VIT-AP UNIVERSITY

TITLE OF PROJECT

Automatic Banking System

UNDER THE GUIDANCE OF:

PROF. GOKAPAY DILIP KUMAR -

Submitted by-

21BCE7578-ISHAN DAS

21BCE8399-ABHILIPSA PADHY

Overview of the project-

The client will have a client interface in which he can interact with the banking system. It is a web-based interface that will be the web page of the banking application. Starting a page is displayed asking the type of customer he is whether ordinary or a corporate customer. Then the page is redirected to the login page where the user can enter the login details. If the login particulars are valid, the user is taken to a home page with the entire transaction list that he can perform with the bank. All the above activities come under the client interface. The administrator will have an administrative interface which is a GUI so that he can view the entire system. He/she will also have a login page where he can enter the login particulars to perform all his actions. This administrative interface provides a different environment such that he can maintain the database & offer backups to the information in the database. He/she can register the users by providing them with usernames, passwords & by creating accounts in the database. He/she can view the cheque book request & perform actions to issue the cheque books to the clients.

Identification of project scope-

An online banking system will be applicable everywhere, where banking exists. It will be more efficient and easier way to have a record on systems through which everyone can easily access it according to his rights as compared to the traditional banking system. Every bank will prefer the online banking system instead of the traditional banking system as it contains many useful features and fastest methods for the transactions.

Objective-

The purpose of this document is to present a detailed description of the Online Banking System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be liable for the approval or disapproval of the project by the community of the Bank.

Infrastructure-

Software Requirement-

- User on Internet: Web Browser, Operating System (any)
- Application Server: WAS Data Base
- Server: DB2
- Network: Internet Development

- Tools: ASP.Net, HTML, OS(Windows)

Hardware Requirement-

Server Side:

- Operating System: Windows 7/8/8.1/10
- Processor: Pentium 3.0 GHz or higher.
- RAM: 256 Mb or more.
- Hard Drive: 10 GB or more.

Client side :

- Operating System: Windows 7/8/8.1/10
- Processor: Pentium III or 2.0 GHz or higher.
- RAM: 256 Mb or more

User Characteristics

- Customers: The normal users will have an account of fixed or savings and should have a minimum balance of Rs. 500. He can transfer funds to another account of the same bank & may view his monthly or annual statements.
- Industrialists, Entrepreneur, Organizations academicians: These users will have all the three accounts & should have a minimum balance of 20,000 Rs. He can view the statements of his organization or industry.

General Constraints-

- This system works only on a single server.
- This is designed in ASP.Net.
- Language used is java.
- Limited to HTTP/HTTPS protocols

Assumptions and Dependencies

The details of customers such as username, password, account type and their corresponding authority details should be manually entered by the administrator before using this system. Every user should be comfortable of working with computer and net browsing. He should be aware of the banking system. He must have basic knowledge of English too.

Functional Requirements

- Following are the services which this system will provide. These are the facilities and functions required by the customer.
- Online balance check.
- Online shopping opportunity.
- Online data entry by the staff.
- Updating the data.
- Balance transfer.
- Check book Allotment

Process Specification-

- Customer Login: Each Customer will have their account Id and password. This page will require both attributes for them to access their account.
- Bank Features: It is not sure that each visitor to the Bank's website will be a customer. He/she would be a normal visitor interested in reading the features the bank provides. The website's main page should provide him with the basic features and benefits of the bank to these types of users.
- Order for an Account : A new visitor to the Bank's website would be interested in opening a new account in the Bank. So, he must be provided with an easy path to create a new account in the bank.
- Fill out the Form : Newcomers should have to fill out the form to register him/her self with the bank. After filling out the form, If the values inputted by the user were logically correct, his contact details will be sent to the administration block else he will be asked to input the values again.
- Welcome Page : After a user will be login, he will provide an interface offering different tasks (Here this interface will provide many of the functionalities, which the customer needs in the software). He must choose a task to carry on with his work.
- Staff Login: On the Website's main page, A staff login link will also be provided. Bank staff will use to input their IDs and passwords to access their account. Here the type of staff will also be recognized, if he will be of the administration block, he will be sent to the administration module else he will be sent to the record management module.
- Check the balance: After logging in, if the user wants to check his balance, he will have to click the balance check link. It will tell him the current balance of the account through which he is logged in.
- Transfer Balance: If the user wants to transfer his money to some other account, then this module will provide him with this opportunity. He will input the account details of the receiver. After this process, the server will check the balance of

