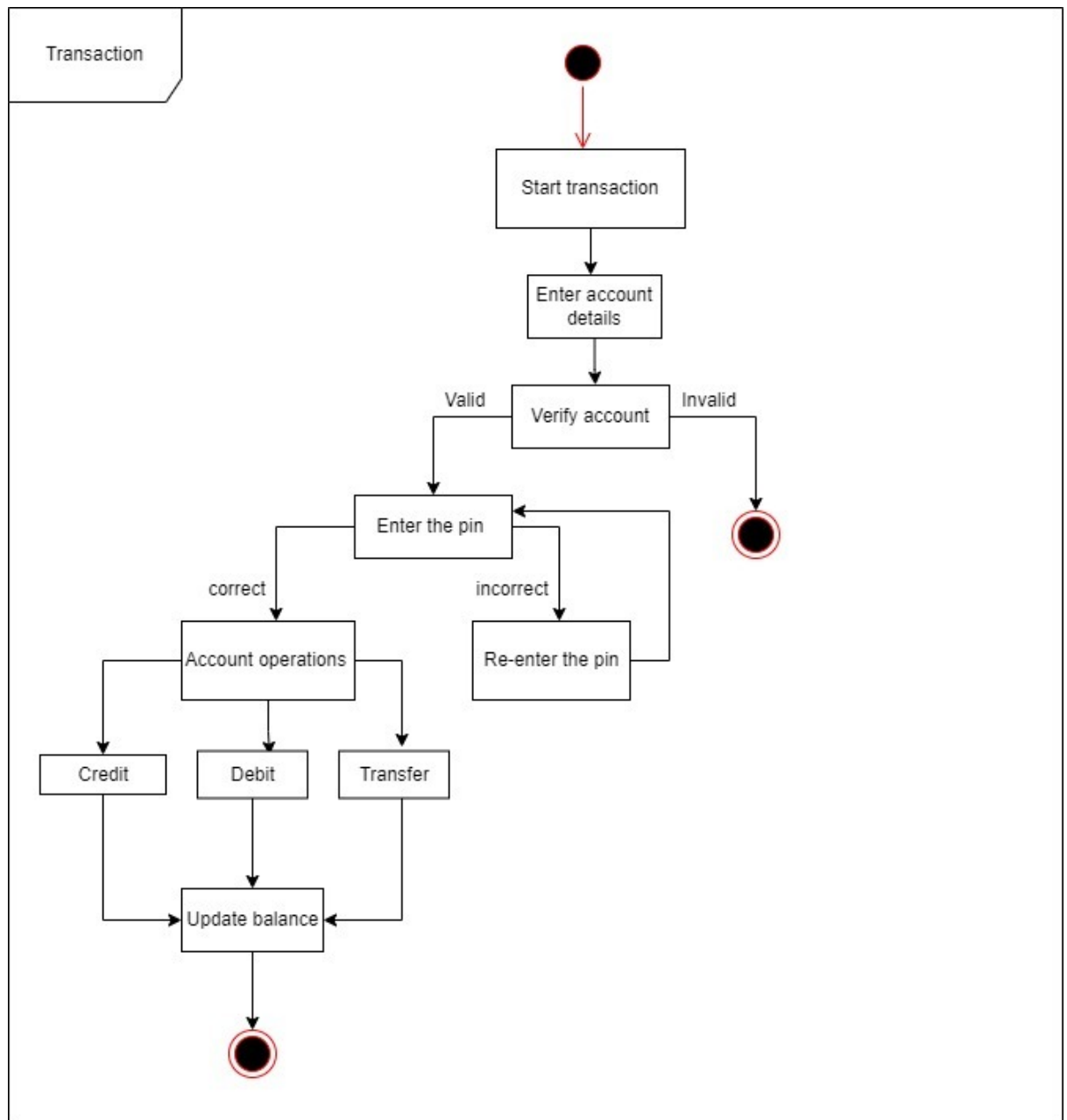


## ACTIVITY DIAGRAM FOR TRANSACTIONS:





## STATE DIAGRAM FOR TRANSACTION:



### 3. Tools and Frameworks Used

The tool used for our project is Java Netbeans and the framework used to create the GUI is Swing and JDBC for connecting to the database.

