**CHAPTER 1**

* 1. **INTRODUCTION**

Today, banks are looking beyond the transactions to the full opportunity on how to manage their customers. Accordingly, they are moving beyond managing clients as simple contacts to a whole new level of client relationship management, crafting a superior commercial client experience that gives the bank a competitive advantage and a more loyal, profitable and committed customers. Banking System refers to systems that enable bank employees to access accounts and general information on bank products and services through a personal computer or other intelligent device. But most of these systems do not focus on how best to manage and keep their customer’s data more secured. The chances and threats that the internet symbolizes is no longer news to the present day banking sector. No traditional bank would dare face investment analysts or new customers without an internet strategy. The main intention behind the commencement of electronic banking services is to provide the customers with an alternative that is more responsive and with less expensive options. With options more secured, customers have more control than ever. Their expectations are how safe and secured their personal information would be. They also want personal attention and highly customized products and services. This Bank Management System aims to provide critical information for managing the bank customers more effectively, and encourages other banks that already have similar systems to move beyond transactions to better employee management approach. Employee management is a broad concept that essentially covers the following: Understanding customer needs, matching customer needs to the best offers in terms of product, service and delivery channels, and effectively managing the critical moments during the customer life cycle. Present experience shows that most of the banks in emerging markets are not fully managing their customers effectively due to lack of the necessary capabilities to manage these customer. Better customer management has been shown to directly affect the profitability of banking institutions. No doubt that this Bank management system would provide such bank with much better profits and other benefits. However, financial services providers, who may find this application particularly helpful in expanding their operations and better tailoring their products and services, include commercial banks and other interested institutions in the financial services space would be more than satisfied and also provides a better customer satisfaction. I was motivated to develop this Bank Customers Management System by various bank specific experiences I had with some banks, my current existing research on other bank systems, and interviews with users of such systems. More importantly, this system is not designed to be prescriptive and does not advocate for a single approach to customer management in the banking sectors. Rather, to encourage financial services providers in making informed choices in the challenges, opportunities, and effective practices in providing a safe and secured customer management across the globe.

**Problem Definition**

At present most of the banking applications are yet to overcome the rapidly growing attacks on their customer private data. Issues suck fraud operating within a conventional environment. However current systems are still trying to cope with the existing institutional structure, which is really meant for usual banking system only and not managing their customer’s information more effectively. Lack of adequate security measure is making it really challenging to successfully transform the bank customers’ management systems from where it is now to where it should be. This challenge still lingers.

## 1.2.OBJECTIVE OF THE PROJECT

This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updation can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

The main objective of our project is providing the different typed of customers facility, the main objective of this system is to find out the actual customer service. Etc.

## It should fulfill almost all the process requirements of any Bank.

* It should increase the productivity of bank by utilizing the working hours more and more, with minimum manpower.

This project includes the entire upgraded feature required for the computerization banking system. This system is very easy to use, so that any user can use without getting pre-knowledge about this. Its very much user friendly and meet almost all daily working process requirements. This system is completely GUI based and can be use by mouse and as well as keyboard. This system is melded in such a way that has got all features to upgrade without making much change in existing components.

* 1. **ORGANIZATION OF REPORT**

**Chapter 1**

covers the overview, problem statement, objective, scope, Justification of the project.

**Chapter 2**

Software and Hardware specifications requirements for the project development.

**Chapter 3**

provides the system analysis, results and the discussions. The functional and non-functional requirement; the system architecture which is use case diagram, activity diagram, data flow or sequence diagram, entity relationship diagram and unit testing.

**Chapter 4**

includes the concluding remarks, contribution, limitation of the system, suggestion of the future work, and summary.

**CHAPTER 2**

**SYSTEM SPECIFACTIONS**

**2.1 SOFTWARE SPECIFICATION**

The software specifications for the Banking Management System are

1.NetBeans IDE

2.Mysql

3.Java

4.JDBC

**NetBeans**

NetBeans IDE offers first-class tools for Java web, enterprise, desktop, and mobile application development. It is consistently the first IDE to support the latest versions of the JDK, Java EE, and JavaFX. It provides smart overviews to help you understand and manage your applications, including ouf-of-the-box support for popular technologies such as Maven.

With its end-to-end application development features, constantly improving Java Editor, and continual speed and performance enhancements, NetBeans IDE sets the standard for application development with cutting edge technologies out of the box.

NetBeans is coded in Java and runs on most operating systems with a Java Virtual Machine (JVM), including Solaris, Mac OS, and Linux.

NetBeans manages the following platform features and components:

* User settings
* Windows (placement, appearance, etc.)
* NetBeans Visual Library
* Storage
* Integrated development tools
* Framework wizard

NetBeans uses components, also known as modules, to enable software development. NetBeans dynamically installs modules and allows users to download updated features and digitallyauthenticatedupgrades. NetBeans IDE modules include NetBeans Profiler, a Graphical User Interface (GUI) design tool, and NetBeans JavaScript Editor.  
  
 NetBeans framework reusability simplifies Java Swing desktop application development, which provides platform extension capabilities to third-party developers.

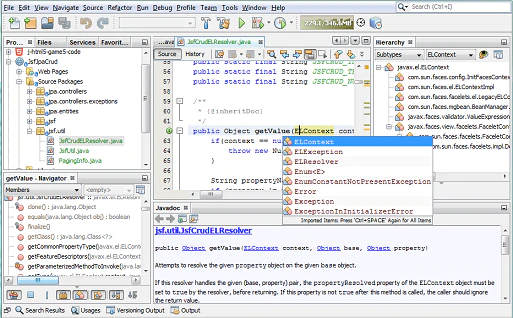


Fig1.1 NetBeans IDE user interface

**MySql**

MySQL is an important component of an open source enterprise stack called [LAMP](https://whatis.techtarget.com/definition/LAMP-Linux-Apache-MySQL-PHP). LAMP is a web development platform that uses Linux as the operating system, [Apache](https://whatis.techtarget.com/definition/Apache) as the web server, MySQL as the relational database management system and [PHP](https://whatis.techtarget.com/definition/PHP-Hypertext-Preprocessor) as the object-oriented scripting language. (Sometimes [Perl](https://whatis.techtarget.com/definition/Perl) or [Python](https://whatis.techtarget.com/definition/Python) is used instead of PHP.)

