BANKING SYSTEM

A PROJECT REPORT

Submitted in partial fulfillment for the award of the degree of

B.Tech

in

Information Technology

By

MEHUL GUPTA
(REG NO: 16BIT0117)

Under the guidance of

Prof. BIMAL KUMAR RAY

School of Information Technology and Engineering

November, 2017

CERTIFICATE

This is to certify that the project work entitled “BANKING MANAGEMENT SYSTEM” that is being submitted for Database Management system (ITE1003) is a record of bonafide work done under my supervision by:

MEHUL GUPTA 16BIT0117

of Information Technology_branch under my supervision in G2 slot during the FALL Semester -2017 at V.I.T. University, Vellore-632 014.

Faculty Signature:

Date:

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided us the possibility to complete this project named “BANKING MANAGEMENT SYSTEM”. A special gratitude I give to our faculty, Prof. Bimal Kumar Ray, whose contribution in stimulating suggestions and encouragement, helped me to coordinate our project especially in providing with the necessary guidance from time to time. Furthermore I would also like to acknowledge with much appreciation the crucial role of the library staff of VIT University, who gave the permission to use all required equipment and the necessary materials. Last but not least, many thanks go to the head of the department, of School SITE whose has invested his full effort in guiding me in achieving the goal. We have to appreciate the guidance given by other faculty once again as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

ABSTRACT

The project focuses on developing an application which is not just an ordinary website with some specific information containing pages, rather it is more like an interface where customers can register with the bank, login with their generated username and password in the bank, view their profile, as well as their balance, transfer funds to another account, change their password, their transaction history (Mini-statement), request for loans, request for any updations in their profile info as well as view the status of their requested loans.

There is also a separate functionality provided for the admin login where admin can view their customer's details, approve/deny multiple loan requests at a time, change a customer's profile information, view any customer's balance, and create a new account.

The complete project has been developed by making use Java for front end development and MySQL for backend development.

OBJECTIVE

Banks are considered to be an integral part of cities and they contribute to the definition of a local geography and identity. They also contribute to the preservation of the collective memory, since they constitute a significant social and cultural practice linked to a specific place, which acts as a common reference or landmark for many individuals.

The project focuses on developing an application for the ease of understanding the complete banking system by people not willing to open their accounts in bank. The project aims in taking a step forward in implementing ‘cashless economy’, where all the transactions, payments, transfers, etc. shall be done through online systems, eradicating the usual pen-paper based method. This shall also help people in understanding the basis of banking system and how can they open an account in a bank as well as understand various other functionalities in a very easy manner. In the process, this banking system shall develop trust and support by the people.

REQUIREMENTS

Requirements analysis is done in order to understand the problem, which is to be solved. That is very important activity for the development of database system. The requirements and the collection analysis phase produce both data requirements and functional requirements. The data requirements are used as a source of database design. The data requirements should be specified in as detailed and complete form as possible.

In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of user-defined operations that will be applied to the database (retrievals and updates). The functional requirements are used as a source of application software design.

Data Requirements:

- The database will contain User Info like Account number, account-type, name, father's name, password, balance, address, mobile, email, security-question, security-answer, gender.
- The database shall also contain the loan table containing Loan-ID, account number, amount and the status of the loan (pending or approved or denied).
- The database shall also contain a transaction table containing all the transactions with account number, amount, date/time and the type of transaction.

Functional Requirements:

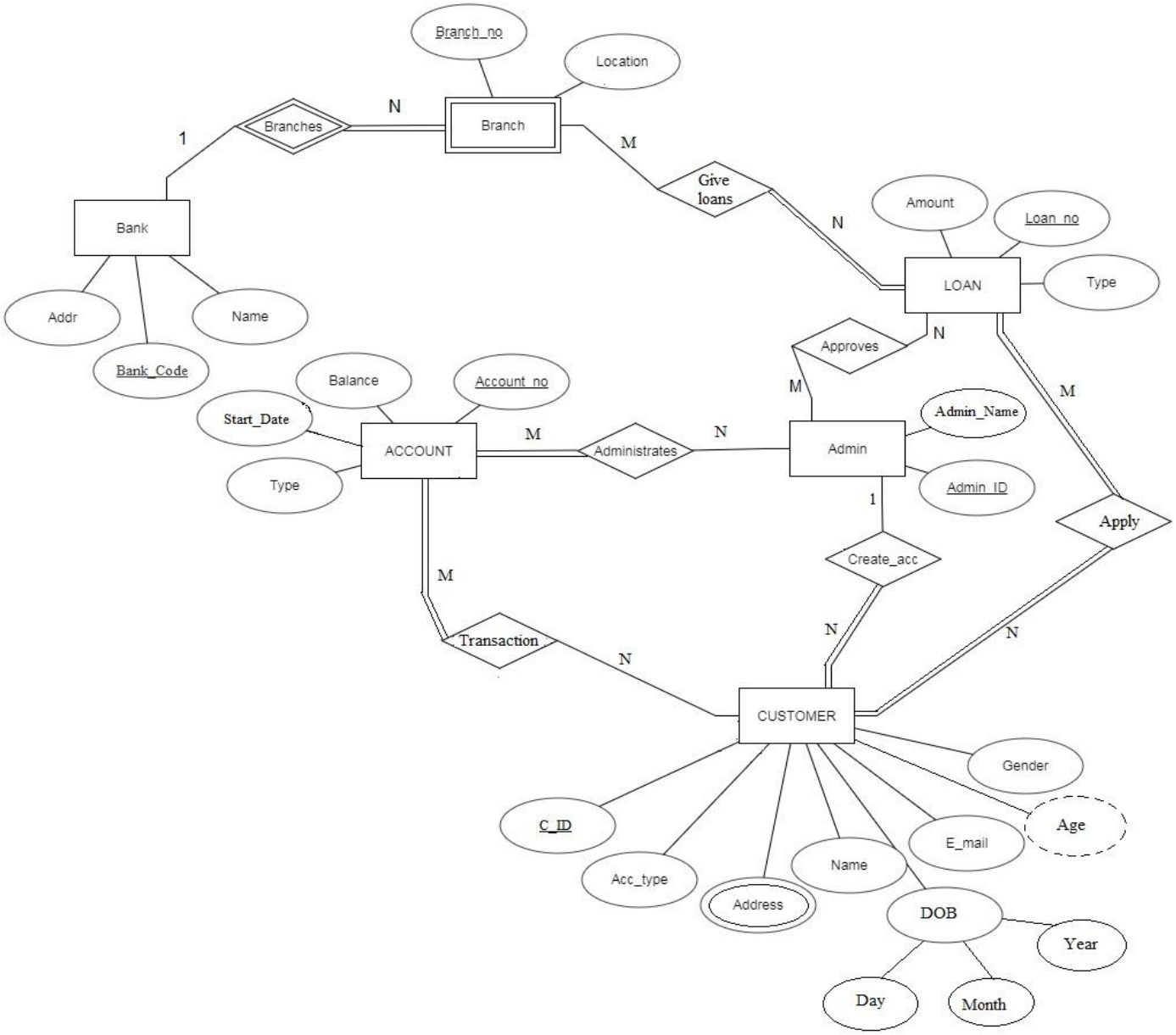
- User must create an account first by requesting the admin before accessing the functionalities.
- All the customers of the bank will have a unique Account no. whose values cannot be NULL.
- Any customer is not allowed to withdraw an amount from his account if after withdrawal the amount in his account falls below the minimum balance of Rs.1000.
- Any customer is eligible to get a loan from the bank if he/she has an account in the bank.
- The customer must request loan first which moves the request to the Approval Workflow.
- Customer must set a Security-Question and answer which will be used to recover the password in case the customer forgets his login password.
- The User-Info of a customer can only be changed by Admin.