#### 4. Design Principles and Design Patterns Applied

<<Explain where in the application these were applied>>

The design pattern that we adopted for our project is a factory pattern.

```
public class AccountFactory {  
    public AccountFactory()  
    {  
    }  
    public Account getAccount(String s1)  
    {  
        if (s1.equals("CurrentAccount"))  
            return new currentAccount();  
        else if(s1.equals("Savings Account"))  
            return new savingsAccount();  
        else  
            return new RDAccount();  
    }  
}
```

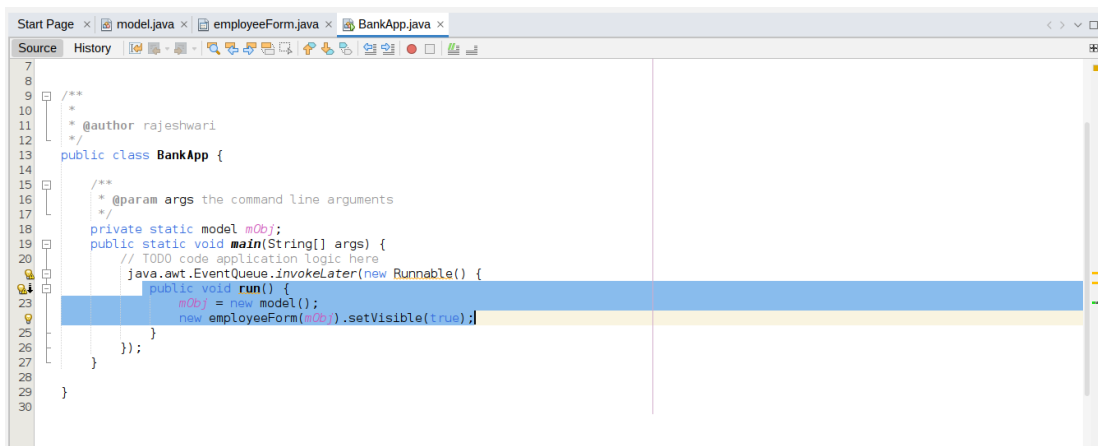
There are three different accounts for now and each of them have different functionality and hence Factory design pattern has been used to implement creation of account.

The principles that were kept in line were :

**controller** :

We have used MVC architecture to implement independent module which is operated by the employee of the

Bank for creating an account for the customers. Here, the controller is connecting the model and view.



```
7
8
9 /**
10  * @author rajeshwari
11  */
12
13 public class BankApp {
14
15     /**
16      * @param args the command line arguments
17      */
18     private static model mObj;
19     public static void main(String[] args) {
20         // TODO code application logic here
21         java.awt.EventQueue.invokeLater(new Runnable() {
22             public void run() {
23                 mObj = new model();
24                 new employeeForm(mObj).setVisible(true);
25             }
26         });
27     }
28
29 }
30
```