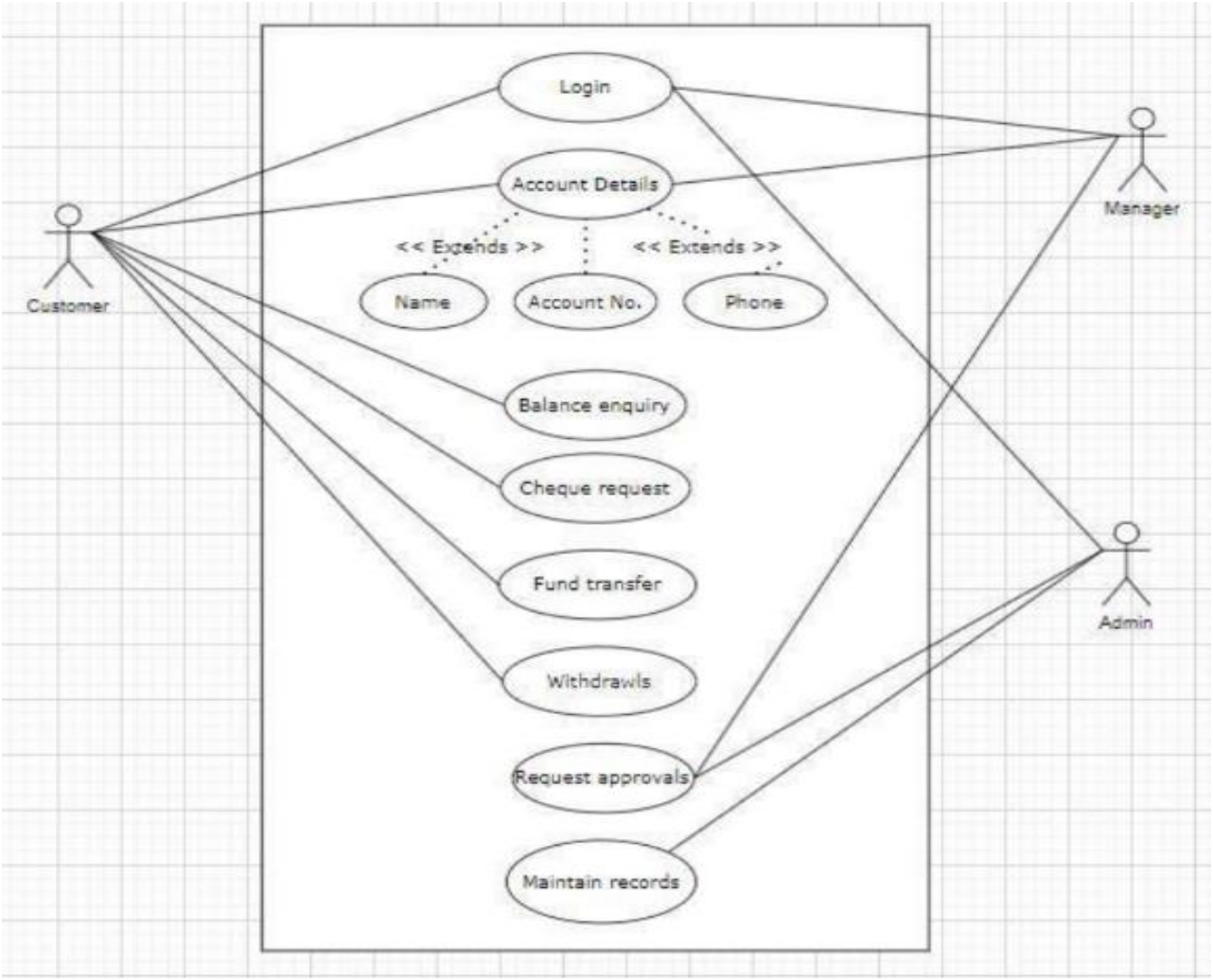
the user and if the transfer balance will be less than the account balance then the transfer will take place else, he will be alarmed that he has lo balance.