Originally conceived by the Swedish company MySQL AB, MySQL was acquired by Sun Microsystems in 2008 and then by Oracle when it bought Sun in 2010. Developers can use MySQL under the GNU General Public License ([GPL](https://searchdatacenter.techtarget.com/definition/GNU-General-Public-License-GNU-GPL-or-simply-GPL)), but enterprises must obtain a commercial license from Oracle.

Today, MySQL is the RDBMS behind many of the top websites in the world and countless corporate and consumer-facing web-based applications, including Facebook, Twitter and YouTube.

### **How MySQL works**

MySQL is based on a [client-server](https://searchnetworking.techtarget.com/definition/client-server) model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). MySQL server is available as a separate program for use in a client-server networked environment and as a library that can be embedded (or linked) into seperate applications.

MySQL operates along with several utility programs which support the administration of MySQL databases. Commands are sent to MySQLServer via the MySQL client, which is installed on a computer.

MySQL was originally developed to handle large databases quickly. Although MySQL is typically installed on only one machine, it is able to send the database to multiple locations, as users are able to access it via different MySQL client interfaces. These interfaces send SQL statements to the server and then display the results.

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

1. DDL – Data Definition Language
2. DQl – Data Query Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language

**DDL(Data Definition Language)**

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

**Examples of DDL commands:**

[CREATE](https://www.geeksforgeeks.org/sql-create/) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).

[DROP](https://www.geeksforgeeks.org/sql-drop-truncate/) – is used to delete objects from the database.

[ALTER](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)-is used to alter the structure of the database.

[TRUNCATE](https://www.geeksforgeeks.org/sql-drop-truncate/)–is used to remove all records from a table, including all spaces allocated for the records are removed.

[COMMENT](https://www.geeksforgeeks.org/sql-comments/) –is used to add comments to the data dictionary.

[RENAME](https://www.geeksforgeeks.org/sql-alter-rename/)–is used to rename an object existing in the database.

**DQL (Data Query Language)**

DML statements are used for performing queries on the data within schema objects. The purpose of DQL Command is to get some schema relation based on the query passed to it.

**Example of DQL:**

[SELECT](https://www.geeksforgeeks.org/sql-select-clause/) – is used to retrieve data from the a database.

**DML(Data Manipulation Language)**

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML:**

[INSERT](https://www.geeksforgeeks.org/sql-insert-statement/) – is used to insert data into a table.

[UPDATE](https://www.geeksforgeeks.org/sql-update-statement/) – is used to update existing data within a table.

[DELETE](https://www.geeksforgeeks.org/sql-delete-statement/) – is used to delete records from a database table.

**DCL(Data Control Language)**

DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

**Examples of DCL commands:**

GRANT-gives user’s access privileges to database.

REVOKE-withdraw user’s access privileges given by using the GRANT command.

**TCL(transaction Control Language)**

TCL commands deals with the [transaction within the database](https://www.geeksforgeeks.org/sql-transactions/).

**Examples of TCL commands:**

COMMIT– commits a Transaction.

[ROLLBACK](https://www.geeksforgeeks.org/sql-transactions/)– rollbacks a transaction in case of any error occurs.

SAVEPOINT–sets a savepoint within a transaction.

SET TRANSACTION–specify characteristics for the transaction.

**JDBC**

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

* JDBC-ODBC Bridge Driver,
* Native Driver,
* Network Protocol Driver, and
* Thin Driver



Fig 1.2 JDBC connection.

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

* Register the Driver class
* Create connection
* Create statement
* Execute queries
* Close connection

**2.2 HARDWARE SPECIFICATIONS**

Hardware is referred to as computing potential.

Int this project, three hardware categories have been considered during software planning:

DEVELOPMENT SYSTEM, which is a computer and related peripherals that will be used during the software development phase. The development sysrem has been used because it can accommodate multiple users,maintain large volumes of information and support a rich assortment of software tools.

TARGET MACHINE on which the software will eventual is executed.

OTHER hardware elements of the new system.

**CHAPTER 3**

**DESIGN AND IMPLEMENTATON**

**3.1 INTRODUCTION**

In this chapter, we will discuss the implementation and working of the system.This project is divided into two parts which are frontend and backend. For the Frontend implementation, we used java for development of user interface for displaying the contents to the admin or user.While the backend of this project uses sql ,where the data of each customer is stored in stable storage. By using the sql commands it is possible to retrieve ,update ,insert the data in the database.

This bank management system also allow user to add new customer account, delete account and user can also modify existing user account information. Using this system user can also search any individual account in few seconds. Using our bank management system user can also check any translation in any account. Our system also provide security check to reduce fraud. The system will check the user’s existence in the database and provide the set of services with respect to the role of the user.

The Project Banking system has been made to automate the Banking system. Through this bank management system user can manage all bank account activity like deposit money, withdraw money, transfer money from one account to another account, online payment etc. Using this bank management system user can check his account detail online like balance in account, bank statement etc.  The Administrator can check bank account with a login can work out with A/C holders of the bank can withdraw/ deposit cash /  cheque  /DD to/from their accounts. This system is also help bank user to create New account easily. The project makes a sincere effort to provide all the below-mentioned features to meet the requirements of the bank.

Using this system user can manage following account type

* Savings Account
* Current Account

The main objective of the system is to automate all the banking process  with improved performance an realize the vision of paperless banking. Salient features of the proposed bank management system is given below.

Using this bank management system any information can be easily searched. User can view all the details of the customer.

Using this system user can create new customer account and maintain its data efficiently and effectively. All records of account and customer are stored in separate files. Which are maintained constantly update by system.

Manage large number of customer details with ease. Particular A/c information can be modified A particular customer record can be modified for one or more field’s customer name, address by providing A/c number.

Create a statistical report to facilitate the finance department work. Activities like updating, modification, deletion of records should be easier. A customer record can be easily deleted by authorize user by providing A/c number.

The proposed system provides faster data access, data entry and retrieval.