### low coupling.

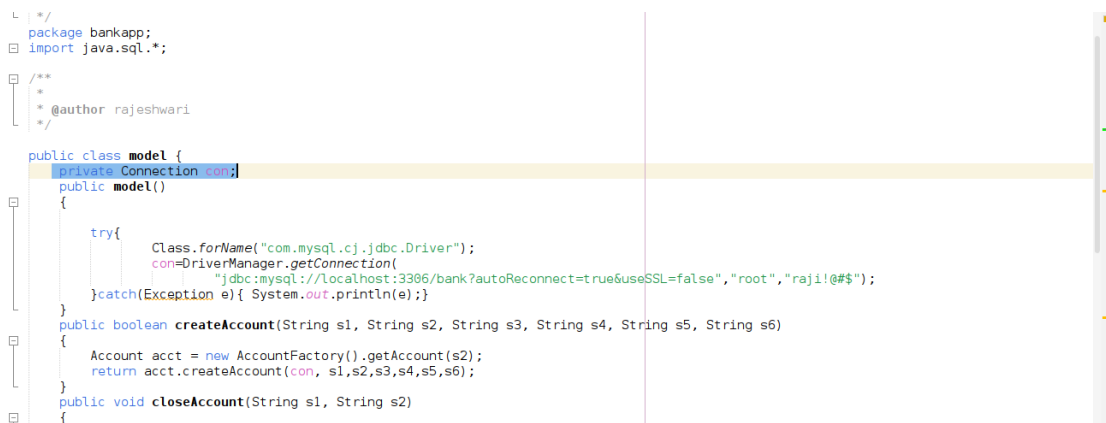
Low coupling design has been implemented by creating two different modules :

employee module and customer module and both function independent of each other.

### high cohesion.

We have created a single connection object for the entire module.

This connection object **con** is used in the entire employee module.



```
1 /**
2  *
3  */
4 package bankapp;
5 import java.sql.*;
6
7 /**
8  * @author rajeshwari
9  */
10
11 public class model {
12     private Connection con;
13     public model() {
14         try {
15             Class.forName("com.mysql.cj.jdbc.Driver");
16             con=DriverManager.getConnection(
17                 "jdbc:mysql://localhost:3306/bank?autoReconnect=true&useSSL=false", "root", "rajil!@#$");
18         } catch (Exception e) { System.out.println(e); }
19     }
20     public boolean createAccount(String s1, String s2, String s3, String s4, String s5, String s6) {
21         Account acct = new AccountFactory().getAccount(s2);
22         return acct.createAccount(con, s1,s2,s3,s4,s5,s6);
23     }
24     public void closeAccount(String s1, String s2) {
25     }
26 }
```

This connection object **cobj** is used in the entire customer module.

```
Start Page x model.java x employeeForm.java x BankApp.java x firstPanel.java x
Source Design History
19
20
21 /**
22  * Creates new form firstPanel
23  */
24 public static String g_actnum;
25 public static Connection cobj;
26
27 public firstPanel() {
28     initComponents();
29     try
30     {
31         Class.forName("com.mysql.cj.jdbc.Driver");
32         cobj=DriverManager.getConnection("jdbc:mysql://localhost:3306/bank?autoReconnect=true&useSSL=false","root","raji!@#$");
33     }catch(Exception e){ System.out.println(e);return;}
34
35 /**
36  * This method is called from within the constructor to initialize the form.
37  * WARNING: Do NOT modify this code. The content of this method is always
38  * regenerated by the Form Editor.
39  */
40 @SuppressWarnings("unchecked")
41 [Generated Code]
113
114 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
115     // TODO add your handling code here:
116     homePage.sIn.setVisible(true);
```

## 5. Application Screenshots (3-4 important pages)

### Employee module :

#### 1) Entering the details to create an account for the customer

The screenshot shows a Java Swing window titled "Banking App - Employee Module". Inside, there's a dialog box titled "Create Customer Account". The dialog has a "Create" button and a "Close" button at the top left. It contains several input fields: "Account Number" with the value "123456799", "Name" with "Rajeshwari", "Account Type" with a dropdown menu showing "Savings Account", "EmailId" with "raji@gmail.com", "Opening Balance" with "1000", and "Bank Branch" with "RR Nagar". At the bottom, there are three buttons: "Confirm", "Clear", and "Exit". Below the dialog box, there's an "Output - BankApp (run)" window.

2) Successfully creating the account with default loginid and password

3) Database view showing that the details have been stored in the customer table.

