**JAVA SWING BASED – Fingerprint based ATM system(using aadhar Linking)-**

**SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION TECHNOLOGY**

**By**

**H.Nagaraju <1602-21-737-035>**

**Under the guidance of Ms B. Leelavathy**

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2022-2023**

**BONAFIDE CERTIFICATE**

This is to certify that this project report titled

‘***Fingerprint based ATM system***’

is a project work of ***H.Nagaraju***

bearing roll no. 1602-21-737-035

who carried out

this project under my supervision

in the IV semester

of the academic year 2022- 2023

Signature Signature

External Examiner Internal Examiner

**DATABASE MANAGEMENT SYSTEM**

ASSIGNMENT-3

**PROJECT TITLE:** ’FINGERPRINT BASED ATM(LINKING ADHARCARD)’

**ABSTARCT**:

The Fingerprint Based ATM System is an innovative project that aims to provide a secure and convenient way for customers to access their bank accounts. The system uses fingerprint scanning technology to verify the identity of the customer and link their Aadhaar card to their bank account. By doing so, the system eliminates the need for ATM cards and PINs, which can be lost or stolen, and provides a more secure method of accessing bank accounts.

The system requires the creation of a customer database that contains customer information such as name, address, contact number, Aadhaar card number, fingerprint, and PIN. The customer database is linked to the bank account database, which contains information such as account number, account type, and balance. When a customer visits an ATM machine, they can access their bank account by scanning their fingerprint, which is compared with the fingerprint stored in the customer database. If the fingerprint is a match, the system verifies the customer's identity and provides access to their bank account.

The system also includes an ATM machine database that contains information about the location of the ATM machine, the number of ATM machines available, and other details. The system also includes a bank database that contains information about the bank, such as the bank name, branch address, contact number, and the number of employees.

Overall, the Fingerprint Based ATM System is a secure and convenient method for customers to access their bank accounts without the need for ATM cards and PINs. By using fingerprint scanning technology and linking Aadhaar cards to bank accounts, the system provides a more secure and efficient way to access bank accounts.

**REQUIREMENTS:**

Overall 5 tables are required for the ER model representation.They are:

1.Customer

2.Account

3.Transaction

4.ATM machine

5.Bank table

**Table 1: CUSTOMER**

**ATTRIBUTES:**

-customer ID(integer,primary key)

-Name(varchar)

-Address(string)

-Contact Number(string)

-Adhar Number(string,Unique key)

-Fingerprint Data

-Pin(integer)

**Domain Types:**

-Customer ID,pin:integer

-Address ,contact number,adhar number(string)

**Constraints:**

-Primary key:Customer ID

-Unique key:adhar card

**Table 2: ACCOUNT**

**Attributes:**

-Account Number(integer,primary key)

-Customer ID(integer,foreign key)

-Account Type(varchar)

-Balance(int)

**Domain type:**

**-**Account number,customer ID,balance:integer

-Account type:varchar

**Constraints:**

-customer ID:foreign key

**Table 3:TRANSACTION**

**Attributes:**

-Transaction ID(primary key)

-Account number(Foreign key)

-Transaction Type(Deposit/Withdraw)

-Transaction Amount(integer)

-transaction Date/Time(date)

**Domain Type:**

-Transaction ID,Acc number,Transaction Amount:integer

-Transaction Type:varchar

**Constraints:**

**-**Acc number,ATM ID:Foreign key

**Table 4:ATM Machine Table**

**Attributes:**

-ATM ID(integer,primary key)

-Location(string)

-Number of ATM machines(integer)

**Domain Type:**

-ATM ID,Number of ATM machines:integer

-Location:varchar

**Constraints:**

-ATM ID:Primary key

**Table 5:BANK TABLE**

**Attributes:**

-Bank ID(integer,primary key)

-Bank name

-Branch address

-Contact Number(varchar)

-Number of employees(integer)

**Domain Type:**

-Bank ID,Number of employees:integer

**Constraints:**

**-**Bank ID :primary key

**Mapping Cardinalities:**

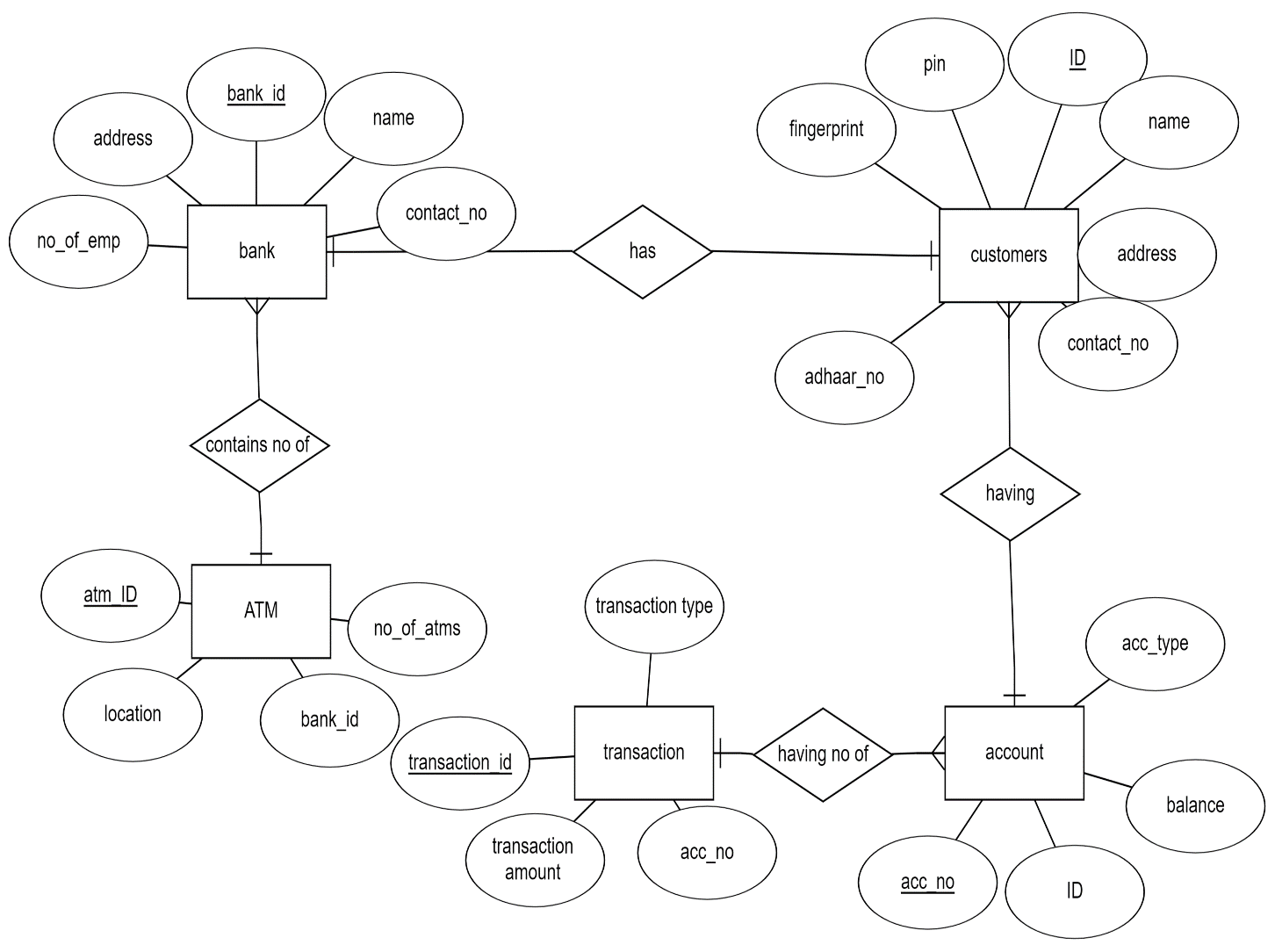
- One Customer can have multiple Accounts(one to many)