The proposed system is more efficient, fast, reliable, user friendly. Over and above the proposed system does not have any possibility of data loss during processing.

**ROLE OF ADMIN**

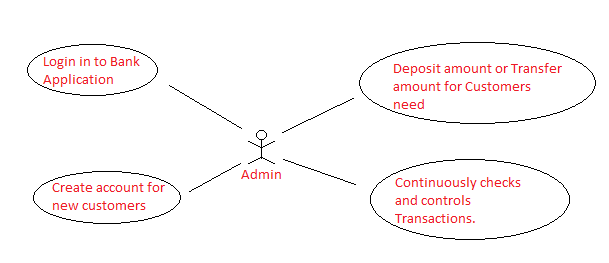


Fig 3.1.1 Picture depecting the role of Admin.

**ADMIN Module**

1. Admin Login.
2. Create Customers Account.
3. Deposit Amount into Customers Account.
4. Can Edit Customer Details.
5. Withdraw,Transfer Money from one account to another account Can View Transaction details.

**ER Diagram**

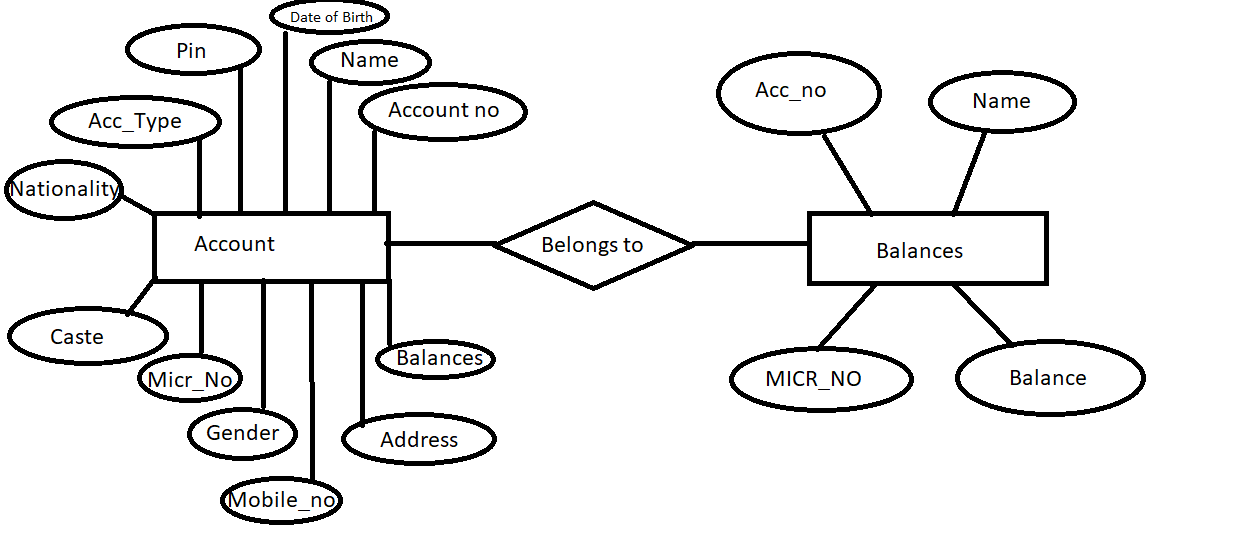
****

Fig 3.1.2 ER Diagram for Banking management system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Id** | **Name** | **Type** | **Not Null** | **Default Value** | **Primary Key** |
| 0 | Account no | INTEGER | 1 | Null | 1 |
| 1 | Name | CHAR | 0 | Null | 0 |
| 2 | Date of Birth | INTEGER | 0 | Null | 0 |
| 3 | PIN | INTEGER | 0 | Null | 0 |
| 4 | Account type | INTEGER | 0 | Null | 0 |
| 5 | Nationality | CHAR | 0 | Null | 0 |
| 6 | Caste | CHAR | 0 | Null | 0 |
| 7 | Micr no | INTEGER | 0 | Null | 0 |
| 8 | Gender | CHAR | 0 | Null | 0 |
| 9 | Mobile | INTEGER | 0 | Null | 0 |
| 10 | Address | INTEGER | 0 | Null | 0 |
| 11 | Sec\_q | CHAR | 0 | Null | 0 |
| 12 | Balance | INTEGER | 0 | Null | 0 |

Fig 3.1.3 Account Table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column ID** | **Name** | **Type** | **Not Null** | **Default Value** | **Primary Key** |
| 0 | Name | CHAR | 0 | Null | 0 |
| 1 | Account no | INTEGER | 0 | Null | 0 |
| 2 | Micr no | INTEGER | 0 | Null | 0 |
| 3 | Balance | INTEGER | 0 | Null | 0 |

Fig 3.1.4 Balance Table.

**3.2 SOURCECODE AND SCREENSHOTS**

**3.2.1 Login Page**

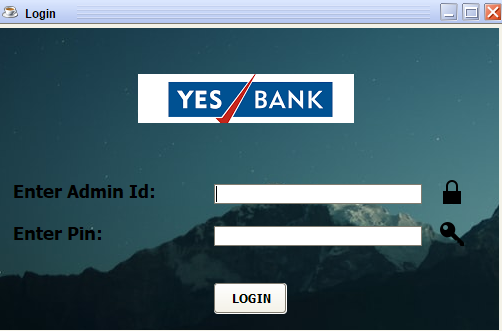
****

Fig 3.2.1.1 Login Page

Login page Has Password protected Authentication ,where only admin can only has access to this login. When pin is entered in the password textfield it will be seen as dots.