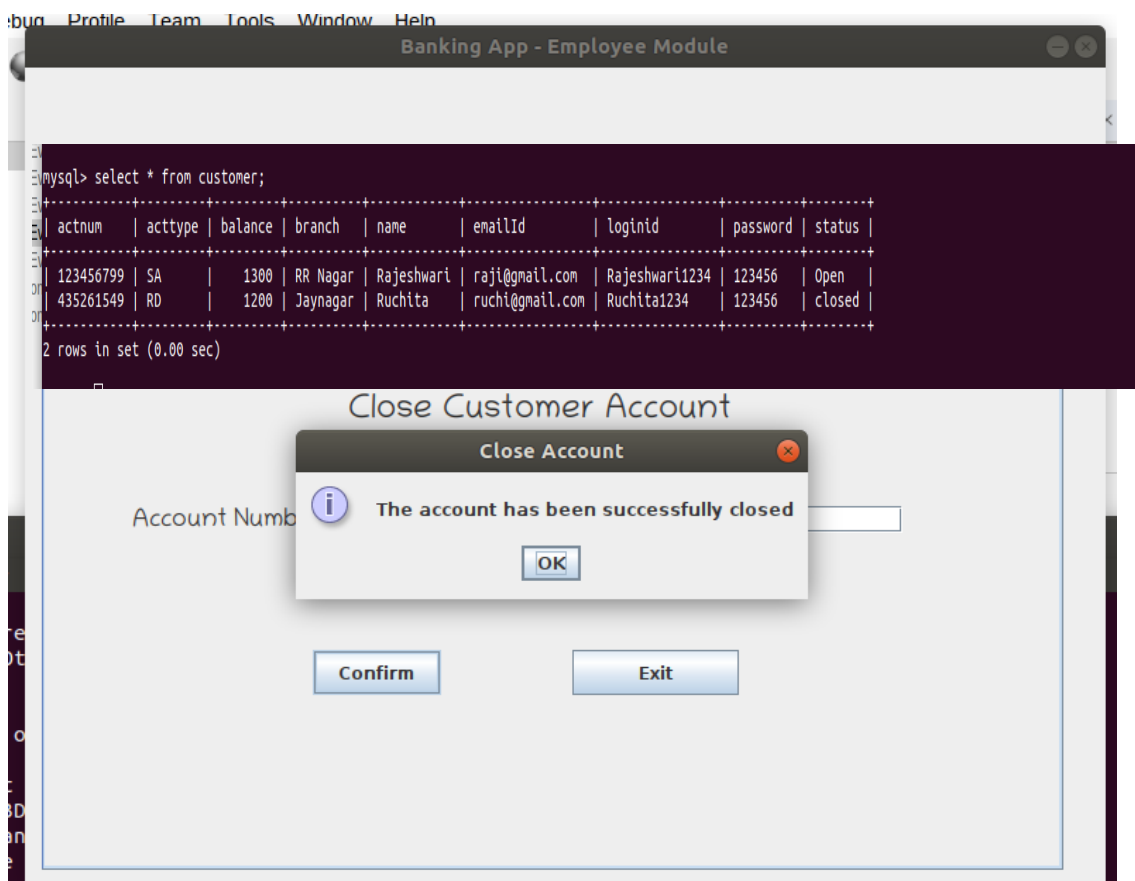
```
mysql> select * from customer;
```

actnum	acttype	balance	branch	name	emailId	loginid	password	status
123456799	SA	1000	RR Nagar	Rajeshwari	raji@gmail.com	Rajeshwari1234	123456	Open

```
1 row in set (0.00 sec)
```

```
mysql>
```