- Account detail teller: If the user physically contacts the Bank branch, then he will provide his account detail to the management staff who will inform him about his account. Users will be able to do every task at the branch that they can do online from their homes.
- Order Cash Book: If user's Cheque book has been finished, he will be able to order a new cheque book from this module

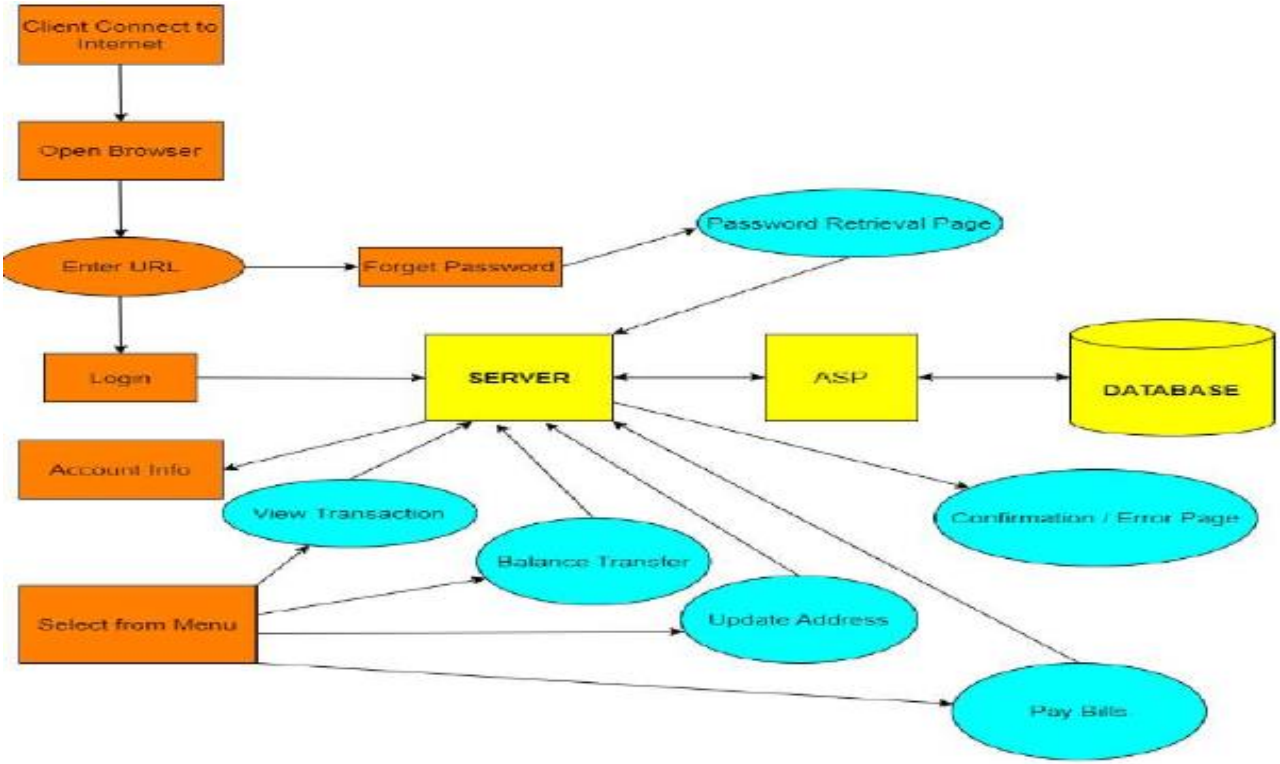
Use case Model Description-

User Case Selection	Description
Use Case Name	Users add accounts and can do all the traditional banking transactions
Level	Sub-Functional Level
Primary Actor	Users, Bank
Stakeholders and Interests	**Users:** want to register their bank accounts in the system and get access to all the online banking facilities **Banks:** want to have access to all the transaction data and maintain all the exchanges that are happening and keep an eye on all the user activities on the system Maintain a record of all the transactions and activities **Bank Administrator:** responsible for the smooth running of the systems and deal any bugs and problems that arise. Maintain a smooth transaction experience for both the bank and the users.
Pre-Condition	Banks have created their systems and done all registrations properly Users have a bank account in a particular bank and have added their account to the system
Post-Condition	All the records have been added properly
Main Success Scenario	1. Users can perform all the transactions properly and have access to all the online banking services 2 Users can sign up to add their bank accounts and get registered online 3. Users create their Login password/MPIN 4. Banks have created the server properly and have given users the required access 5. Banks maintain the transaction servers properly without fail 6 Users have given correct details
Alternative Flow	1. User installs the application on their device 2. They log in and register. 3. Registration fails and an error occurs 4. The User reports the problem to their bank and the bank resolves the issue as soon as possible
Specific Reports	1. The response time for registration is within limits 2. Response time for login is within limits 3. Transactions are happening without lags and connection drops