**Source code for Login Button**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String sql="select \* from Account where Acc=? and Pin=?";

try{

pst=conn.prepareStatement(sql);

pst.setString(1, jTextField1.getText());

pst.setString(2, jPasswordField1.getText());

rs=pst.executeQuery();

if(rs.next()){

setVisible(false);

Loading ob=new Loading();

ob.setUpLoading();

ob.setVisible(true);

rs.close();

pst.close();

}

else{

JOptionPane.showMessageDialog(null, "Incorrect Credentials");

}

}catch(Exception e){ JOptionPane.showMessageDialog(null,e);

}finally{

try{

rs.close();

pst.close();

}catch(Exception e){

}

}

}

**3.2.2 Loading Screen**

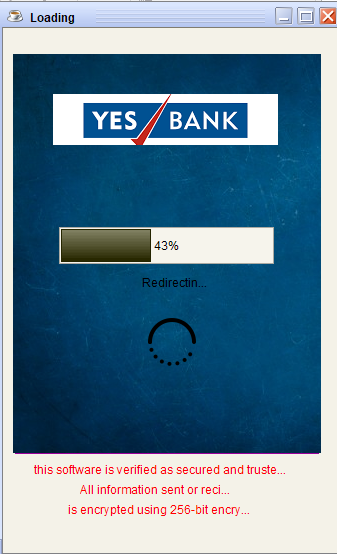
****

Fig 3.2.2.1 Loading Screen

public void run(){

try{

for(int i=0;i<=200;i++){

s=s+i;

int m=jProgressBar1.getMaximum();

int v=jProgressBar1.getValue();

if(v<m){

jProgressBar1.setValue(jProgressBar1.getValue()+1);

}

else{

i=201;

setVisible(false);

MyPage ob=new MyPage();

ob.setVisible(true);

}

Thread.sleep(50);

}

}catch(Exception e){

}

}

**3.2.3 Home Page**

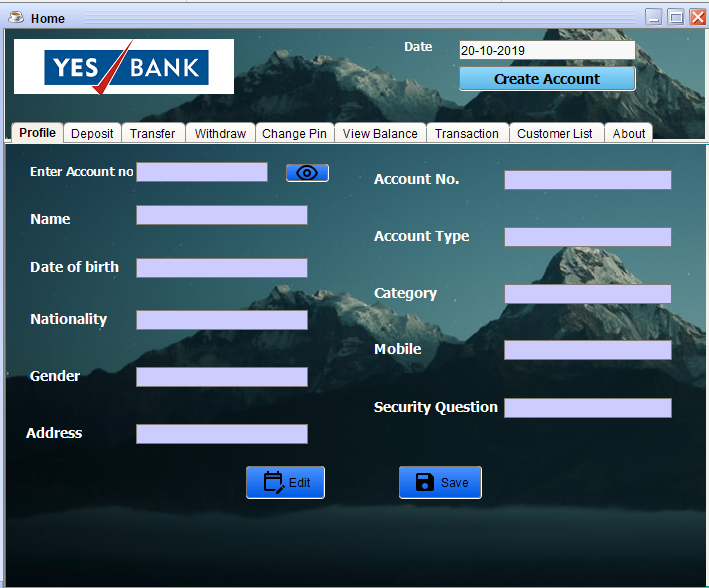
****

Fig 3.2.3.1 Home Page

**Edit Button Code**

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

jTextField5.setEditable(true);

jTextField6.setEditable(true);

jTextField7.setEditable(true);

jTextField10.setEditable(true);

jTextField11.setEditable(true);

jTextField12.setEditable(true);

}

**Save Button code**

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try{

String value1=jTextField5.getText();

String value2=jTextField6.getText();

String value3=jTextField7.getText();

String value4=jTextField10.getText();

String value5=jTextField11.getText();

String value6=jTextField12.getText();

String value7=jTextField1.getText();

String sql="update Account set Nationality='"+value1+"',Gender='"+value2+"',Address='"+value3+"',Caste='"+value4+"',Mob='"+value5+"',Sec\_Q='"+value6+"' where Acc='"+value7+"'";

pst=conn.prepareStatement(sql);

pst.execute();

JOptionPane.showMessageDialog(null, "Profile Updated");

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

**3.2.4 Create Account**

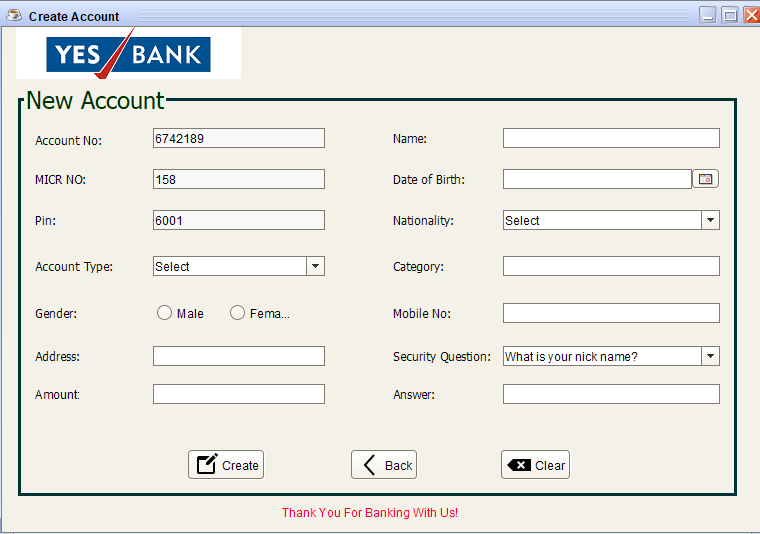


Fig 3.2.4.1 Create Account table.

**Source Code For Create account**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String sql="insert into Account(Acc,Name,DOB,Pin,Acc\_Type,Nationality,Caste,MICR\_No,Gender,Mob,Address,Sec\_Q,Sec\_A,Balance) values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

try{

pst=conn.prepareStatement(sql);

pst.setString(1,jTextField1.getText());

pst.setString(2,jTextField8.getText());

pst.setString(3,((JTextField)jDateChooser1.getDateEditor().getUiComponent()).getText());

pst.setString(4,jTextField3.getText());

pst.setString(5, (String) jComboBox1.getSelectedItem());

pst.setString(6, (String) jComboBox2.getSelectedItem());

pst.setString(7,jTextField5.getText());

pst.setString(8,jTextField2.getText());

jRadioButton1.setActionCommand("Male");

jRadioButton2.setActionCommand("Female");

pst.setString(9,buttonGroup1.getSelection().getActionCommand());

pst.setString(10,jTextField11.getText());

pst.setString(11,jTextField6.getText());

pst.setString(12, (String) jComboBox3.getSelectedItem());

pst.setString(13,jTextField13.getText());

pst.setString(14,jTextField7.getText());

pst.execute();