4)closing the account



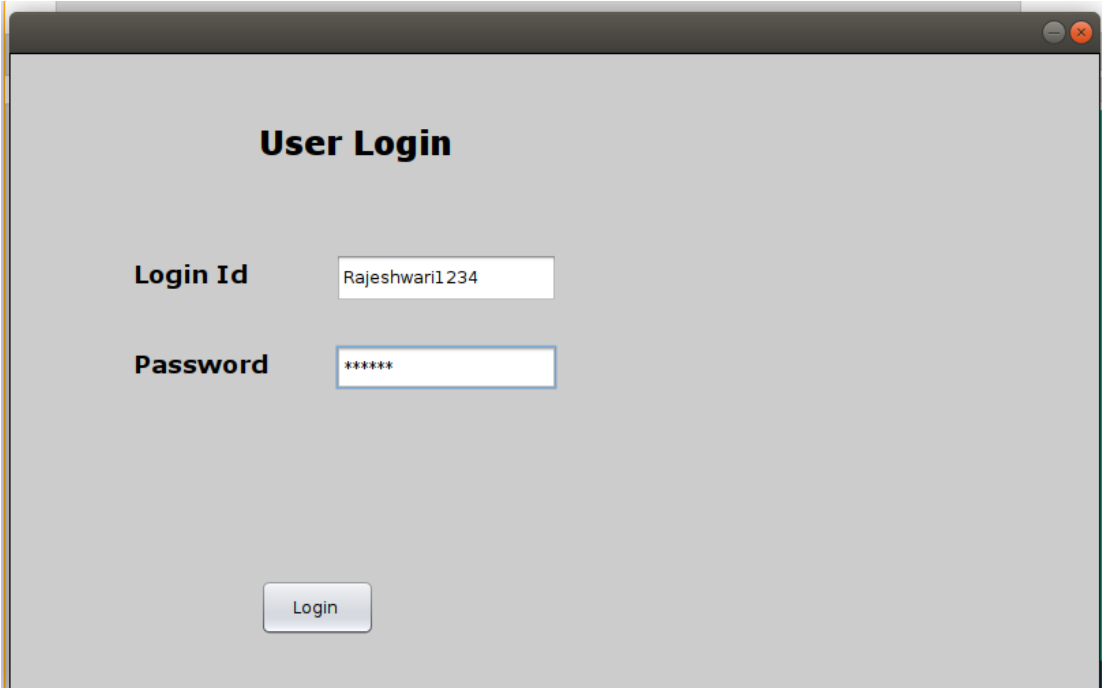
5) Database view of the account closed

Customer Module

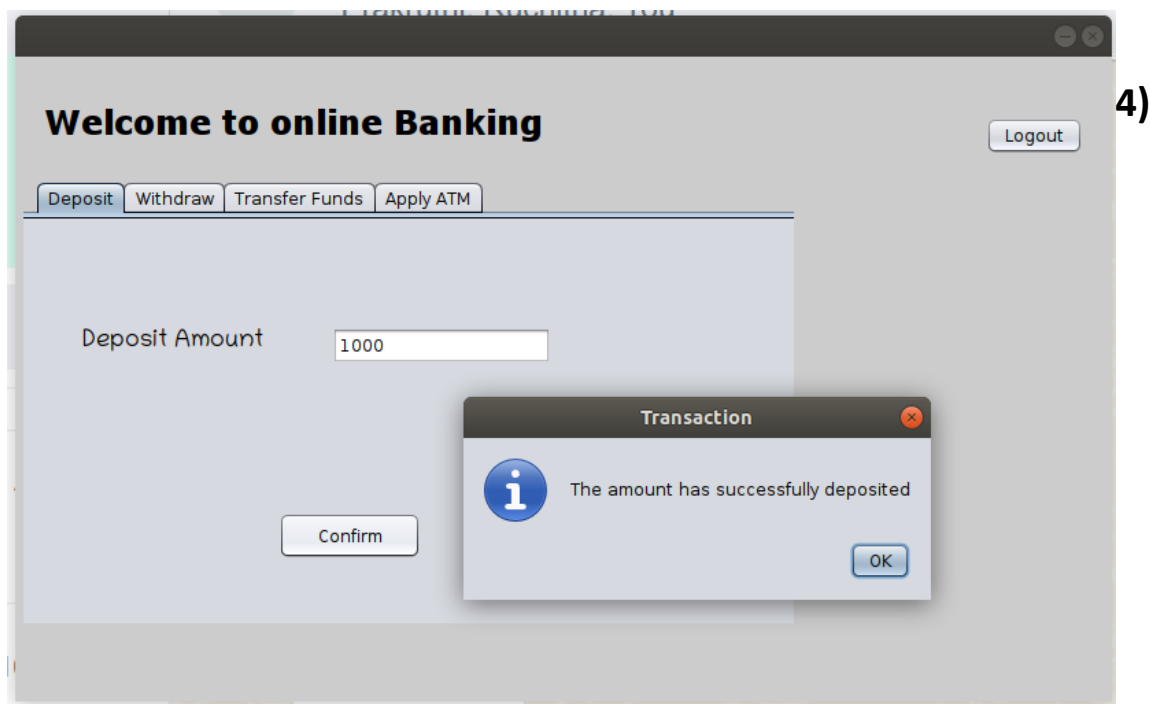
## 1) Sign In page



Entering the default loginid and password provided by the employee

A screenshot of a web browser window displaying the 'User Login' form. The form has a light gray background and contains two input fields: 'Login Id' with the value 'Rajeshwari1234' and 'Password' with the value '\*\*\*\*\*'. A 'Login' button is positioned at the bottom center of the form.