Hardware and software requirements:

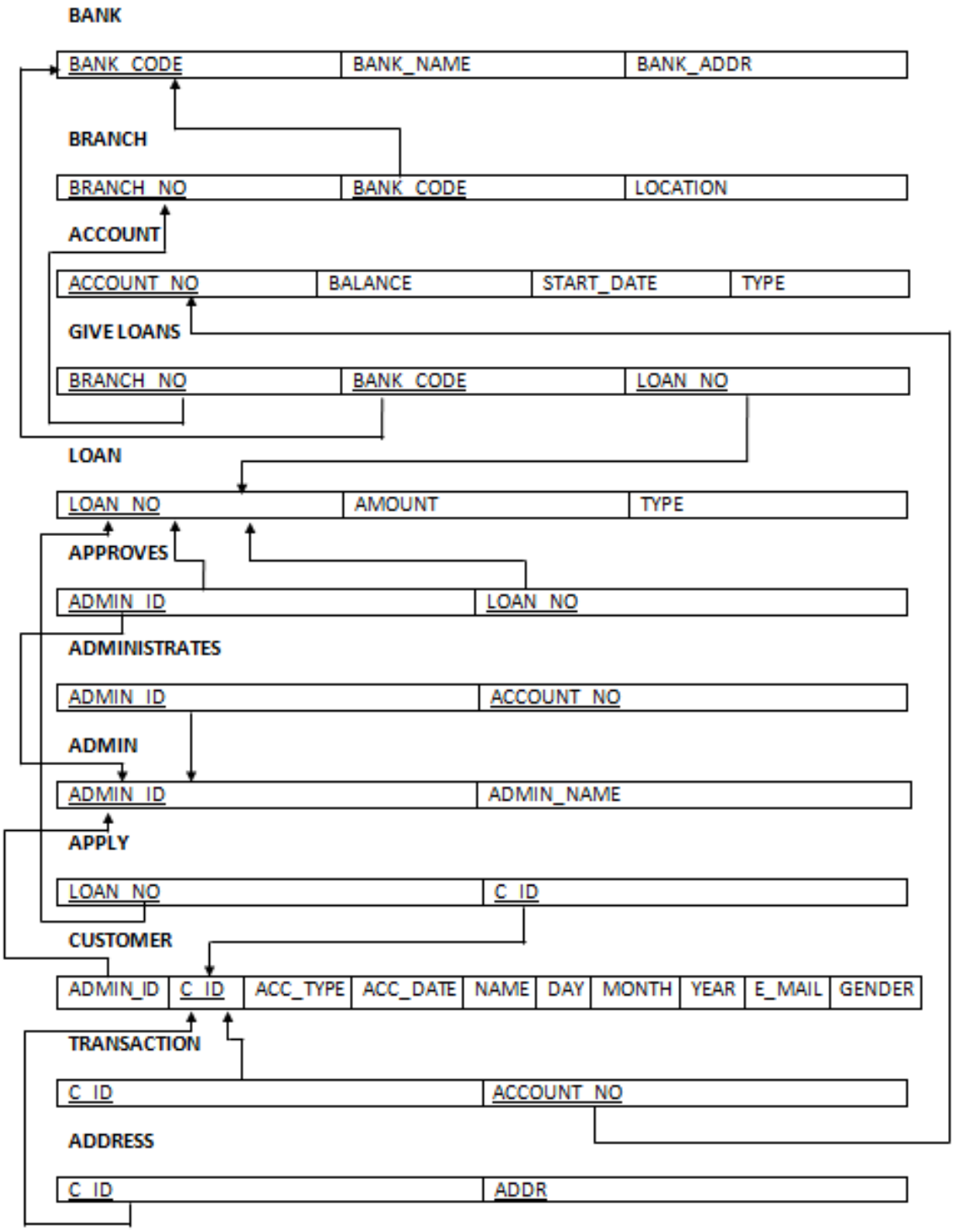
The application shall work on any machine with Java SE Development kit 8 or higher and a server such as apache tomcat version 7.1 or higher along with Mysql database installed with proper data.

For fast processing, a processor of 64-bit, four-core, and 2.5 GHz minimum per core is needed.

Entity-relationship diagram:



Relationship diagram:



Normalized tables:

BANK_1

BANK_CODE	BANK_NAME

BANK_2

BANK_NAME	BANK_ADDR

BRANCH_1

BRANCH_NO	BANK_CODE

BRANCH_2

BANK_CODE	LOCATION

ACCOUNT

ACCOUNT_NO	BALANCE	START_DATE	TYPE

GIVE LOANS

BRANCH_NO	BANK_CODE	LOAN_NO

LOAN

LOAN_NO	AMOUNT	TYPE

APPROVES

ADMIN_ID	LOAN_NO

ADMINISTRATES

ADMIN_ID	ACCOUNT_NO

ADMIN

ADMIN_ID	ADMIN_NAME

APPLY

LOAN_NO	C_ID

CUSTOMER

ADMIN_ID	C_ID	ACC_TYPE	ACC_DATE	NAME	DAY	MONTH	YEAR	E_MAIL	GENDER

ADDRESS

C_ID	ADDR

TRANSACTION

C_ID	ACCOUNT_NO

SQL Queries:

SQL> CREATE TABLE bank1(bank_code number(10) NOT NULL, bank_name varchar2(30) NOT NULL, CONSTRAINT code PRIMARY KEY(bank_code));

```
SQL> CREATE TABLE bank1(bank_code number(10) NOT NULL, bank_name varchar2(30) NOT NULL, CONSTRAINT c
ode PRIMARY KEY(bank_code));

Table created.

SQL> desc bank1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BANK_CODE                                 NOT NULL NUMBER(10)
 BANK_NAME                                 NOT NULL VARCHAR2(30)
```

SQL> CREATE TABLE bank2(bank_name varchar2(30) NOT NULL,bank_addr varchar2(30) NOT NULL, CONSTRAINT name PRIMARY KEY(bank_name));

```
SQL> CREATE TABLE bank2(bank_name varchar2(30) NOT NULL,bank_addr varchar2(30) NOT NULL, CONSTRAINT
name PRIMARY KEY(bank_name));

Table created.

SQL> desc bank2;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BANK_NAME                                 NOT NULL VARCHAR2(30)
 BANK_ADDR                                 NOT NULL VARCHAR2(30)
```

SQL> CREATE TABLE branch1(branch_no number(20) NOT NULL, bank_code number(10), CONSTRAINT fk_code FOREIGN KEY(bank_code) references bank1(bank_code), CONSTRAINT p_branch1 PRIMARY KEY(branch_no,bank_code));

```
SQL> CREATE TABLE branch1(branch_no number(20) NOT NULL, bank_code number(10), CONSTRAINT fk_code FO
REIGN KEY(bank_code) references bank1(bank_code), CONSTRAINT p_branch1 PRIMARY KEY(branch_no,bank_co
de));

Table created.

SQL> desc branch1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NO                                 NOT NULL NUMBER(20)
 BANK_CODE                                 NOT NULL NUMBER(10)
```

SQL> CREATE TABLE branch2 (branch_no number(20) NOT NULL, location varchar2(20));

```
SQL> CREATE TABLE branch2(branch_no number(20) NOT NULL, location varchar2(20));

Table created.

SQL> desc branch2;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NO                                 NOT NULL NUMBER(20)
 LOCATION                                           VARCHAR2(20)
```

SQL> create table loan(loan_no number(20),amount number(20), type varchar2(20), CONSTRAINT p_loan PRIMARY KEY(loan_no));

```
SQL> create table loan(loan_no number(20),amount number(20), type varchar2(20), CONSTRAINT p_loan PR
IMARY KEY(loan_no));

Table created.

SQL> desc loan;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOAN_NO                                   NOT NULL NUMBER(20)
 AMOUNT                                             NUMBER(20)
 TYPE                                               VARCHAR2(20)
```

SQL> create table give_loans(branch_no number(20) NOT NULL,bank_code number(10),loan_no number(20),CONSTRAINT gloans PRIMARY KEY (branch_no,bank_code,loan_no), CONSTRAINT fk_loan FOREIGN KEY (loan_no) references loan(loan_no));

```
SQL> create table give_loans(branch_no number(20) NOT NULL,bank_code number(10),loan_no number(20),C
ONSTRAINT gloans PRIMARY KEY (branch_no,bank_code,loan_no), CONSTRAINT fk_loan FOREIGN KEY (loan_no)
 references loan(loan_no));

Table created.

SQL> desc give_loans;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NO                                 NOT NULL NUMBER(20)
 BANK_CODE                                 NOT NULL NUMBER(10)
 LOAN_NO                                   NOT NULL NUMBER(20)
```

SQL> create table address(c_id number(20), addr varchar2(30),CONSTRAINT p_add PRIMARY KEY(c_id,addr), CONSTRAINT f_add FOREIGN KEY(c_id) references customer(c_id));

```
SQL> create table address(c_id number(20), addr varchar2(30), CONSTRAINT p_add PRIMARY KEY(c_id,addr), CONSTRA
INT f_add FOREIGN KEY(c_id) references customer(c_id));

Table created.

SQL> desc address;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(20)
 ADDR                                      NOT NULL VARCHAR2(30)
```

SQL> create table approves(admin_id number(20) NOT NULL,loan_no number(20), CONSTRAINT admin PRIMARY KEY(admin_id,loan_no), CONSTRAINT loan FOREIGN KEY(loan_no) references loan(loan_no));

```
SQL> create table approves(admin_id number(20) NOT NULL,loan_no number(20), CONSTRAINT admin PRIMARY
 KEY(admin_id,loan_no), CONSTRAINT loan FOREIGN KEY(loan_no) references loan(loan_no));

Table created.

SQL> desc approves;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ADMIN_ID                                  NOT NULL NUMBER(20)
 LOAN_NO                                   NOT NULL NUMBER(20)
```

SQL> create table admin(admin_id number(20) NOT NULL,admin_name varchar2(20) NOT NULL,CONSTRAINT adm PRIMARY KEY(admin_id));

```
SQL> create table admin(admin_id number(20) NOT NULL,admin_name varchar2(20) NOT NULL,CONSTRAINT adm
 PRIMARY KEY(admin_id));

Table created.

SQL> desc admin;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ADMIN_ID                                  NOT NULL NUMBER(20)
 ADMIN_NAME                                NOT NULL VARCHAR2(20)
```

SQL> create table customer(c_id number(20), admin_id number(20), acc_type varchar2(10), name varchar2(20), dob varchar2(10), email varchar2(20), gender varchar2(10), CONSTRAINT ad_id FOREIGN KEY(admin_id) references admin (admin_id), CONSTRAINT p_cid PRIMARY KEY(c_id));

```
SQL> create table customer(c_id number(20), admin_id number(20), acc_type varchar2(10), name varchar
2(20), dob varchar2(10), email varchar2(20), gender varchar2(10), CONSTRAINT ad_id FOREIGN KEY(admin
_id) references admin (admin_id), CONSTRAINT p_cid PRIMARY KEY(c_id));

Table created.

SQL> desc customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(20)
 ADMIN_ID                                           NUMBER(20)
 ACC_TYPE                                           VARCHAR2(10)
 NAME                                               VARCHAR2(20)
 DOB                                                VARCHAR2(10)
 EMAIL                                              VARCHAR2(20)
 GENDER                                             VARCHAR2(10)
```