Bal();

JOptionPane.showMessageDialog(null, "Congratulations\n Account has been Created");

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

**3.2.5 Deposit Panel**

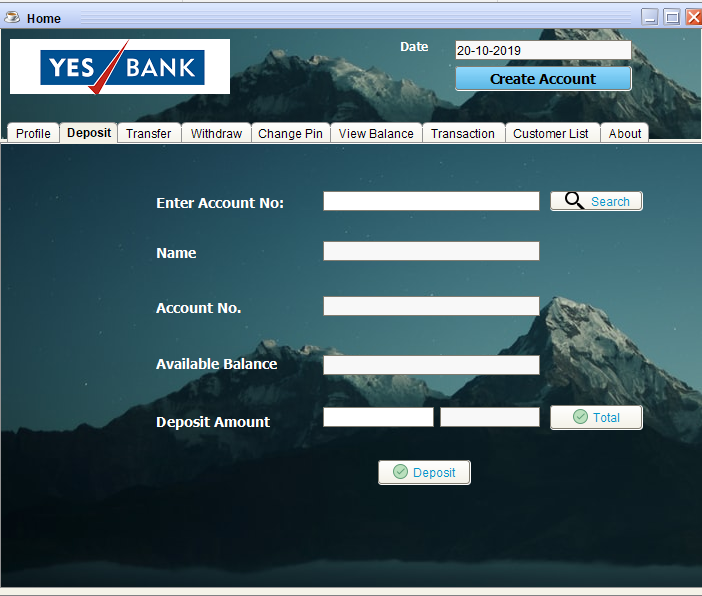


Fig 3.2.5.1 Deposit Panel

**Source Code For Deposit**

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try{

String value1=jTextField13.getText();

String value2=jTextField18.getText();

String sql="update Balances set Balance='"+value2+"' where Acc='"+value1+"'";

pst=conn.prepareStatement(sql);

pst.execute();

JOptionPane.showMessageDialog(null, "Successfully Deposited");

jTextField13.setText("");

jTextField14.setText("");

jTextField15.setText("");

jTextField16.setText("");

jTextField17.setText("");

jTextField18.setText("");

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

**3.2.6 Transfer Panel**

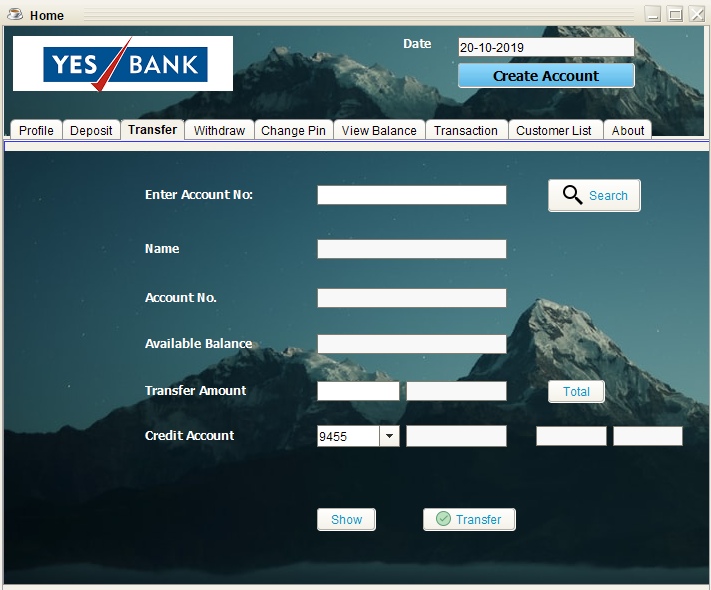


Fig 3.2.6.1 Transfer Panel

**Source Code For Transfer**

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

TransferD();

TransferC();

}

public void TransferC(){

try{

String value1=(String) jComboBox1.getSelectedItem();

String value2=jTextField26.getText();

String sql="update Balances set Balance='"+value2+"' where Acc='"+value1+"'";

pst=conn.prepareStatement(sql);

pst.execute();

JOptionPane.showMessageDialog(null, "Succesfully Transfered");

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

public void TransferD(){

try{

String value1=jTextField19.getText();

String value2=jTextField24.getText();

String sql="update Balances set Balance='"+value2+"' where Acc='"+value1+"'";

pst=conn.prepareStatement(sql);

pst.execute();

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

**3.2.7 WithDraw**

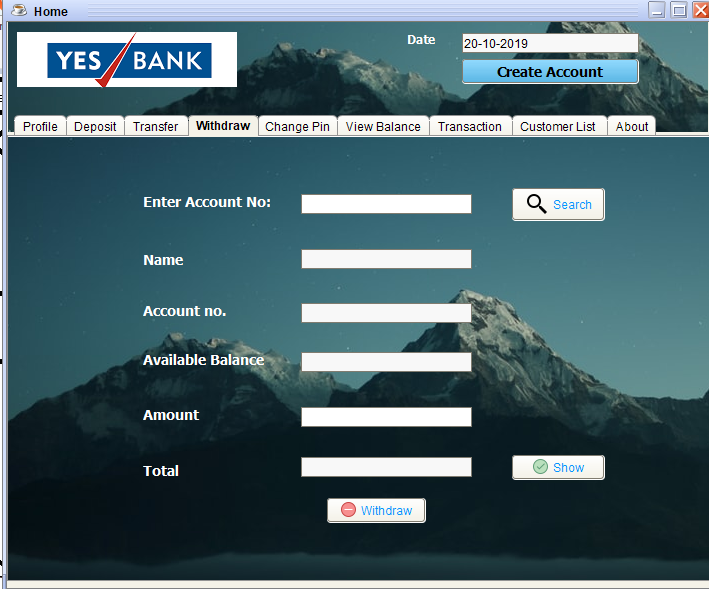


Fig 3.2.7.1 Transfer Panel.