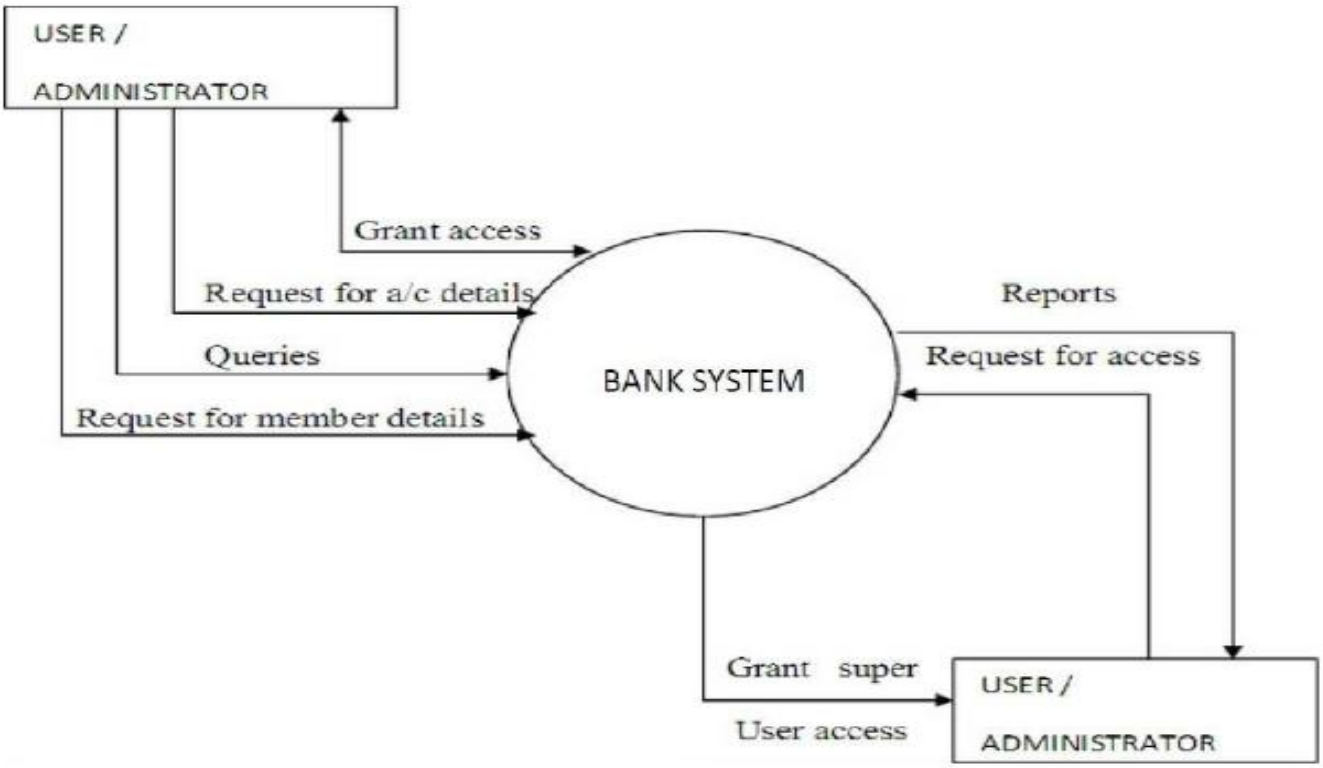
Use-case Diagram-



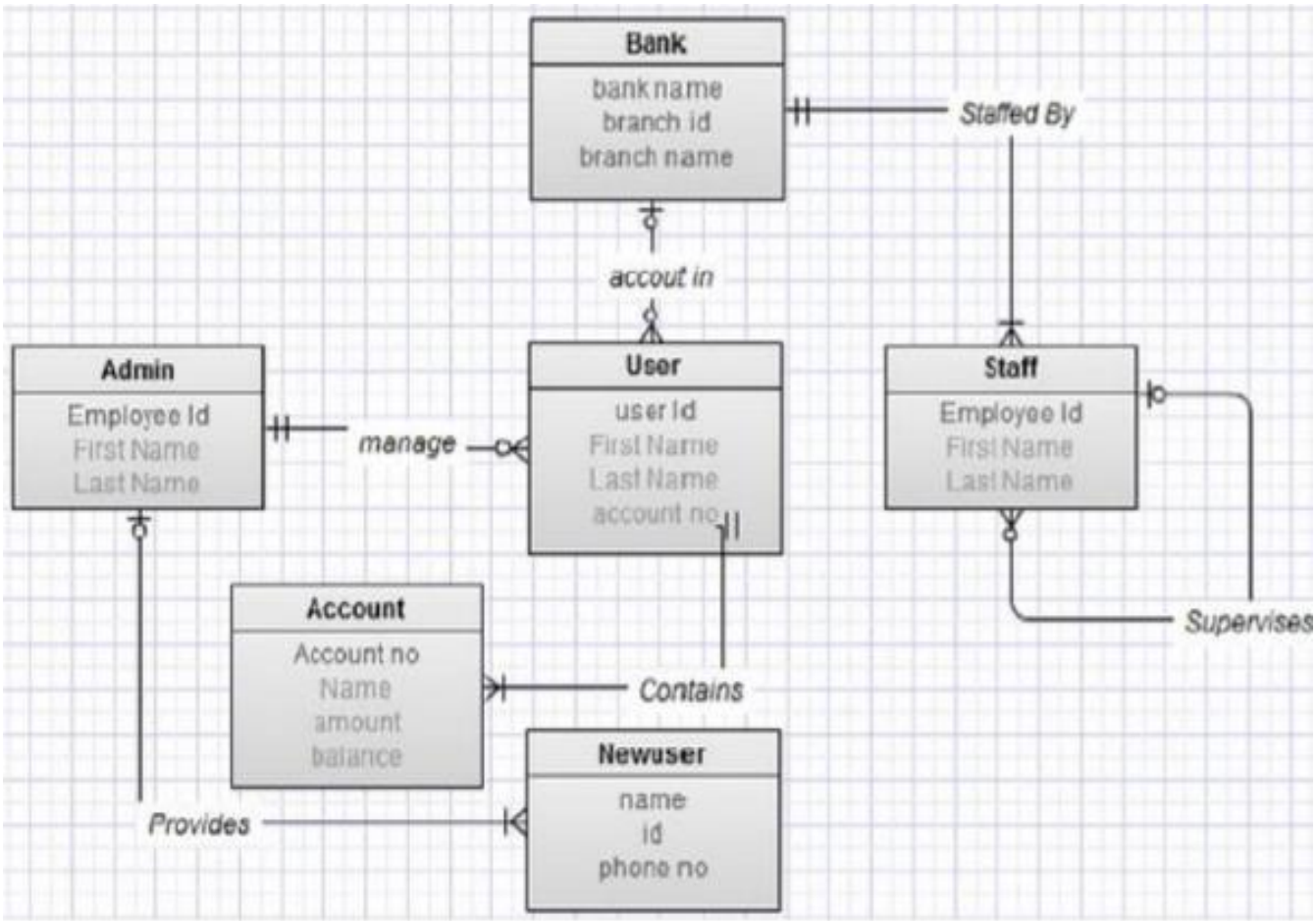
Activity Diagram-



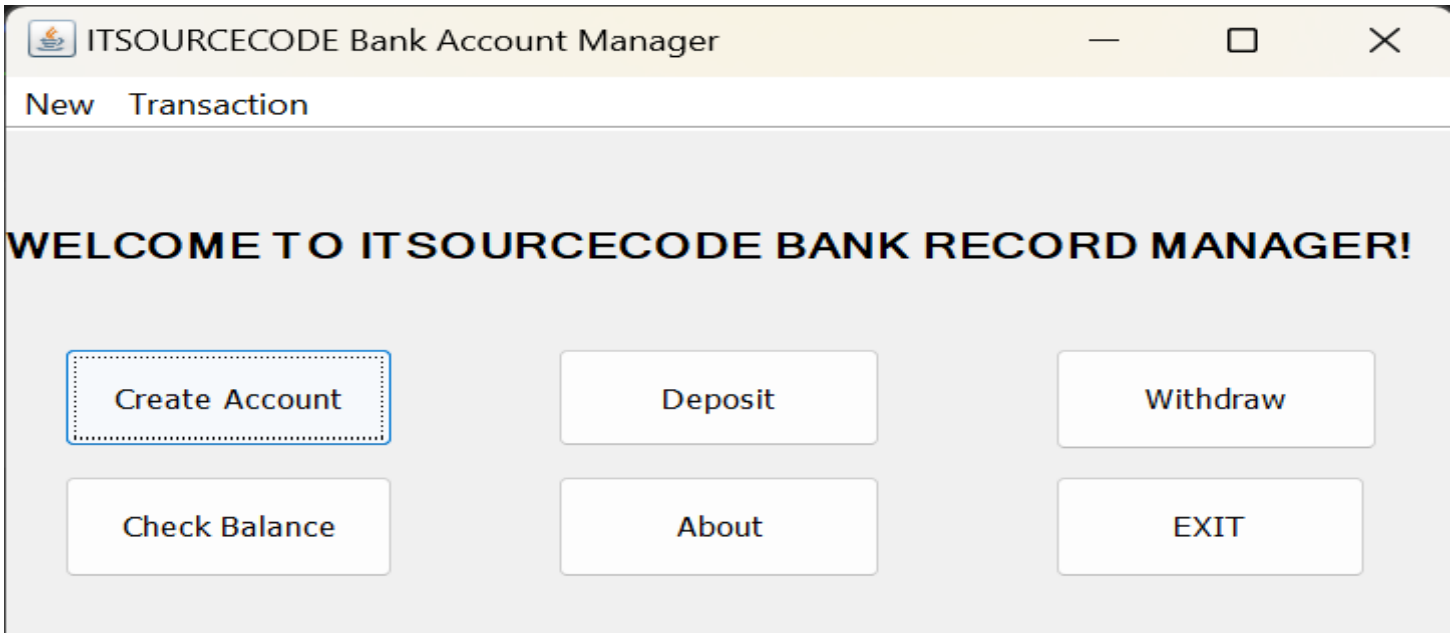
DFD Diagram-



Class Diagram-



Implementation-



Codes-

```
package BankAccount;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

public class BankMain extends javax.swing.JFrame {

public BankMain() {

initComponents(); }

@SuppressWarnings("unchecked")

private void initComponents() {

jButton1 = new JButton();

jLabel1 = new JLabel();

jButton2 = new JButton();

jButton3 = new JButton();

jButton4 = new JButton();

jButton6 = new JButton();

jButton7 = new JButton();

jMenuBar1 = new JMenuBar();

jMenu1 = new JMenu();jMenuItem1 = new JMenuItem();

jSeparator1 = new JPopupMenu.Separator();

jMenuItem3 = new JMenuItem();

jMenuItem4 = new JMenuItem();

jMenu2 = new JMenu();

jMenuItem5 = new JMenuItem();

jMenuItem6 = new JMenuItem();
```

```
jMenuItem7 = new JMenuItem();

setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

setTitle("ITSOURCECODE Bank Account Manager");

jButton1.setText("Create Account");

jButton1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jButton1ActionPerformed(evt); } });

jLabel1.setFont(new Font("Microsoft Sans Serif", 1, 15)); // NOI18N

jLabel1.setText("WELCOME TO ITSOURCECODE BANK RECORD MANAGER!");

jButton2.setText("Deposit");

jButton2.setMaximumSize(new Dimension(107, 23));

jButton2.setMinimumSize(new Dimension(107, 23));

jButton2.setPreferredSize(new Dimension(107, 23));

jButton2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jButton2ActionPerformed(evt); }  });

jButton3.setText("Withdraw");

jButton3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jButton3ActionPerformed(evt); }  });

jButton4.setText("Check Balance");

jButton4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jButton4ActionPerformed(evt);  } });

jButton6.setText("About");

jButton6.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jButton6ActionPerformed(evt); }  });

jButton7.setText("EXIT");