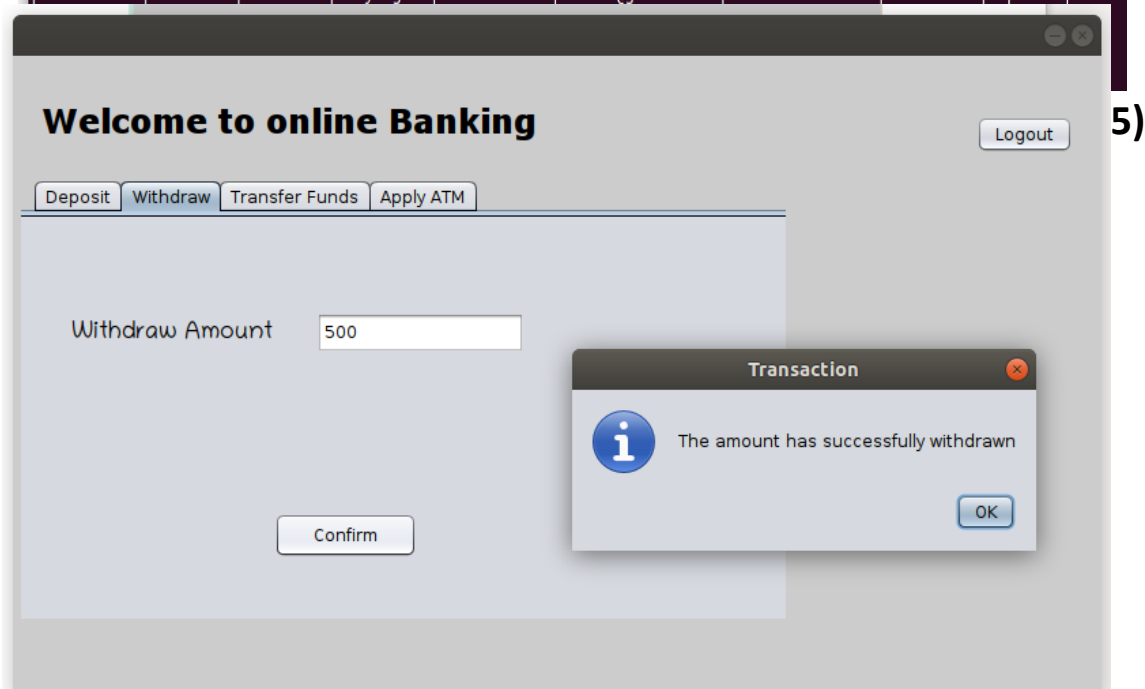
## 3) Depositing cash into the account



**Database view of the amount been deposited**

```
mysql> select * from customer;
```

actnum	acttype	balance	branch	name	emailId	loginid	password	status
123456799	SA	2000	RR Nagar	Rajeshwari	raji@gmail.com	Rajeshwari1234	123456	Open
435261549	RD	1000	Jaynagar	Ruchita	ruchi@gmail.com	Ruchita1234	123456	Open



**Withdrawal of the amount**

## 6) Database view showing that the amount has been successfully withdrawn

```
mysql> select * from customer;
```

actnum	acttype	balance	branch	name	emailid	loginid	password	status
123456799	SA	1500	RR Nagar	Rajeshwari	raj1@gmail.com	Rajeshwari1234	123456	Open
435261549	RD	1000	Jaynagar	Ruchita	ruchi@gmail.com	Ruchita1234	123456	Open

```
2 rows in set (0.00 sec)
```

## 7) Amount that has been deposited and withdrawn

```
mysql> select * from transaction;
```

Date	trans_type	Amount	Description
2022-04-27	credit	1000	deposited cash
2022-04-27	credit	1000	deposited cash
2022-04-27	debit	500	cash withdrwan

```
3 rows in set (0.00 sec)
```

## 8) Transfer funds

The screenshot displays a web application window titled "Welcome to online Banking". In the top right corner, there is a "Logout" button. Below the title, there are four tabs: "Deposit", "Withdraw", "Transfer Funds", and "Apply ATM". The "Transfer Funds" tab is currently selected. The main form area contains two input fields: "Transfer to Account Number" with the value "435261549" and "Amount" with the value "200". Below these fields is a "Confirm" button. A modal dialog box titled "Transaction" is overlaid on the main form. It features an information icon (a lowercase 'i' inside a blue circle) and the text "The amount has successfully transferred". An "OK" button is located at the bottom right of the modal.



## 9) Database view of amount being deducted from one account and added to the other.

```