- One Account can have multiple Transactions(one to many)

- One Bank can have multiple ATMs(one to many)

- One Customer can belong to only one Bank(one to one)

**ER DIAGRAM:**

****

**DDL COMMANDS:**

**CUSTOMER TABLE:**

**a)create():**

SQL>create table customer(

2 customer ID integer,

3 name varchar,

4 address string,

5 contact number string,

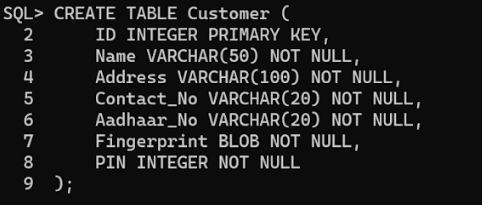
6 adhar number string,

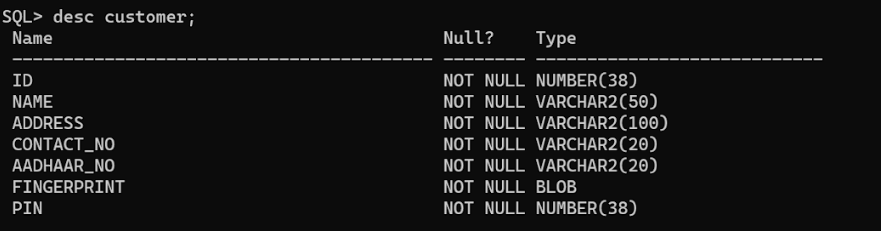
7 fingerprint data,

8 pin integer,

9 Primary key(Customer ID)

10 unique key(adhar card number);





**ACCOUNT TABLE:**

SQL>create table Account(

2 Account Number integer,

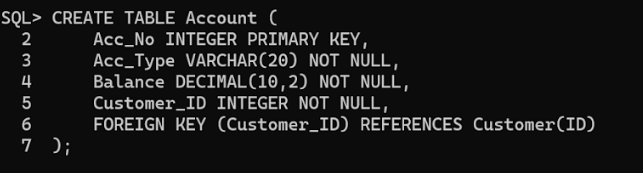
3 Customer ID integer,

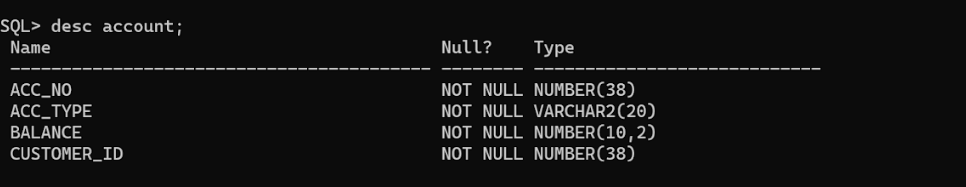
4 Account type varchar,

5 Balance integer,

6 Primary Key(Account number),

7 Foreign Key(Customer ID);



****

**TRANSACTION TABLE:**

SQL>create table Transaction(

2 Transaction ID integer,

3 Account number integer,

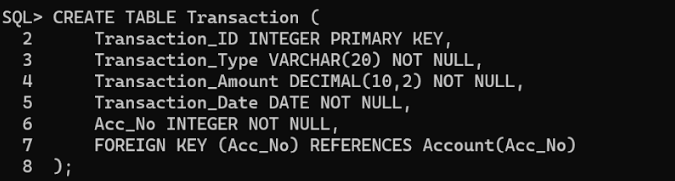
4 Transaction type deposit/withdraw,

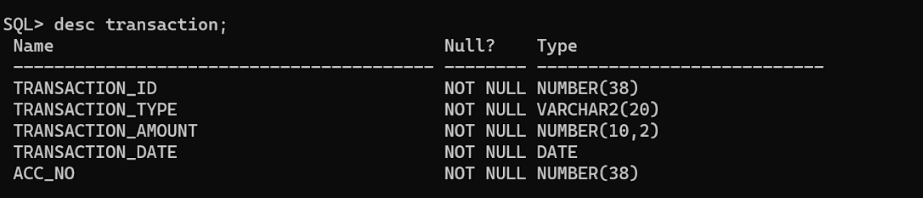
5 Transaction amount integer,

6 Transaction date date,

7 Primary key(Transaction ID),

8 Foreign key(Account number);





**ATM MACHINE TABLE:**

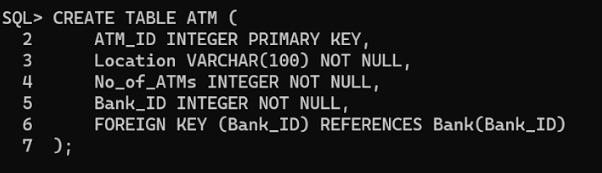
SQL>create table ATM (

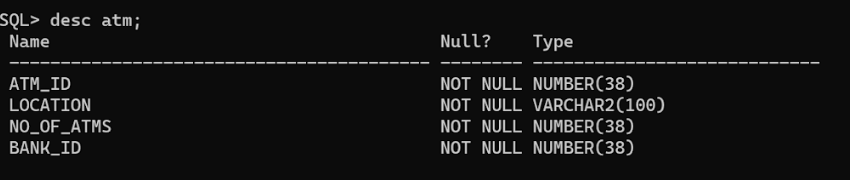
2 ATM ID integer,

3 Location string,

4 Number of ATM’s integer,

5 Primary key(ATM ID);





**BANK TABLE:**

SQL>ctreate table Bank(

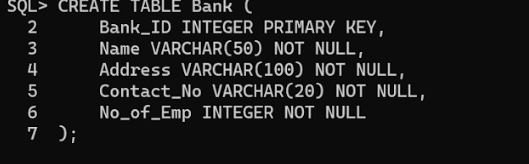
2 Bank ID integer,

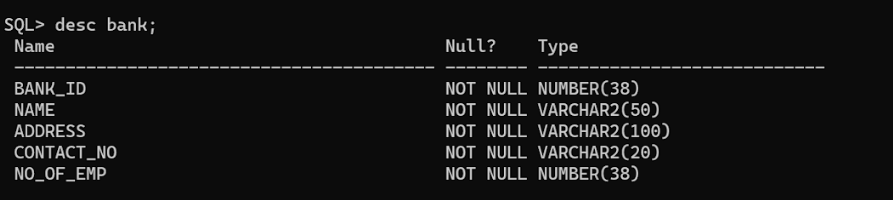
3 Bank Name varchar,

4 Contact Number varchar,

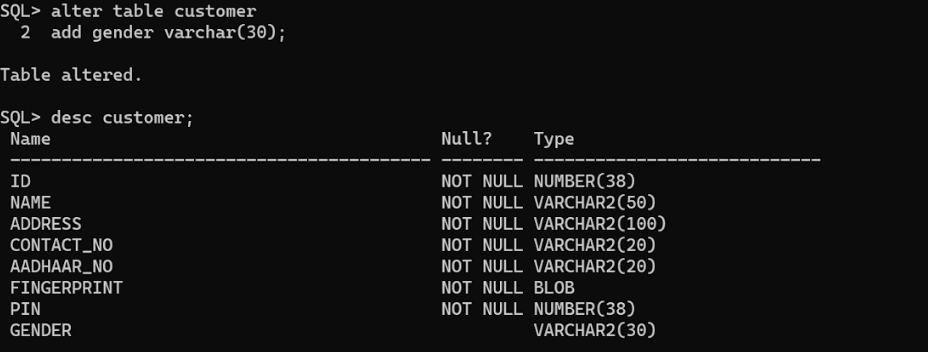
5 Number of Employees integer,

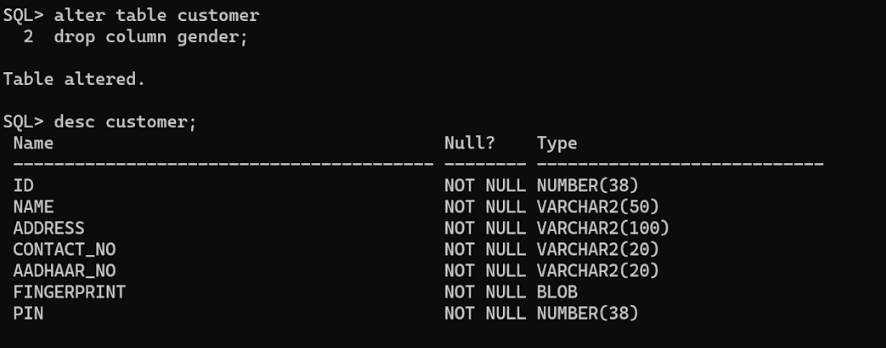
6 Primary key(Bank ID);