jButton7.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jButton7ActionPerformed(evt);  }  });

jMenu1.setText("New");

jMenuItem1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N, InputEvent.CTRL_MASK));

jMenuItem1.setText("New Account..");

jMenuItem1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jMenuItem1ActionPerformed(evt); }  });

jMenu1.add(jMenuItem1);

jMenu1.add(jSeparator1);

jMenuItem3.setText("About");

jMenuItem3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

jMenuItem3ActionPerformed(evt);  }  });

jMenu1.add(jMenuItem3);

jMenuItem4.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4, InputEvent.ALT_MASK));

jMenuItem4.setText("Exit");

jMenuItem4.addActionListener(new ActionListener() {
```

```
ublic void actionPerformed(ActionEvent evt) {

jMenuItem4ActionPerformed(evt);  }  ));

jMenu1.add(jMenuItem4);

jMenuBar1.add(jMenu1);

jMenu2.setText("Transaction");

jMenuItem5.setText("Deposit Amount..");

jMenuItem5.addActionListener(new ActionListener() {

ublic void actionPerformed(ActionEvent evt) {

jMenuItem5ActionPerformed(evt);  }  ));

jMenu2.add(jMenuItem5);

jMenuItem6.setText("Withdraw Amount..");

jMenuItem6.addActionListener(new ActionListener() {

ublic void actionPerformed(ActionEvent evt) {

jMenuItem6ActionPerformed(evt);  }  ));

jMenu2.add(jMenuItem6);

jMenuItem7.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_B, InputEvent.CTRL_MASK));

jMenuItem7.setText("Check Account Balance..");

jMenuItem7.addActionListener(new ActionListener() {

ublic void actionPerformed(ActionEvent evt) {

jMenuItem7ActionPerformed(evt);  } ));

jMenu2.add(jMenuItem7);

jMenuBar1.add(jMenu2);

setJMenuBar(jMenuBar1);

GroupLayout layout = new GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(20, 20, 20)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)

.addComponent(jButton4, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addComponent(jButton1, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addGap(55, 55, 55)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING)