**Source Code For WithDraw**

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try{

String a1=jTextField27.getText();

String a2=jTextField32.getText();

String sql="update Balances set Balance='"+a2+"' where Acc='"+a1+"'";

pst=conn.prepareStatement(sql);

pst.execute();

JOptionPane.showMessageDialog(null, "Withdraw Successful");

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

**3.2.8 Change Pin**

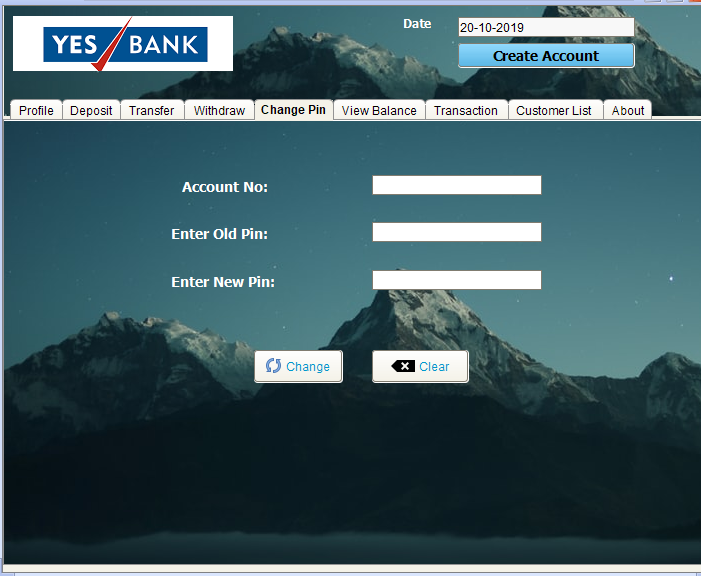


Fig 3.2.8.1 Change Pin Panel

**Source Code For Change Pin**

private void jButton15ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try{

String value1=jTextField44.getText();

String value2=jTextField42.getText();

String sql="update Account set Pin='"+value2+"' where Acc='"+value1+"'";

pst=conn.prepareStatement(sql);

pst.execute();

JOptionPane.showMessageDialog(null, "Pin Successfully Changed");

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

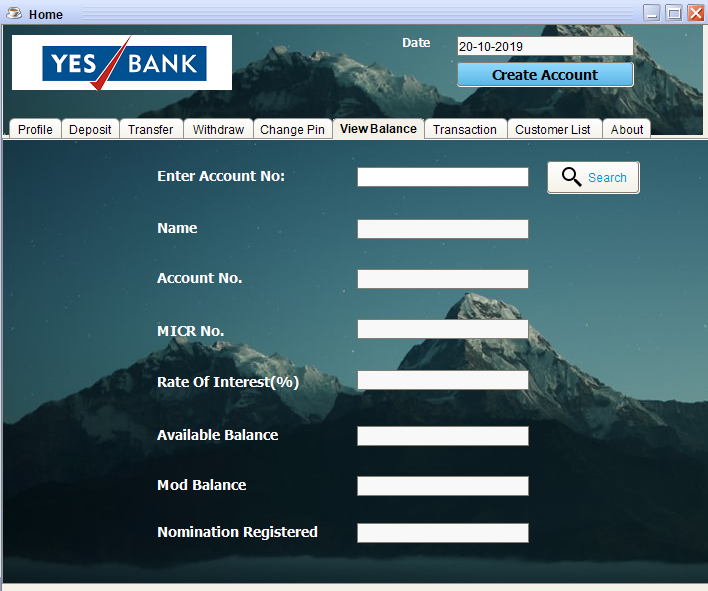
**3.2.9 View Balance**

Fig 3.2.9.1 View Balance Panel.

**Source Code For View Balances**

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String sql="select \* from Balances where Acc=?";

try{

pst=conn.prepareStatement(sql);

pst.setString(1, jTextField33.getText());

rs=pst.executeQuery();

if(rs.next()){

String add1=rs.getString("Name");

jTextField34.setText(add1);

String add2=rs.getString("Acc");

jTextField35.setText(add2);

String add3=rs.getString("MICR\_No");

jTextField36.setText(add3);

String add4=rs.getString("Balance");

jTextField38.setText(add4);

jTextField37.setText("4 %");

jTextField39.setText("Rs 0.00");

jTextField40.setText("No");

}

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);}

}

**3.2.10 Transactions**

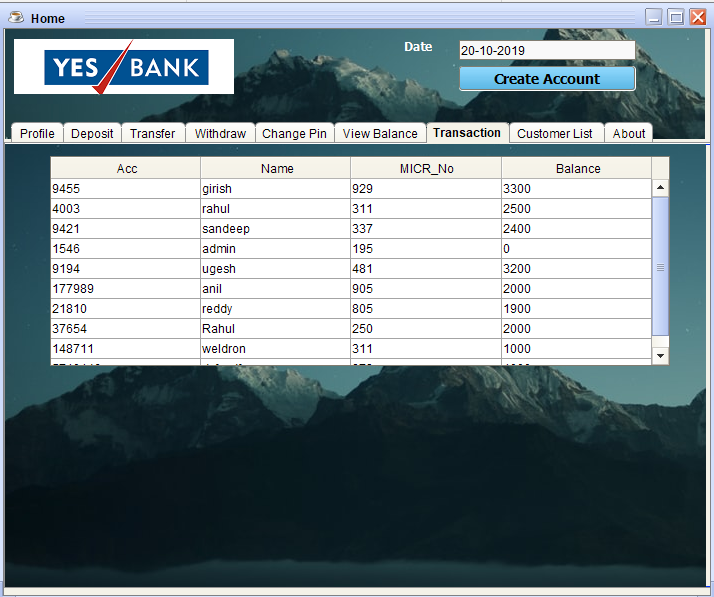


Fig 3.2.10.1 Transactions Panel.