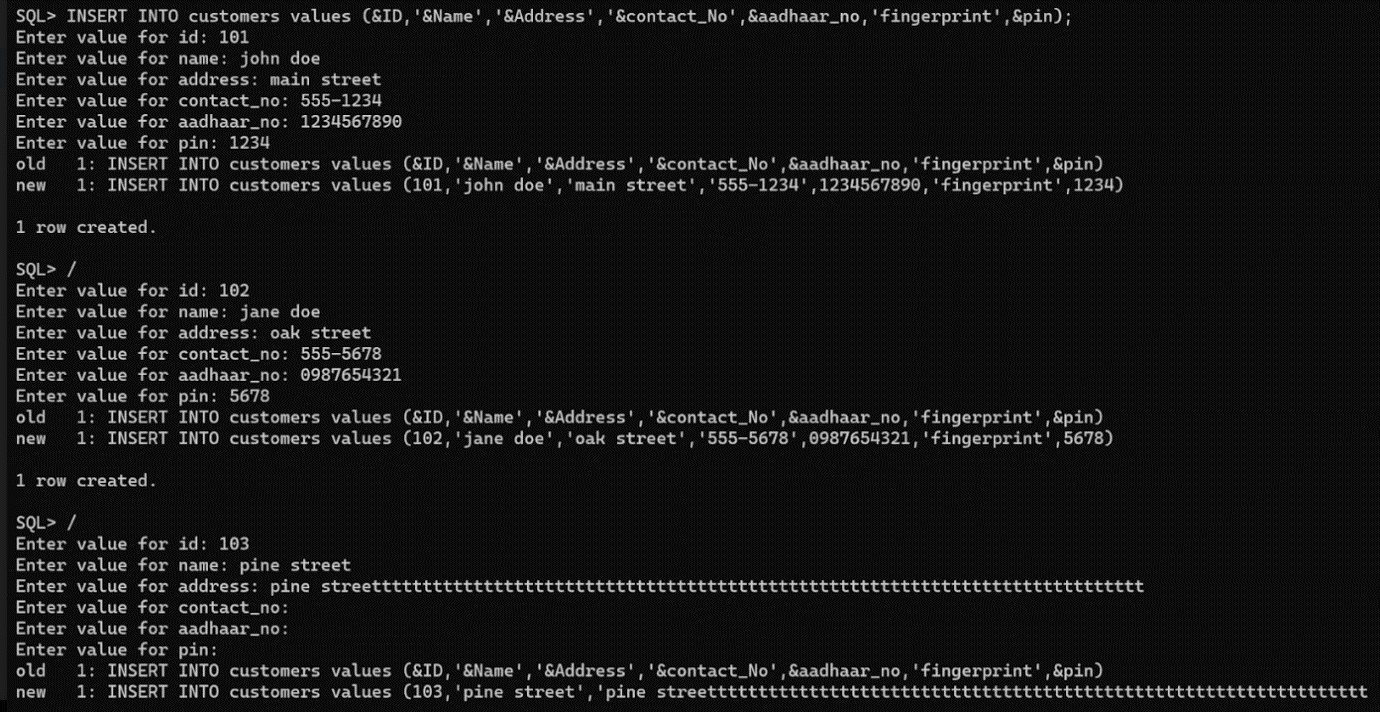
**b)alter():**

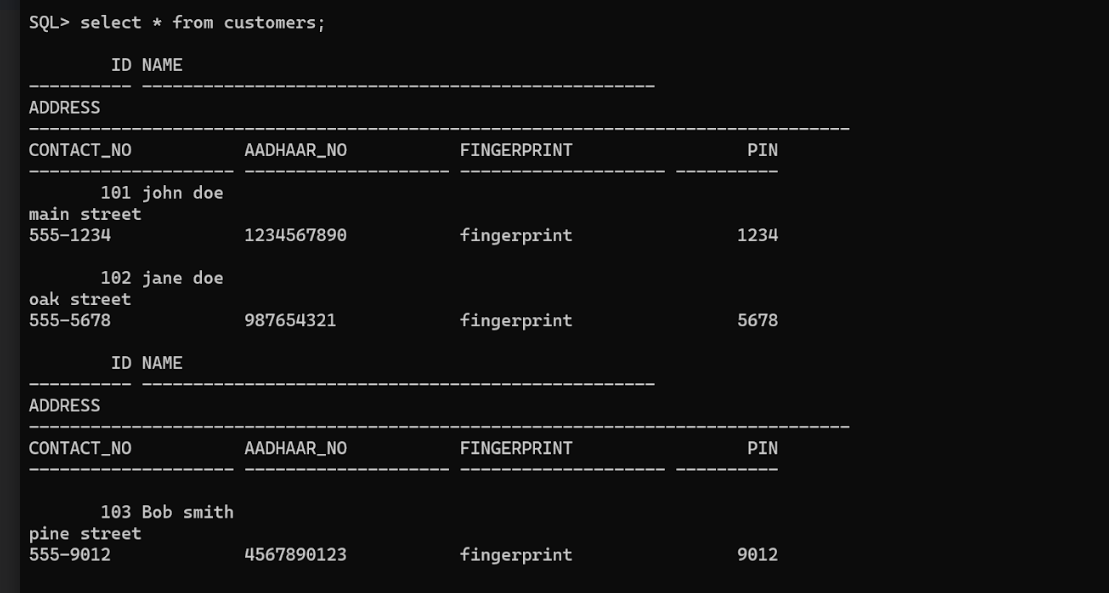
****

**c)drop():**

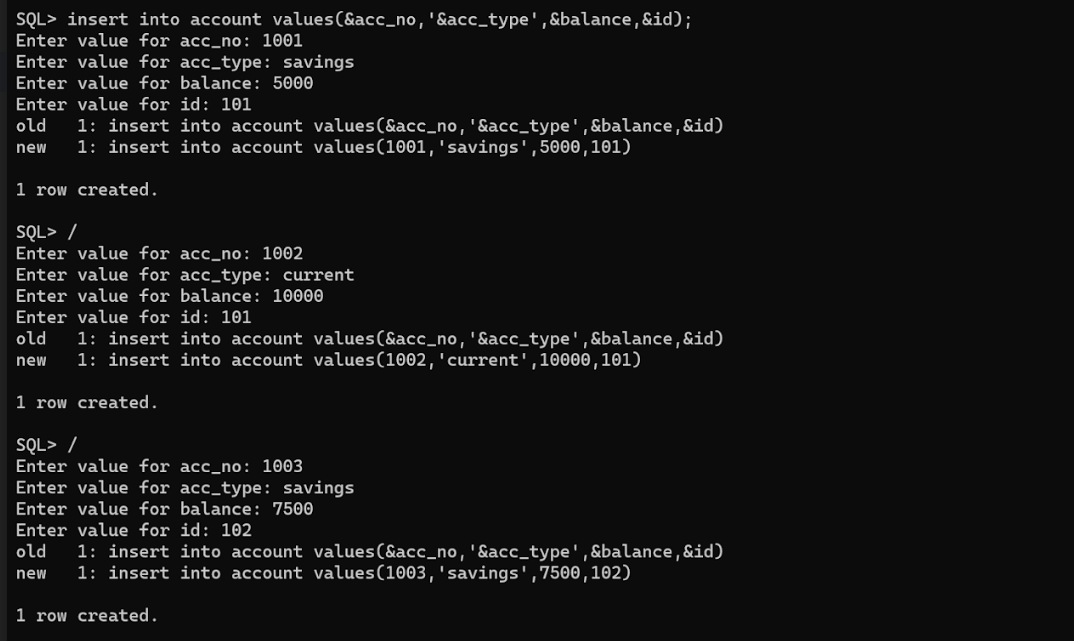
**DML COMMANDS:**

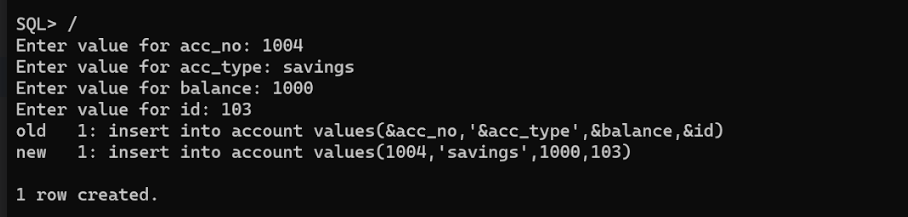
**CUSTOMER TABLE:**

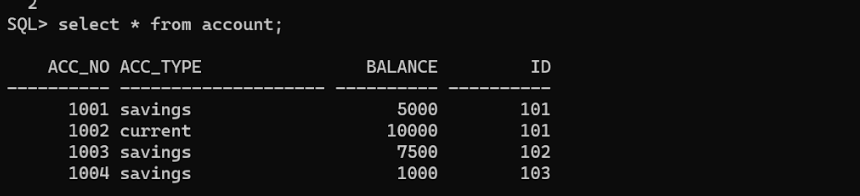
****

****

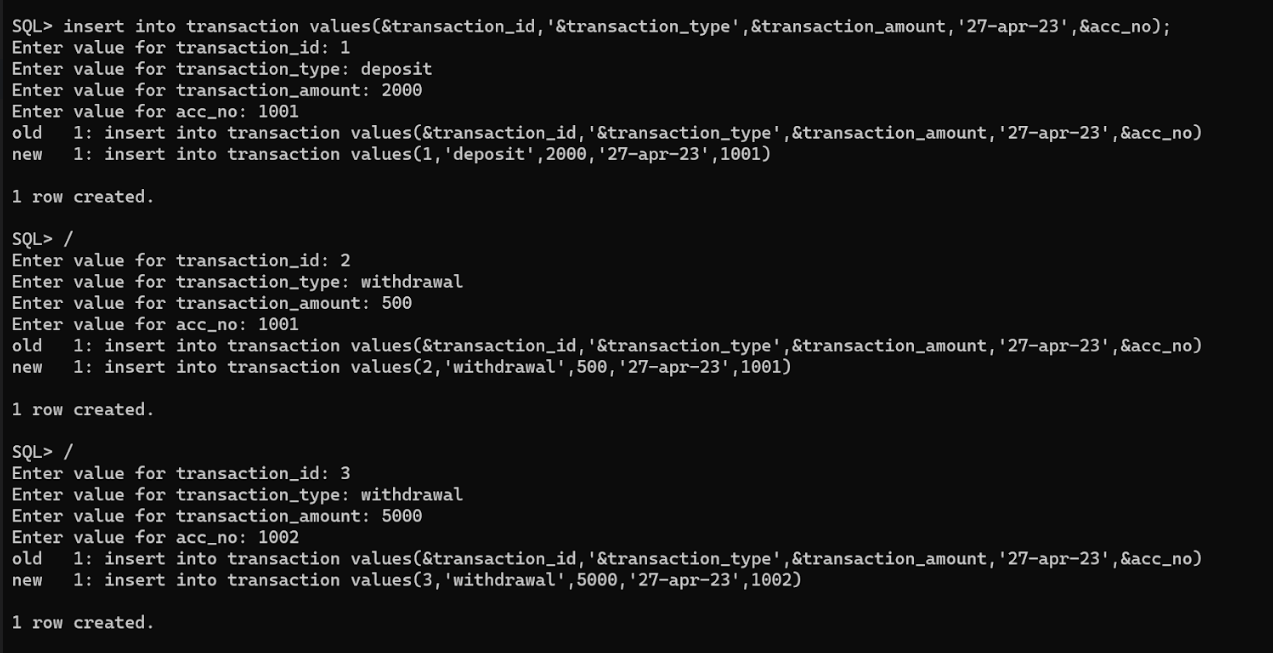
**ACCOUNT TABLE:**

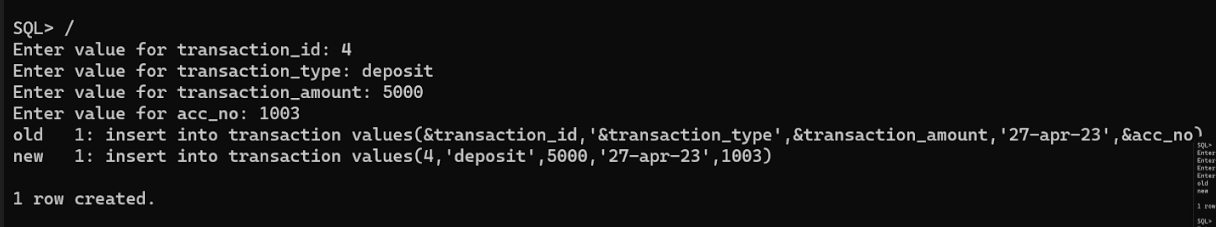
****

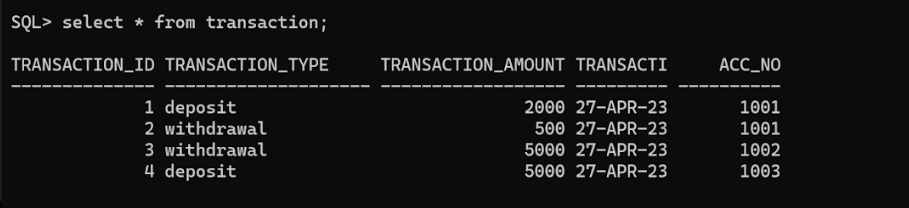
****

****

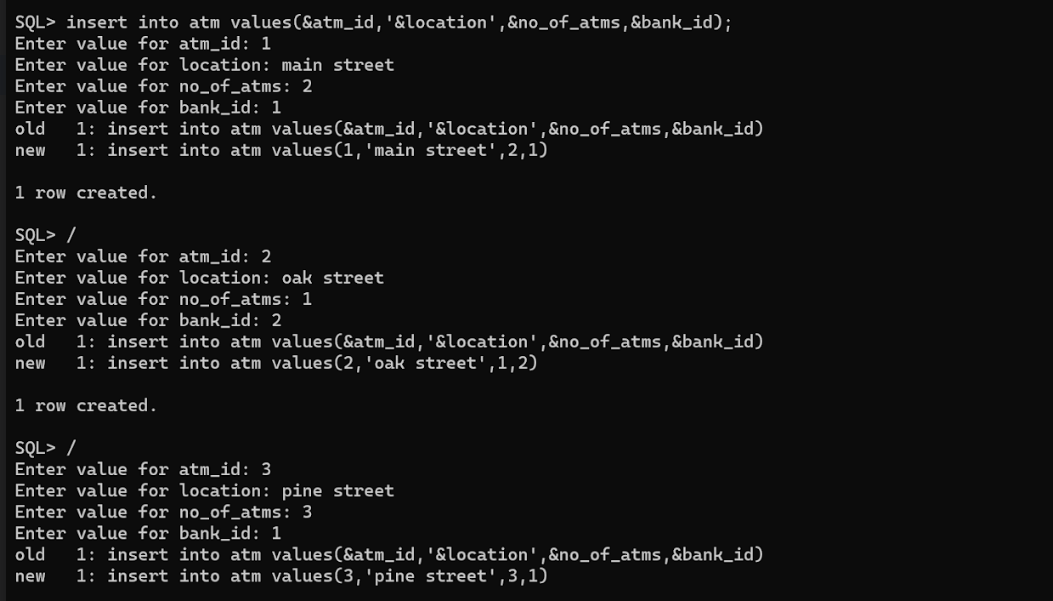
**TRANSACTION TABLE:**

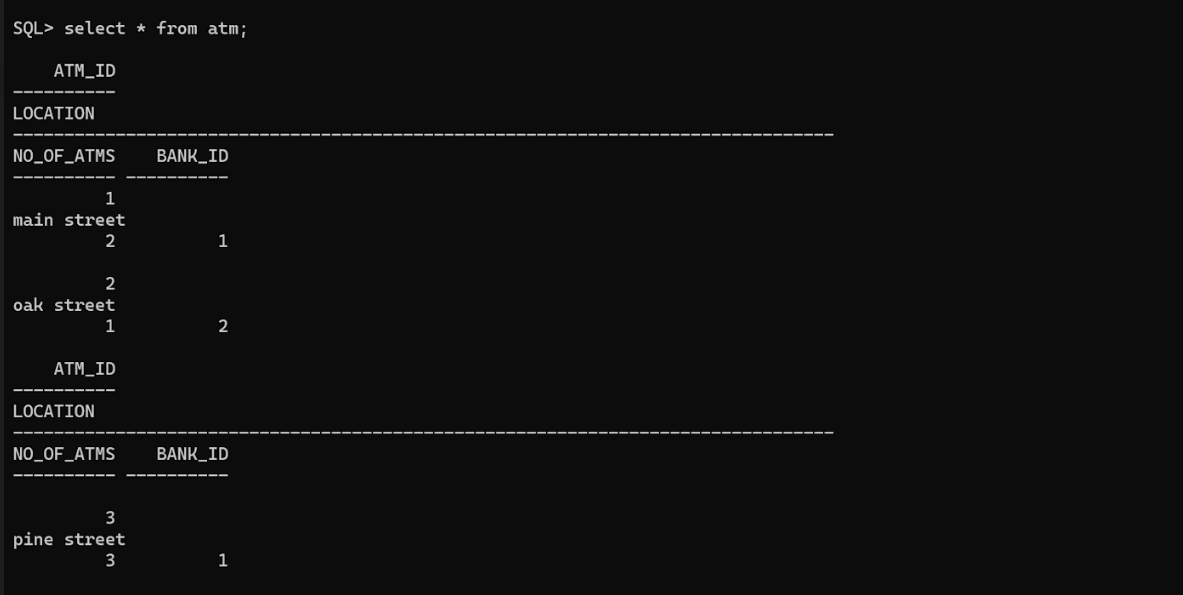
****

****

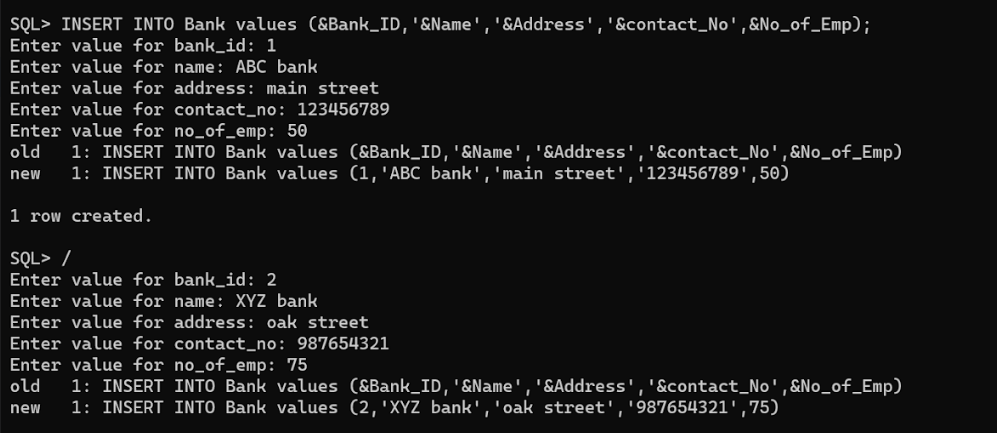
****

**ATM TABLE:**

****

****

**BANK TABLE:**

****

****

**d)UPDATE():**

****

**IMPLEMENTATION**

JAVA\_SQL Connectivity using JDBC