SQL> create table account(account_no number(20), branch_no number(20), bank_code number(10),start_date varchar2(20), balance number(20), type varchar2(10), CONSTRAINT fk_acc FOREIGN KEY (branch_no,bank_code) references branch1 (branch_no,bank_code), CONSTRAINT pacc PRIMARY KEY(account_no,branch_no,bank_code));

```
SQL> create table account(account_no number(20), branch_no number(20), bank_code number(10),start_da
te varchar2(20), balance number(20), type varchar2(10), CONSTRAINT fk_acc FOREIGN KEY (branch_no,ban
k_code) references branch1 (branch_no,bank_code), CONSTRAINT pacc PRIMARY KEY(account_no));

Table created.

SQL> desc account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NO                                NOT NULL NUMBER(20)
 BRANCH_NO                                          NUMBER(20)
 BANK_CODE                                          NUMBER(10)
 START_DATE                                         VARCHAR2(20)
 BALANCE                                            NUMBER(20)
 TYPE                                               VARCHAR2(10)
```

SQL> create table transaction(c_id number(20), account_no number(20), t_date varchar2(20), CONSTRAINT cid FOREIGN KEY(c_id) references customer(c_id), CONSTRAINT acc FOREIGN KEY(account_no) references account(account_no));

```
SQL> create table transaction(c_id number(20), account_no number(20), t_date varchar2(20), CONSTRAIN
T cid FOREIGN KEY(c_id) references customer(c_id), CONSTRAINT acc FOREIGN KEY(account_no) references
 account(account_no));

Table created.

SQL> desc transaction;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                               NUMBER(20)
 ACCOUNT_NO                                         NUMBER(20)
 T_DATE                                             VARCHAR2(20)
```

SQL> create table apply(loan_no number(20), c_id number(20),CONSTRAINT p_apply PRIMARY KEY(loan_no,c_id), CONSTRAINT f_apply FOREIGN KEY(loan_no) references loan(loan_no), CONSTRAINT f2_app FOREIGN KEY(c_id) references customer(c_id));

```
SQL> create table apply(loan_no number(20), c_id number(20),CONSTRAINT p_apply PRIMARY KEY(loan_no,c_id), CONS
TRAINT f_apply FOREIGN KEY(loan_no) references loan(loan_no), CONSTRAINT f2_app FOREIGN KEY(c_id) references c
ustomer(c_id));

Table created.

SQL> desc apply;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOAN_NO                                   NOT NULL NUMBER(20)
 C_ID                                      NOT NULL NUMBER(20)
```

IMPLEMENTATION

Front end:

The front end is developed completely using Java using the Eclipse interface. Eclipse is an integrated development environment used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment.

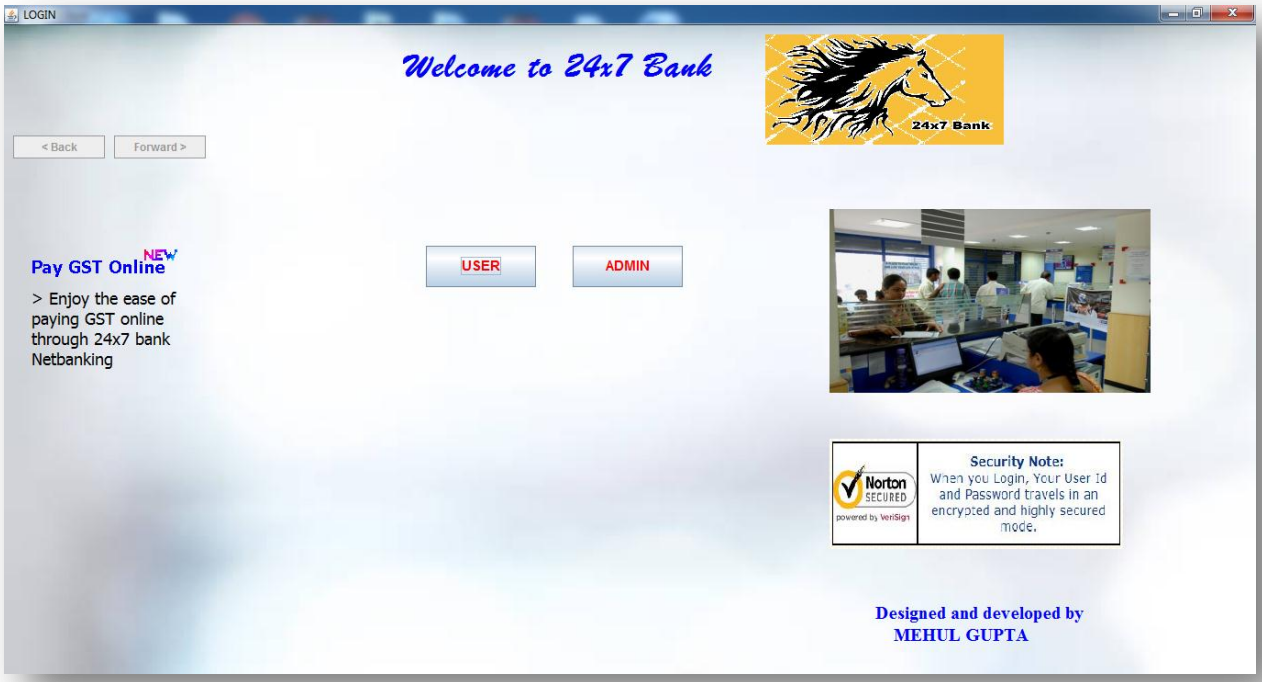
Also, for the project to run, we must have the Java SE development kit (JDK) installed. The JDK includes the Java Runtime Environment (JRE), as well as various tools and API's necessary for development in Java.

Back end:

The back end of the project is implemented using MySQL. MySQL is the most popular Open Source Relational SQL Database Management System. The SQL phrase stands for structured query language.

APPLICATION:

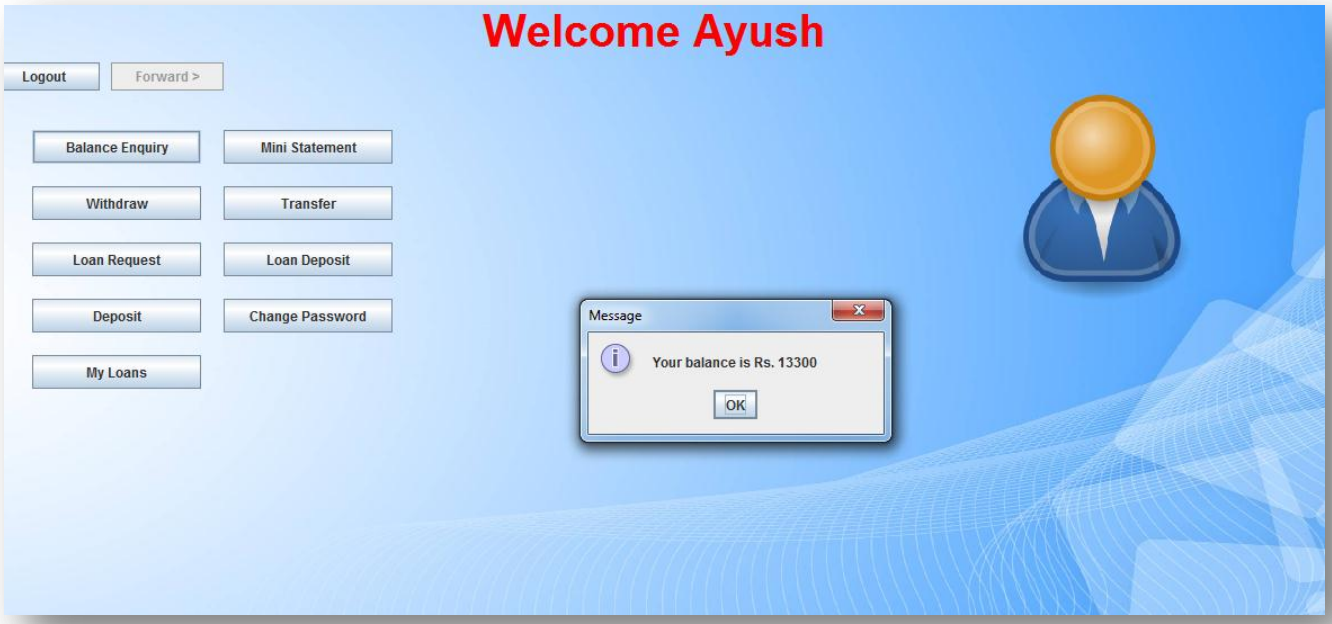
1) HOME PAGE:



2) USER LOGIN:

3) USER HOME PAGE:

4) Balance Enquiry



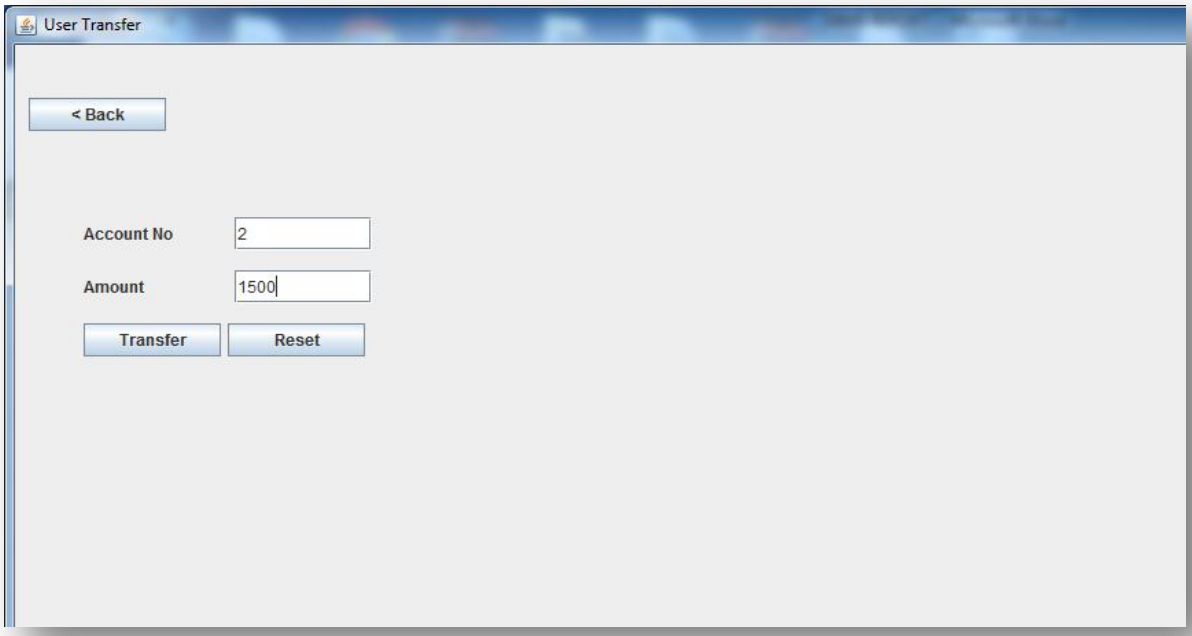
5) Mini statement

S. No.	Transaction id	Account No	Amount	Date	Type
1	1	1	2000	19 09:16:43 IST 2017	Transfer-withdraw
2	4	1	100	19 09:17:32 IST 2017	withdraw
3	5	1	500	19 09:17:46 IST 2017	deposit
4	6	1	1500	19 23:06:04 IST 2017	Transfer-withdraw
5	8	1	1000	19 23:06:21 IST 2017	withdraw
6	12	1	6000	19 23:09:37 IST 2017	transfer-deposit
7	14	1	7000	19 23:10:03 IST 2017	withdraw
8	16	1	1500	29 12:41:51 IST 2017	deposit

6) Withdraw/Deposit

After withdrawal, balance is:

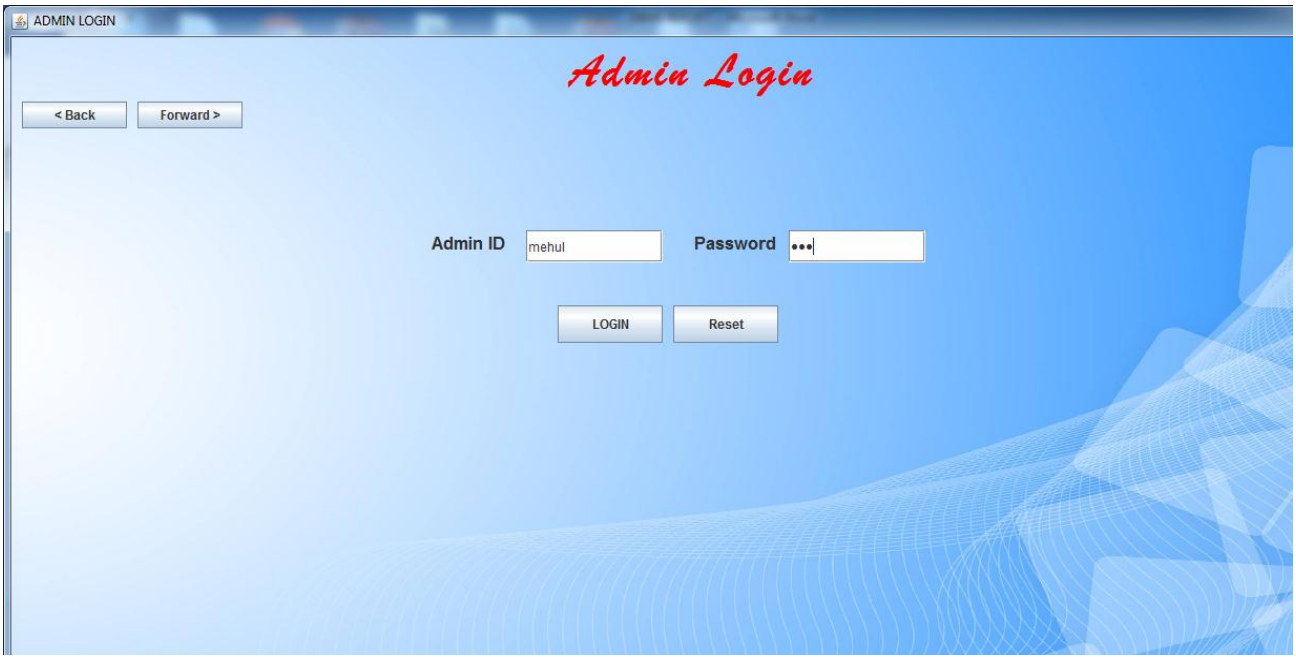
7) Transfer amount



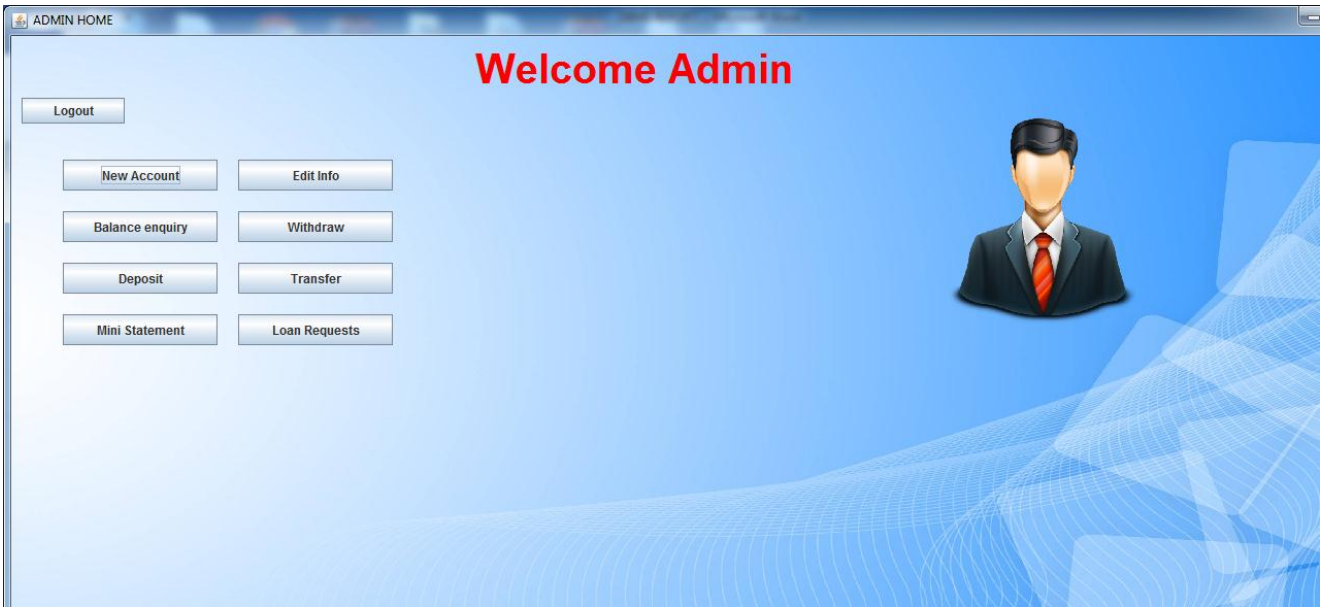
8) Request for loan

9) View my loans

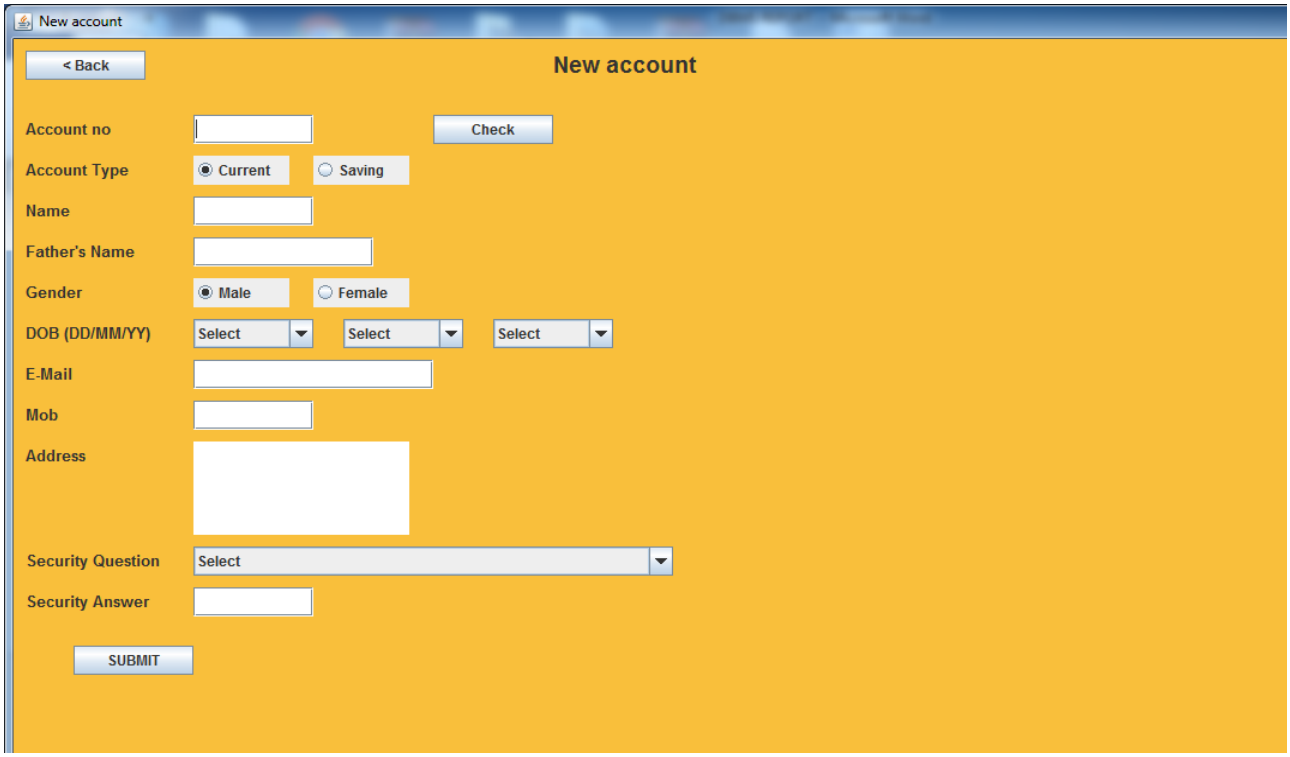
10) Admin Login

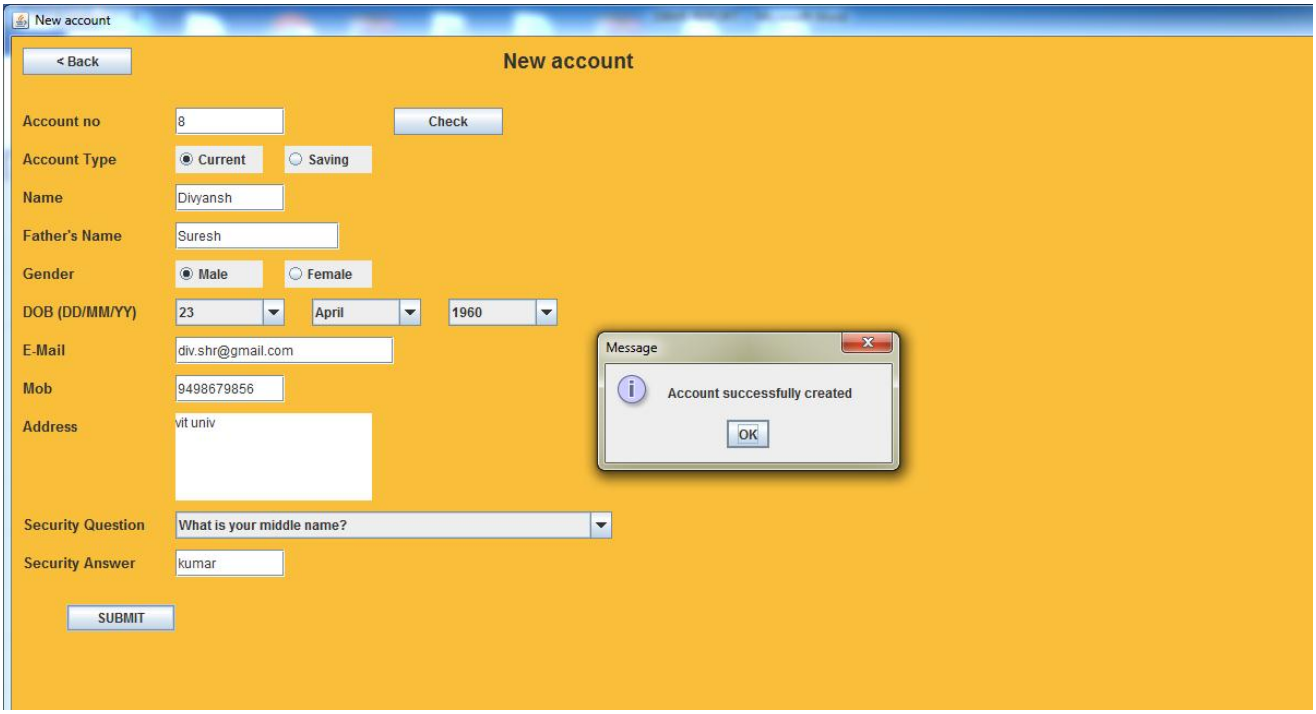


11) Admin Home page

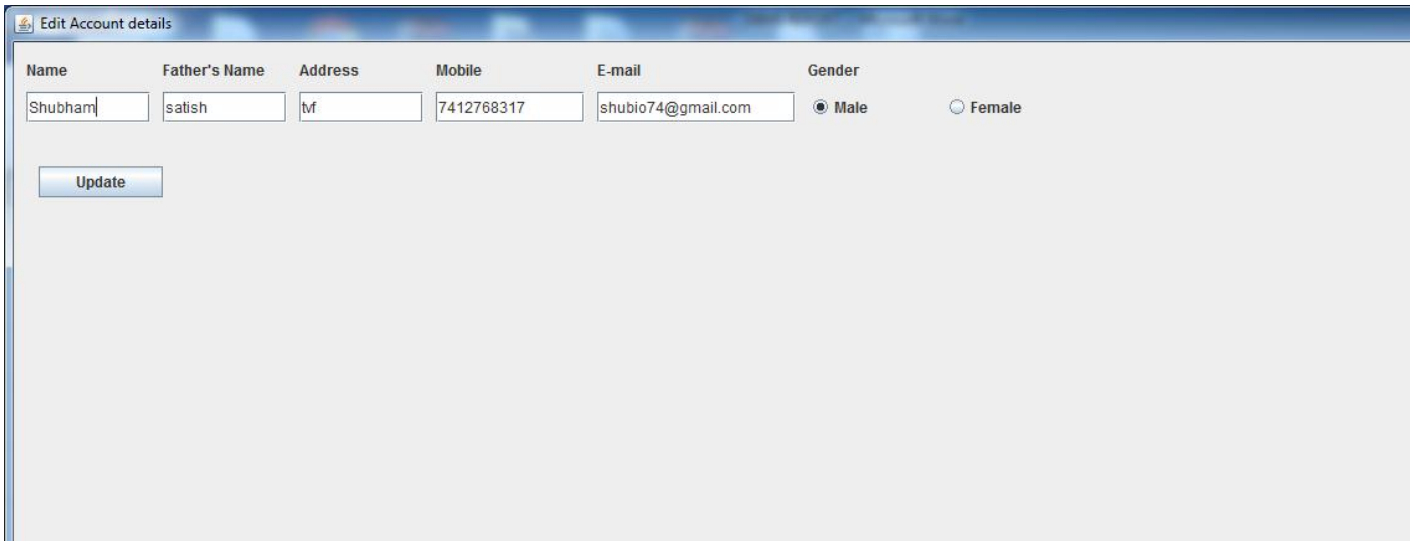


12) Opening a new account



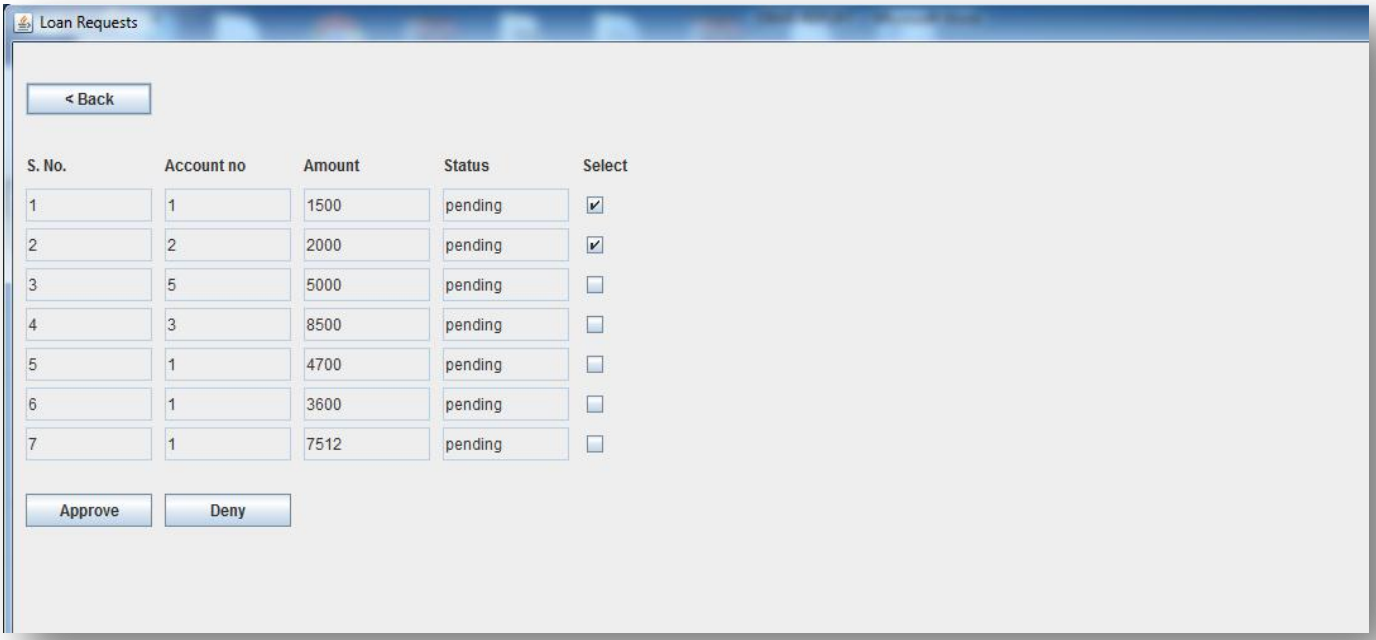


13) Edit account details

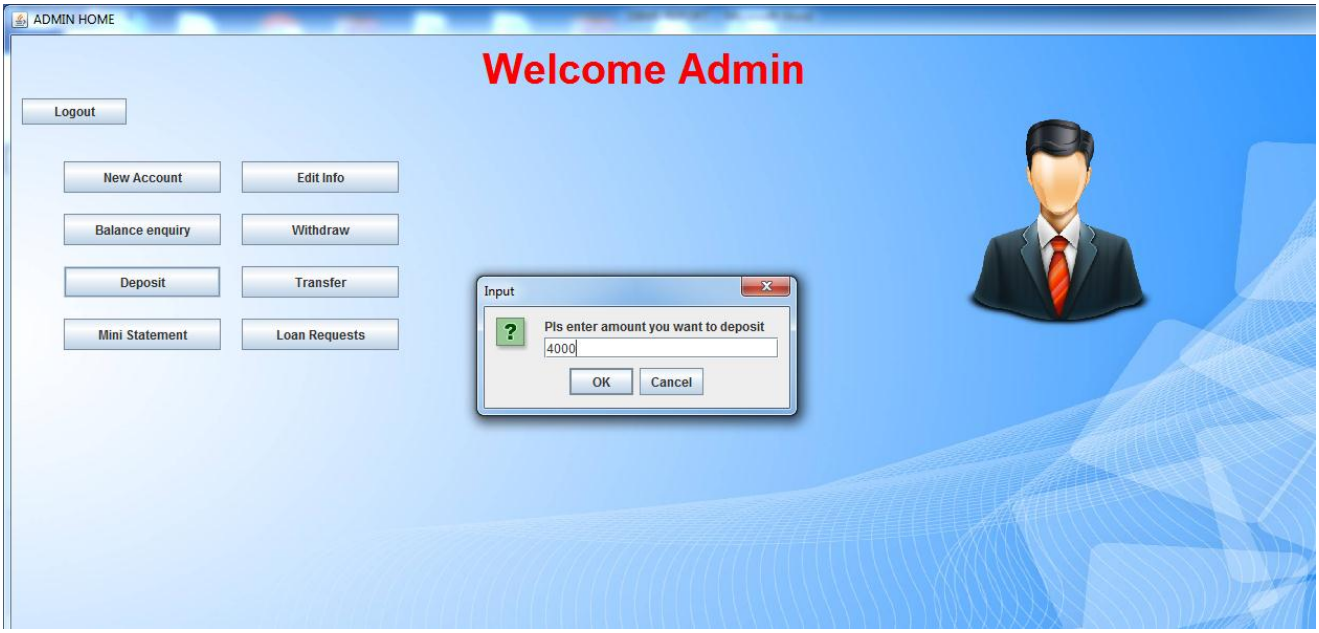


14) Viewing balance of a customer

15) Approve/deny Loans



16) Withdraw/deposit/transfer by admin account



CONCLUSION:

Nowadays, traditional reservation ways of where technology dominates human life. With the software and technological devices, exceptions are reduced and even terminated. Also, people prefer easy, quick and safe way for every part of his life. This project is designed to meet the requirements of an advanced banking system. It has been developed in Java and the database has been built in My SQL server keeping in mind the specifications of the system.

In our project: with this banking management system; banking companies can provide comfortable facilities to their customers. The relationship between bank manager, employee, and customer satisfy a good communication to complete the process. With this platform we developed, we are hoping to reduce time wasting, avoid misunderstandings, provide easy data flow, customer pleasure, and less hard work. We believe that we have accomplished our goals and satisfied with the code we developed.

REFERENCES

[1] Elmasri and Navathe, “Fundamentals of Database Systems” , 3/e,
Addison - Wesley, 2001

[2] A Silberschaltz, H.F. Korth, and Concepts”, 3/e,l,1997 Tata Mcgraw Hil

[3] Thomas M. Connolly, Carolyn E. B Practical Approach to Design Implementation
Addison –Wesley, 2005

[4] “Herbert Schildt” Java: The Complete Reference, Ninth Edition 9th Edition

[5] “Kathy Sierra” Head First Java, 2nd Edition 2nd Edition