.addComponent(jButton2, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)

.addComponent(jButton6, GroupLayout.PREFERRED_SIZE, 107, GroupLayout.PREFERRED_SIZE))

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(jButton3, GroupLayout.PREFERRED_SIZE, 107, GroupLayout.PREFERRED_SIZE)

.addComponent(jButton7, GroupLayout.PREFERRED_SIZE, 103, GroupLayout.PREFERRED_SIZE))

.addGap(26, 26, 26))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1, GroupLayout.DEFAULT_SIZE, 472, Short.MAX_VALUE)

.addContainerGap())  );

layout.setVerticalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1, GroupLayout.PREFERRED_SIZE, 72, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
.addComponent(jButton3, GroupLayout.Alignment.TRAILING, GroupLayout.PREFERRED_SIZE, 41, GroupLayout.PREFERRED_SIZE)
.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
.addComponent(jButton1, GroupLayout.PREFERRED_SIZE, 40, GroupLayout.PREFERRED_SIZE)
.addComponent(jButton2, GroupLayout.PREFERRED_SIZE, 40, GroupLayout.PREFERRED_SIZE)))
.addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
.addComponent(jButton4, GroupLayout.DEFAULT_SIZE, 41, Short.MAX_VALUE)
.addComponent(jButton6, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(jButton7, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addGap(110, 110, 110))  );
pack(); }
private static Account acc;
private void jMenuItem6ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jMenuItem6ActionPerformed
try {
String num;
num = JOptionPane.showInputDialog(null, "Enter a Amount To Withdraw:", "Withdraw Amount", 1);
double num1 = Double.parseDouble(num);
acc.withdraw(num1);  }
catch(BankAccountException | NumberFormatException | NullPointerException ex)  {
JOptionPane.showMessageDialog(null, ex, "Error", 2);  } }
private void jButton2ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
try {
String num;
num = JOptionPane.showInputDialog(null, "Enter a Value To Deposit:", "Deposit Amount", 1);
double num1 = Double.parseDouble(num);
acc.deposit(num1);  }
catch(NumberFormatException | NullPointerException nfe1)  {