package main;

import java.sql.\*;

public class ConnectionManager (

private static String url = "jdbc:oracle: thin:@localhost:1521:xe"; private static String username = "nagaraju";

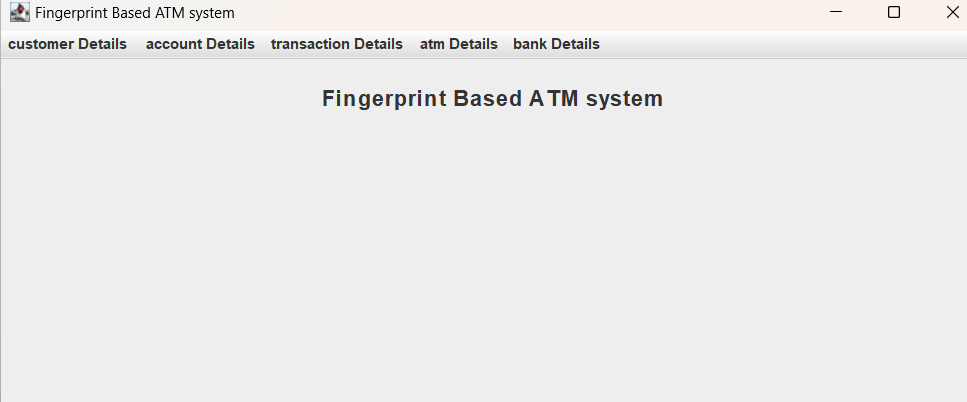
private static String password = "nagaraju";

private static Connection con;

public static Connection getConnection () throws Exception [ con = DriverManager.getConnection (url, username, password);

return con;

main page:



**package** fingerprint;

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.\*;

**public** **class** MainPage **extends** JFrame {

/\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

//private JButton retrieveMarksButton;

**public** MainPage() {

// Set frame properties

setTitle("Fingerprint Based ATM system");

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

// Create label

JLabel welcomeLabel = **new** JLabel("Fingerprint Based ATM system");

welcomeLabel.setFont(**new** Font("Arial", Font.***BOLD***, 18));

welcomeLabel.setHorizontalAlignment(SwingConstants.***CENTER***);

welcomeLabel.setBorder(BorderFactory.*createEmptyBorder*(20, 0, 20, 0));

add(welcomeLabel, BorderLayout.***NORTH***);

// Create panel for the button

/\*JPanel buttonPanel = new JPanel();

retrieveMarksButton = new JButton("Retrieve Marks");

buttonPanel.add(retrieveMarksButton);

\*/

// Create menu bar

JMenuBar menuBar = **new** JMenuBar();