**Source Code For Transactions**

public void Table1(){

try{

String sql="select Acc,Name,DOB,Acc\_Type,Gender,Mob From Account";

pst=conn.prepareStatement(sql);

rs=pst.executeQuery();

jTable1.setModel(DbUtils.resultSetToTableModel(rs));

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}finally{

try{

rs.close();

pst.close();

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);}}

}

**3.2.11 Customer List**

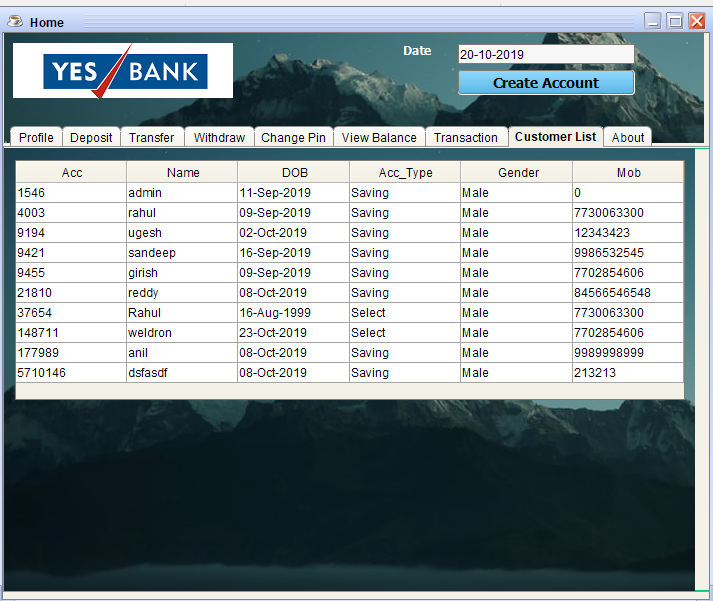
****

Fig 3.2.11.1 Customer List Panel

**Source Code For Transaction Table**

public void Table2(){

try{

String sql="select Acc,Name,MICR\_No,Balance From Balances";

pst=conn.prepareStatement(sql);

rs=pst.executeQuery();

jTable2.setModel(DbUtils.resultSetToTableModel(rs));

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}finally{

try{

rs.close();

pst.close();

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

}

**3.3 TESTING AND VALIDATION**

**Login Page**

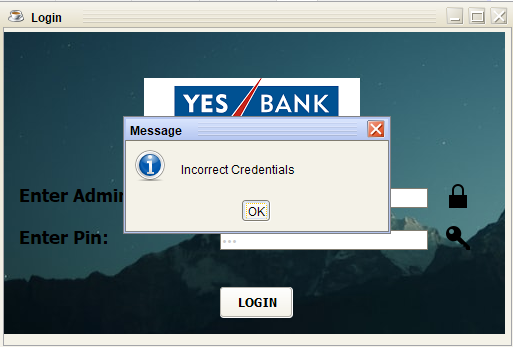
****

Fig 3.3.1 Login Page Testing.

In this Testing of the admin login ,initially enterd wrong credentials to test whether the Authentication is given to the only admin or not, Incase of wrong credentials admin is also not able to login to the bank servers. By this test we can say that the only correct Admin Id and Pin can login to the Bank servers.

**Profile Update**

In profile update testing ,we entered the wrong account no which results to the pop up of dialog box showing “enter account no correctly” from this it is valid that only the account no linked and contained in the database can only be retrieved and can be edited . In the case of editing only the textfields nationality, gender, address, category, mobile are editable and remaining textfields name, dateofbirth, account no, account type are not editable the reason is the bank managing the accounts of customers based on the type of account the customers need and account number should be unique.

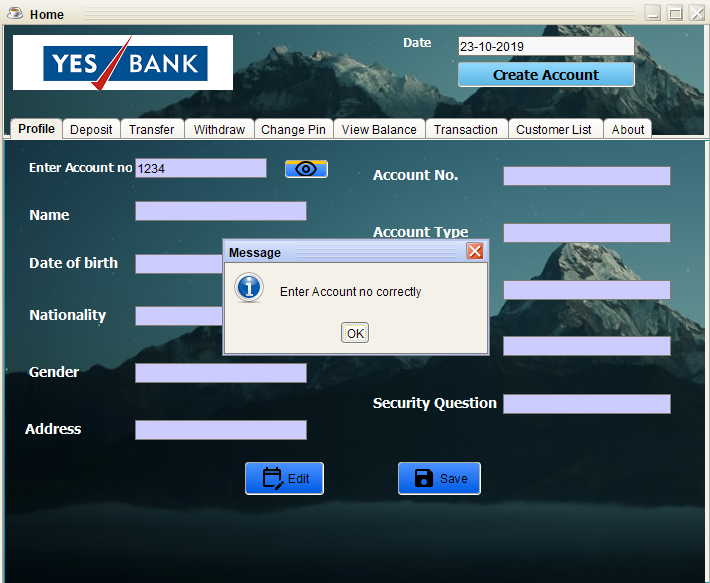
****

Fig 3.3.2 Profile Update Validation.

**Create Account**

In create account panel the account no and the pin are automatically generated using the random function which is inbuilt in the java programming language.The purpose of generating the random account no and random pin is to bhe unique as the admin cannot remember all account numbers . so by random function different account numbers can be get and they are unique this reduces the man power like searching all records for this account number whether it exists or not.

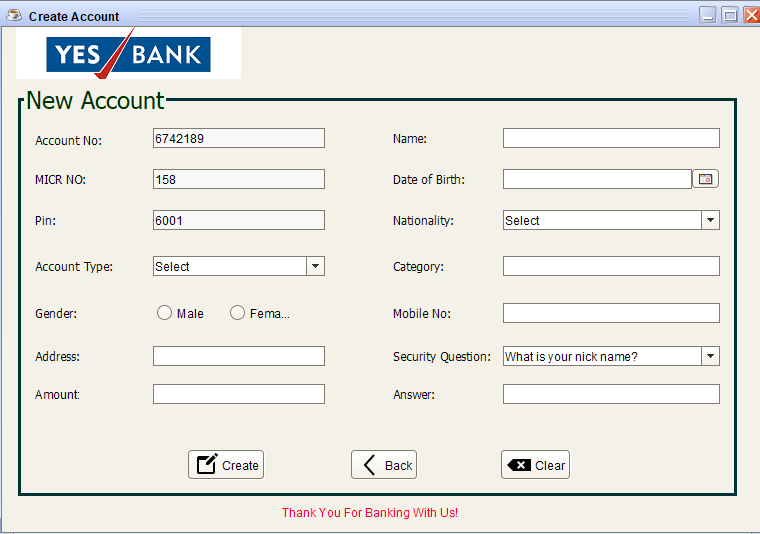


Fig 3.3.3 Create account Testing and Validation.