JOptionPane.showMessageDialog(null, nfe1, "Error", 2);    } }
private void jMenuItem1ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jMenuItem1ActionPerformed
String name;
long num;
name = JOptionPane.showInputDialog(null, "Please Enter Account Holder Name:", "Create Account", 1);
num = 1332522123;
acc = new Account(name, num);
JOptionPane.showMessageDialog(null, "Account Succesfully Created!", "Create Account", 1); }
private void jButton5ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jButton5ActionPerformed
JOptionPane.showMessageDialog(null,"Created By ITSOURCECODE", "About", 1);}
private void jMenuItem3ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jMenuItem3ActionPerformed
JOptionPane.showMessageDialog(null,"Created By ITSOURCECODE", "About", 1);}
private void jMenuItem4ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jMenuItem4ActionPerformed
System.exit(0);}
private void jMenuItem5ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jMenuItem5ActionPerformed
try  {
String num;
num = JOptionPane.showInputDialog(null, "Enter a Value To Deposit:", "Deposit Amount", 1);
double num1 = Double.parseDouble(num);
acc.deposit(num1);  }
```

```
catch(NumberFormatException | NullPointerException nfe1) {

JOptionPane.showMessageDialog(null, nfe1, "Error", 2);  }}

private void jMenuItem7ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jMenuItem7ActionPerformed

try  {

double num = acc.getBalance();

JOptionPane.showMessageDialog(null, "Current Balance: " + num, "Current Balance", 1);  }

catch(NullPointerException npe) {

JOptionPane.showMessageDialog(null, npe, "Error", 2);  }}

private void jButton1ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed

String name;

long num;

name = JOptionPane.showInputDialog(null, "Please Enter Account Holder Name:", "Create Account", 1);

num = 1332522123;

acc = new Account(name, num);

JOptionPane.showMessageDialog(null, "Account Succesfully Created!", "Create Account", 1);}

private void jButton3ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jButton3ActionPerformed