// Create menus

JMenu customerMenu = **new** JMenu("customer Details");

JMenu accountMenu = **new** JMenu(" account Details");

JMenu transactionMenu = **new** JMenu("transaction Details");

JMenu atmMenu = **new** JMenu("atm Details");

JMenu bankMenu = **new** JMenu("bank Details");

// Create menu item for student menu

JMenuItem viewcustomerDetails = **new** JMenuItem("View customer Details");

viewcustomerDetails.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**new** CustomerTableGUI();

}

});

// Create menu item for course menu

JMenuItem viewaccountDetails = **new** JMenuItem("View account Details");

viewaccountDetails.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**new** AccountTableGUI();

}

});

// Create menu item for enrollment menu

JMenuItem viewtransactionDetails = **new** JMenuItem("View transaction Details");

viewtransactionDetails.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**new** TransactionTableGUI();

}

});

// Create menu item for semester menu

JMenuItem viewatmDetails = **new** JMenuItem("View atm Details");

viewatmDetails.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**new** AtmTableGUI();

}

});

// Create menu item for grade menu

JMenuItem viewbankDetails = **new** JMenuItem("View bank Details");

viewbankDetails.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**new** BankTableGUI();

}

});

// Add menu items to respective menus

customerMenu.add(viewcustomerDetails);

accountMenu.add(viewaccountDetails);

transactionMenu.add(viewtransactionDetails);

atmMenu.add(viewatmDetails);

bankMenu.add(viewbankDetails);

// Add menus to the menu bar

menuBar.add(customerMenu);

menuBar.add(accountMenu);

menuBar.add(transactionMenu);

menuBar.add(atmMenu);

menuBar.add(bankMenu);

// Set the menu bar

setJMenuBar(menuBar);

addWindowStateListener(**new** WindowStateListener() {

**public** **void** windowStateChanged(WindowEvent e) {

**if** ((e.getNewState() & Frame.***MAXIMIZED\_BOTH***) == Frame.***MAXIMIZED\_BOTH***) {

System.***out***.println("Window maximized");

} **else** {

System.***out***.println("Window not maximized");

}

}

});

// Set frame size and visibility

setSize(800, 600);

setVisible(**true**);

}

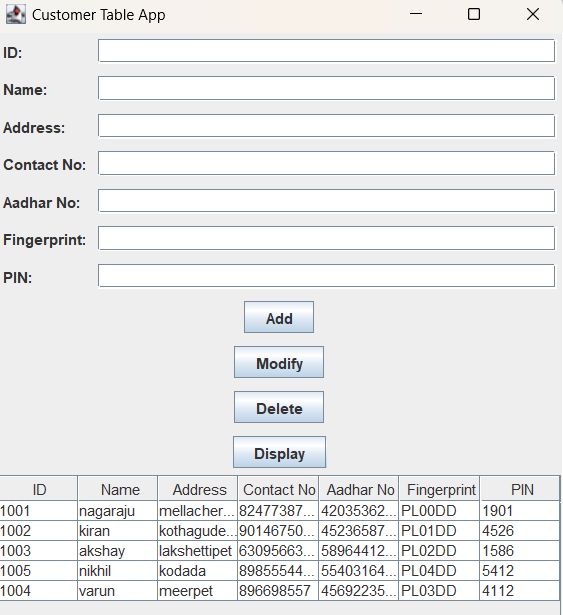
**public** **static** **void** main(String[] args) {

**new** MainPage();

}

}

**Customer page:**



**package** fingerprint;

**import** javax.swing.\*;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**import** java.sql.\*;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** CustomerTableGUI **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JTextField txtId, txtName, txtAddress, txtContactNo, txtAadharNo, txtFingerprint, txtPin;

**private** JTable tblCustomers;

**private** JButton btnAdd, btnModify, btnDelete, btnDisplay;

**private** Connection connection;

**public** CustomerTableGUI() {

initializeUI();

connectToDatabase();

displayCustomers();

}

**private** **void** initializeUI() {

txtId = **new** JTextField();

txtName = **new** JTextField();

txtAddress = **new** JTextField();

txtContactNo = **new** JTextField();

txtAadharNo = **new** JTextField();

txtFingerprint = **new** JTextField();

txtPin = **new** JTextField();

tblCustomers = **new** JTable();

tblCustomers.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

tblCustomers.getSelectionModel().addListSelectionListener(e -> selectCustomer());

JScrollPane scrollPane = **new** JScrollPane(tblCustomers);

btnAdd = **new** JButton("Add");

btnModify = **new** JButton("Modify");

btnDelete = **new** JButton("Delete");

btnDisplay = **new** JButton("Display");

JPanel panel = **new** JPanel(**new** GridBagLayout());

GridBagConstraints gbc = **new** GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.***WEST***;

gbc.insets = **new** Insets(5, 5, 5, 5);

panel.add(**new** JLabel("ID:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Name:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Address:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Contact No:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Aadhar No:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Fingerprint:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("PIN:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.***HORIZONTAL***;

gbc.weightx = 1;

panel.add(txtId, gbc);

gbc.gridy++;

panel.add(txtName, gbc);

gbc.gridy++;

panel.add(txtAddress, gbc);

gbc.gridy++;

panel.add(txtContactNo, gbc);

gbc.gridy++;

panel.add(txtAadharNo, gbc);

gbc.gridy++;

panel.add(txtFingerprint, gbc);

gbc.gridy++;

panel.add(txtPin, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.***NONE***;

gbc.anchor = GridBagConstraints.***CENTER***;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(**new** BorderLayout());

add(panel, BorderLayout.***NORTH***);

add(scrollPane, BorderLayout.***CENTER***);

btnAdd.addActionListener(e -> insertCustomer());

btnModify.addActionListener(e -> modifyCustomer());

btnDelete.addActionListener(e -> deleteCustomer());

btnDisplay.addActionListener(e -> displayCustomers());

setTitle("Customer Table App");

pack();

setLocationRelativeTo(**null**);

setVisible(**true**);

}

**private** **void** connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "nagaraju";

String password = "nagaraju";

**try** {

connection = DriverManager.*getConnection*(url, username, password);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** insertCustomer() {

String id = txtId.getText();

String name = txtName.getText();

String address = txtAddress.getText();

String contactNo = txtContactNo.getText();

String aadharNo = txtAadharNo.getText();

String fingerprint = txtFingerprint.getText();

String pin = txtPin.getText();

**try** {

String query = "INSERT INTO customer (id, name, address, contact\_no, aadhar\_no, fingerprint, pin) VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, id);

statement.setString(2, name);

statement.setString(3, address);

statement.setString(4, contactNo);

statement.setString(5, aadharNo);

statement.setString(6, fingerprint);

statement.setString(7, pin);

statement.executeUpdate();

clearFields();

displayCustomers();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** modifyCustomer() {

**int** selectedRow = tblCustomers.getSelectedRow();

**if** (selectedRow >= 0) {

String id = txtId.getText();

String name = txtName.getText();

String address = txtAddress.getText();

String contactNo = txtContactNo.getText();

String aadharNo = txtAadharNo.getText();

String fingerprint = txtFingerprint.getText();

String pin = txtPin.getText();

**try** {

String query = "UPDATE customer SET name=?, address=?, contact\_no=?, aadhar\_no=?, fingerprint=?, pin=? WHERE id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, name);

statement.setString(2, address);

statement.setString(3, contactNo);

statement.setString(4, aadharNo);

statement.setString(5, fingerprint);

statement.setString(6, pin);

statement.setString(7, id);

statement.executeUpdate();

clearFields();

displayCustomers();

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select a customer to modify.");

}

}

**private** **void** deleteCustomer() {

**int** selectedRow = tblCustomers.getSelectedRow();

**if** (selectedRow >= 0) {

String id = tblCustomers.getValueAt(selectedRow, 0).toString();