mysql> select * from customer;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| actnum | acttype | balance | branch | name | emailId | loginid | password | status |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 123456799 | SA | 1300 | RR Nagar | Rajeshwari | raji@gmail.com | Rajeshwari1234 | 123456 | Open |
| 435261549 | RD | 1200 | Jaynagar | Ruchita | ruchita@gmail.com | Ruchita1234 | 123456 | Open |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

## 10)Applying for an ATM card

The screenshot shows a web browser window titled "Welcome to online Banking". In the top right corner, there is a "Logout" button. Below the title bar, there are four tabs: "Deposit", "Withdraw", "Transfer Funds", and "Apply ATM", with "Apply ATM" being the active tab. The form contains the following fields:

- Aadhar Card Number: 1234567892
- Name: Rajeshwari
- Mobile Number: 5432876120
- EmailId: raji@gmail.com
- Type: Debit (selected from a dropdown menu)

A "Submit" button is located at the bottom of the form.

## 11) Database view of the atm card details stored.

```
mysql> select * from atm;
+-----+-----+-----+-----+-----+-----+
| actnum | adharCard | name | emailId | mobileNum | type |
+-----+-----+-----+-----+-----+-----+
| 123456799 | 1234567892 | Rajeshwari | raji@gmail.com | 5432876120 | Debit |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

## 6.Team member contributions

NAME	CONTRIBUTION
Rajeshwari R	Equal contribution in creating the use case diagram, activity diagram and state diagram. Implementation of the registration function.
Ruchita V R	Equal contribution in creating the use case diagram, activity diagram and state diagram. Implementation of the transaction function.
Prakruti P	Equal contribution in creating the use case diagram, activity diagram and state diagram. Implementation of the request of ATM card