**Deposit**

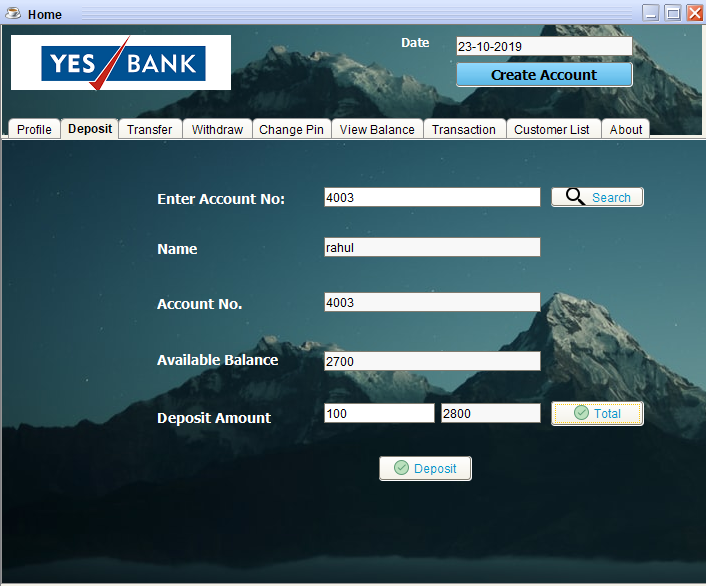
****

Fig 3.3.4 Deposit Testing and validation.

In deposit panel if the accout number is not found in the database it shows dialogbox with message enter accout no correctly. And the other validation in this deposit table are the button show will display after the deposit how much money is there in the customer account this makes the customer can know the correct amount deposited into his account.

**Pin change**

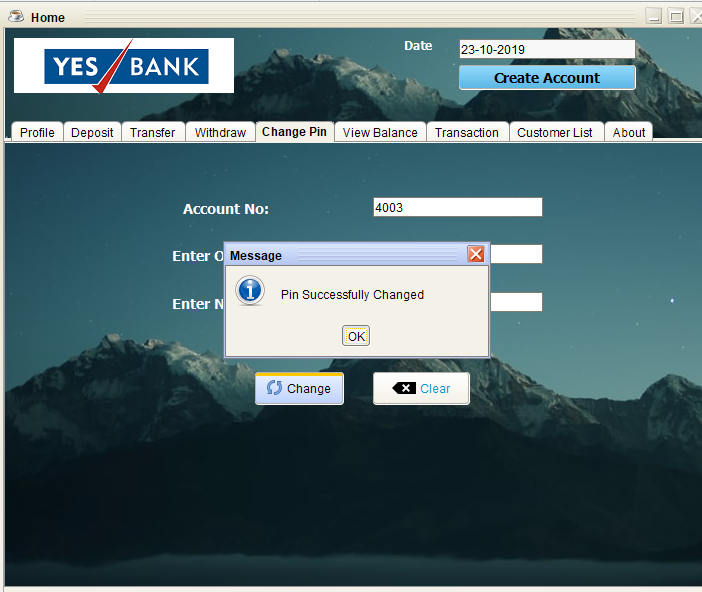


Fig 3.3.5 Pin change Testing and validation.

In the pin change panel with the help of account no and the old pin only the pin can be changed. By this only the real customer of that respective account can only be allowed to change the pin.

**CHAPTER 4**

**CONCLUSION**

**4.1 Concluding Remarks**

I have successfully designed, develop and implemented this Bank customers Management system which provides a more secured approach in managing bank customer’s information and strengthens the relationships between banks and their customers by providing the right solutions that uses a multilevel security to improve customer satisfaction. I therefore encourages other developers of similar application to think twice on how best they can improve in developing a more secured system that will meet the challenges we face today especially on the banking sector and other financial institutions.

**4.2 Summary**

The world is changing at a staggering rate and technology is considered to be the key driver for these changes around us. Many activities are handled electronically due to the acceptance of information technology at home as well as at workplace. Slowly but steadily, Banks around the world is moving towards the internet banking. No one wants to lose his/her money but there are several cases where peoples bank account, personal information are being compromised due to lack of adequate security in their online banking systems. E-banking or Online banking is generic term for the delivery of banking services and products through the electronic channels such as the internet, the cell phone etc. The concept and scope of e-banking is still evolving and customers protection should be prioritize. It is imperative to note that This Bank customers Management System (BCMS) provides a high level security measure for effective customer management.

**Future Enhancements**

This project was developed to fulfill user requirement; however there are lots of scope to improve the performance of the Online Banking System in the area of user interface, database performance, and query processing time. Etc.

So there are many things for future enhancement of this project. The future enhancements that are possible in the project are as follows.

* Linking and integration of any legacy system for accounting.
* Integration with other bank and government agencies through Web Services
* Connection to third-party OLAP applications
* Electronic Data Interchange (EDI) system for ATM machine
* Web Interface for net banking.
* In the area of data security and system security.
* Provide more online tips and help.

**References**

**Book**

* Henry F. Korth& Abraham Silberschatz [2005], [5 Edition], Data Base System Concepts,MC Graw Hill.
* Herbert Schildt [2008], [5th Edition], The Complete Reference Java2, TATA McGraw-Hill. 2.
* E Balaguruswamy [2007], [3 rd Edition], Programming with Java, A Primer, TATA McGraw-Hill

**Websites Referred**

* [www.w3schools.com/sql](http://www.w3schools.com/sql)
* [www.tutorialspoint.com/plsql/index.htm](http://www.tutorialspoint.com/plsql/index.htm)
* [www.javatpoint.com/java-jdbc](http://www.javatpoint.com/java-jdbc)