**int** option = JOptionPane.*showConfirmDialog*(**this**, "Are you sure you want to delete this customer?", "Confirmation", JOptionPane.***YES\_NO\_OPTION***);

**if** (option == JOptionPane.***YES\_OPTION***) {

**try** {

String query = "DELETE FROM customer WHERE id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, id);

statement.executeUpdate();

clearFields();

displayCustomers();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select a customer to delete.");

}

}

**private** **void** displayCustomers() {

**try** {

String query = "SELECT \* FROM customer";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Customer> customers = **new** ArrayList<>();

**while** (resultSet.next()) {

String id = resultSet.getString("id");

String name = resultSet.getString("name");

String address = resultSet.getString("address");

String contactNo = resultSet.getString("contact\_no");

String aadharNo = resultSet.getString("aadhar\_no");

String fingerprint = resultSet.getString("fingerprint");

String pin = resultSet.getString("pin");

customers.add(**new** Customer(id, name, address, contactNo, aadharNo,fingerprint, pin));

}

DefaultTableModel model = **new** DefaultTableModel();

model.setColumnIdentifiers(**new** String[]{"ID", "Name", "Address", "Contact No", "Aadhar No", "Fingerprint", "PIN"});

**for** (Customer customer : customers) {

model.addRow(**new** String[]{customer.getId(), customer.getName(), customer.getAddress(),

customer.getContactNo(), customer.getAadharNo(), customer.getFingerprint(), customer.getPin()});

}

tblCustomers.setModel(model);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** selectCustomer() {

**int** selectedRow = tblCustomers.getSelectedRow();

**if** (selectedRow >= 0) {

String id = tblCustomers.getValueAt(selectedRow, 0).toString();

String name = tblCustomers.getValueAt(selectedRow, 1).toString();

String address = tblCustomers.getValueAt(selectedRow, 2).toString();

String contactNo = tblCustomers.getValueAt(selectedRow, 3).toString();

String aadharNo = tblCustomers.getValueAt(selectedRow, 4).toString();

String fingerprint = tblCustomers.getValueAt(selectedRow, 5).toString();

String pin = tblCustomers.getValueAt(selectedRow, 6).toString();

txtId.setText(id);

txtName.setText(name);

txtAddress.setText(address);

txtContactNo.setText(contactNo);

txtAadharNo.setText(aadharNo);

txtFingerprint.setText(fingerprint);

txtPin.setText(pin);

}

}

**private** **void** clearFields() {

txtId.setText("");

txtName.setText("");

txtAddress.setText("");

txtContactNo.setText("");

txtAadharNo.setText("");

txtFingerprint.setText("");

txtPin.setText("");

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(CustomerTableGUI::**new**);

}

**private** **class** Customer {

**private** String id;

**private** String name;

**private** String address;

**private** String contactNo;

**private** String aadharNo;

**private** String fingerprint;

**private** String pin;

**public** Customer(String id, String name, String address, String contactNo, String aadharNo, String fingerprint, String pin) {

**this**.id = id;

**this**.name = name;

**this**.address = address;

**this**.contactNo = contactNo;

**this**.aadharNo = aadharNo;

**this**.fingerprint = fingerprint;

**this**.pin = pin;

}

**public** String getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** String getAddress() {

**return** address;

}

**public** String getContactNo() {

**return** contactNo;

}

**public** String getAadharNo() {

**return** aadharNo;

}

**public** String getFingerprint() {

**return** fingerprint;

}

**public** String getPin() {

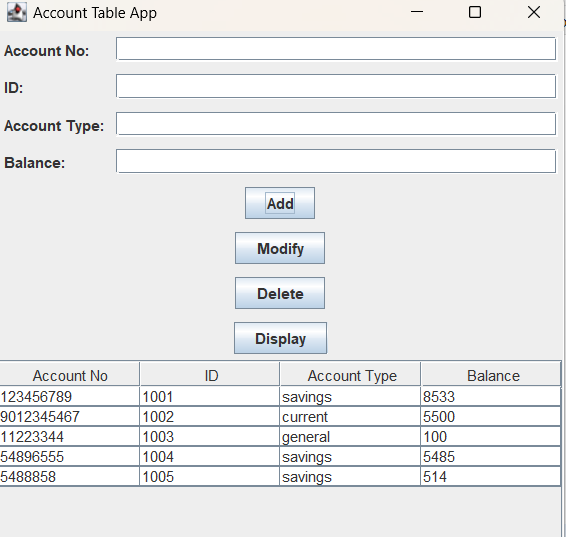
**return** pin;

}

}

}

**ACCOUNT TABLE:**



**package** fingerprint;

**import** javax.swing.\*;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**import** java.sql.\*;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** AccountTableGUI **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JTextField txtAccountNo, txtId, txtAccountType, txtBalance;

**private** JTable tblAccounts;

**private** JButton btnAdd, btnModify, btnDelete, btnDisplay;

**private** Connection connection;

**public** AccountTableGUI() {

initializeUI();

connectToDatabase();

displayAccounts();

}

**private** **void** initializeUI() {

txtAccountNo = **new** JTextField();

txtId = **new** JTextField();

txtAccountType = **new** JTextField();

txtBalance = **new** JTextField();

tblAccounts = **new** JTable();

tblAccounts.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

tblAccounts.getSelectionModel().addListSelectionListener(e -> selectAccount());

JScrollPane scrollPane = **new** JScrollPane(tblAccounts);

btnAdd = **new** JButton("Add");

btnModify = **new** JButton("Modify");

btnDelete = **new** JButton("Delete");

btnDisplay = **new** JButton("Display");

JPanel panel = **new** JPanel(**new** GridBagLayout());

GridBagConstraints gbc = **new** GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.***WEST***;

gbc.insets = **new** Insets(5, 5, 5, 5);

panel.add(**new** JLabel("Account No:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("ID:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Account Type:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Balance:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.***HORIZONTAL***;

gbc.weightx = 1;

panel.add(txtAccountNo, gbc);

gbc.gridy++;

panel.add(txtId, gbc);

gbc.gridy++;

panel.add(txtAccountType, gbc);

gbc.gridy++;

panel.add(txtBalance, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.***NONE***;

gbc.anchor = GridBagConstraints.***CENTER***;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(**new** BorderLayout());

add(panel, BorderLayout.***NORTH***);

add(scrollPane, BorderLayout.***CENTER***);

btnAdd.addActionListener(e -> insertAccount());

btnModify.addActionListener(e -> modifyAccount());

btnDelete.addActionListener(e -> deleteAccount());

btnDisplay.addActionListener(e -> displayAccounts());

setTitle("Account Table App");

pack();

setLocationRelativeTo(**null**);

setVisible(**true**);

}

**private** **void** connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "nagaraju";

String password = "nagaraju";

**try** {

connection = DriverManager.*getConnection*(url, username, password);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** insertAccount() {

String accountNo = txtAccountNo.getText();

String id = txtId.getText();

String accountType = txtAccountType.getText();

String balance = txtBalance.getText();

**try** {

String query = "INSERT INTO account (account\_no, id, account\_type, balance) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, accountNo);

statement.setString(2, id);

statement.setString(3, accountType);

statement.setString(4, balance);

statement.executeUpdate();

clearFields();

displayAccounts();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** modifyAccount() {

**int** selectedRow = tblAccounts.getSelectedRow();

**if** (selectedRow >= 0) {

String accountNo = txtAccountNo.getText();

String id = txtId.getText();

String accountType = txtAccountType.getText();

String balance = txtBalance.getText();

**try** {

String query = "UPDATE account SET id=?, account\_type=?, balance=? WHERE account\_no=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, id);

statement.setString(2, accountType);

statement.setString(3, balance);

statement.setString(4, accountNo);

statement.executeUpdate();

clearFields();

displayAccounts();