try  {

String num;

num = JOptionPane.showInputDialog(null, "Enter a Amount To Withdraw:", "Withdraw Amount", 1);

double num1 = Double.parseDouble(num);

acc.withdraw(num1);   }

catch(BankAccountException | NumberFormatException | NullPointerException ex)     {

JOptionPane.showMessageDialog(null, ex, "Error", 2);     }}

private void jButton4ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jButton4ActionPerformed

try  {

double num = acc.getBalance();

JOptionPane.showMessageDialog(null, "Current Balance: " + num, "Current Balance", 1);  }

catch(NullPointerException npe)  {

JOptionPane.showMessageDialog(null, npe, "Error", 2); }}

private void jButton7ActionPerformed(ActionEvent evt) {//GEN-FIRST:event_jButton7ActionPerformed

System.exit(0);}

static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Windows".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break; } }

 catch (ClassNotFoundException | InstantiationException | IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(BankMain.class.getName()).log(java.util.logging.Level.SEVERE, null, ex); }

java.awt.EventQueue.invokeLater(() -> {

new BankMain().setVisible(true); }); }

private JButton jButton1;

private JButton jButton2;

private JButton jButton3;

private JButton jButton4;

private JButton jButton6;

private JButton jButton7;

private JLabel jLabel1;
```

Testing

Unit testing

- login/logout functional testing
- registration testing
- DB connection and migration testing
- API testing

Integration testing

- Fronted functional testing
- Backend functional testing
- Transaction testing
- CORS testing

Conclusion

The web application developed serves services required by customers to cater to their banking needs in the most reliable and safe way. Various optimization has been done in the software to intelligently use hardware and efficiently serve the client. Safety and security has been taken care at each stage of development. The banking application is served in the web platform for maximum reach and compatibility.