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select an account to modify.");

}

}

**private** **void** deleteAccount() {

**int** selectedRow = tblAccounts.getSelectedRow();

**if** (selectedRow >= 0) {

String accountNo = tblAccounts.getValueAt(selectedRow, 0).toString();

**int** option = JOptionPane.*showConfirmDialog*(**this**, "Are you sure you want to delete this account?", "Confirmation", JOptionPane.***YES\_NO\_OPTION***);

**if** (option == JOptionPane.***YES\_OPTION***) {

**try** {

String query = "DELETE FROM account WHERE account\_no=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, accountNo);

statement.executeUpdate();

clearFields();

displayAccounts();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select an account to delete.");

}

}

**private** **void** displayAccounts() {

**try** {

String query = "SELECT \* FROM account";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Account> accounts = **new** ArrayList<>();

**while** (resultSet.next()) {

String accountNo = resultSet.getString("account\_no");

String id = resultSet.getString("id");

String accountType = resultSet.getString("account\_type");

String balance = resultSet.getString("balance");

accounts.add(**new** Account(accountNo, id, accountType, balance));

}

DefaultTableModel model = **new** DefaultTableModel();

model.setColumnIdentifiers(**new** String[]{"Account No", "ID", "Account Type", "Balance"});

**for** (Account account : accounts) {

model.addRow(**new** String[]{account.getAccountNo(), account.getId(), account.getAccountType(), account.getBalance()});

}

tblAccounts.setModel(model);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** selectAccount() {

**int** selectedRow = tblAccounts.getSelectedRow();

**if** (selectedRow >= 0) {

String accountNo = tblAccounts.getValueAt(selectedRow, 0).toString();

String id = tblAccounts.getValueAt(selectedRow, 1).toString();

String accountType = tblAccounts.getValueAt(selectedRow, 2).toString();

String balance = tblAccounts.getValueAt(selectedRow, 3).toString();

txtAccountNo.setText(accountNo);

txtId.setText(id);

txtAccountType.setText(accountType);

txtBalance.setText(balance);

}

}

**private** **void** clearFields() {

txtAccountNo.setText("");

txtId.setText("");

txtAccountType.setText("");

txtBalance.setText("");

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(AccountTableGUI::**new**);

}

**private** **class** Account {

**private** String accountNo;

**private** String id;

**private** String accountType;

**private** String balance;

**public** Account(String accountNo, String id, String accountType, String balance) {

**this**.accountNo = accountNo;

**this**.id = id;

**this**.accountType = accountType;

**this**.balance = balance;

}

**public** String getAccountNo() {

**return** accountNo;

}

**public** String getId() {

**return** id;

}

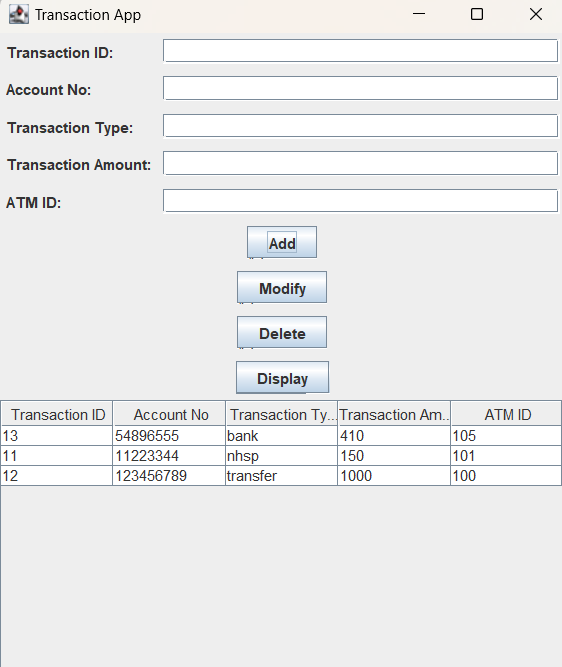
**public** String getAccountType() {

**return** accountType;

}

**public** String getBalance() {

**return** balance;

**Transaction Page:**

**package** fingerprint;

**import** javax.swing.\*;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**import** java.sql.\*;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** TransactionTableGUI **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JTextField txtTransactionId, txtAccountNo, txtTransactionType, txtTransactionAmount, txtAtmId;

**private** JTable tblTransactions;

**private** JButton btnAdd, btnModify, btnDelete, btnDisplay;

**private** Connection connection;

**public** TransactionTableGUI() {

initializeUI();

connectToDatabase();

displayTransactions();

}

**private** **void** initializeUI() {

txtTransactionId = **new** JTextField();

txtAccountNo = **new** JTextField();

txtTransactionType = **new** JTextField();

txtTransactionAmount = **new** JTextField();

txtAtmId = **new** JTextField();

tblTransactions = **new** JTable();

tblTransactions.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

tblTransactions.getSelectionModel().addListSelectionListener(e -> selectTransaction());

JScrollPane scrollPane = **new** JScrollPane(tblTransactions);

btnAdd = **new** JButton("Add");

btnModify = **new** JButton("Modify");

btnDelete = **new** JButton("Delete");

btnDisplay = **new** JButton("Display");

JPanel panel = **new** JPanel(**new** GridBagLayout());

GridBagConstraints gbc = **new** GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.***WEST***;

gbc.insets = **new** Insets(5, 5, 5, 5);

panel.add(**new** JLabel("Transaction ID:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Account No:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Transaction Type:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Transaction Amount:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("ATM ID:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.***HORIZONTAL***;

gbc.weightx = 1;

panel.add(txtTransactionId, gbc);

gbc.gridy++;

panel.add(txtAccountNo, gbc);

gbc.gridy++;

panel.add(txtTransactionType, gbc);

gbc.gridy++;

panel.add(txtTransactionAmount, gbc);

gbc.gridy++;

panel.add(txtAtmId, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.***NONE***;

gbc.anchor = GridBagConstraints.***CENTER***;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(**new** BorderLayout());

add(panel, BorderLayout.***NORTH***);

add(scrollPane, BorderLayout.***CENTER***);

btnAdd.addActionListener(e -> insertTransaction());

btnModify.addActionListener(e -> modifyTransaction());

btnDelete.addActionListener(e -> deleteTransaction());

btnDisplay.addActionListener(e -> displayTransactions());

setTitle("Transaction App");

pack();

setLocationRelativeTo(**null**);

setVisible(**true**);

}

**private** **void** connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "nagaraju";

String password = "nagaraju";

**try** {

connection = DriverManager.*getConnection*(url, username, password);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** insertTransaction() {

String transactionId = txtTransactionId.getText();

String accountNo = txtAccountNo.getText();

String transactionType = txtTransactionType.getText();

String transactionAmount = txtTransactionAmount.getText();

String atmId = txtAtmId.getText();

**try** {

String query = "INSERT INTO transaction (transaction\_id, account\_no, transaction\_type, transaction\_amount, atm\_id) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, transactionId);

statement.setString(2, accountNo);

statement.setString(3, transactionType);

statement.setString(4, transactionAmount);

statement.setString(5, atmId);

statement.executeUpdate();

clearFields();

displayTransactions();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** modifyTransaction() {

**int** selectedRow = tblTransactions.getSelectedRow();

**if** (selectedRow >= 0) {

String transactionId = txtTransactionId.getText();

String accountNo = txtAccountNo.getText();

String transactionType = txtTransactionType.getText();

String transactionAmount = txtTransactionAmount.getText();

String atmId = txtAtmId.getText();

**try** {

String query = "UPDATE transaction SET account\_no=?, transaction\_type=?, transaction\_amount=?, atm\_id=? WHERE transaction\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, accountNo);

statement.setString(2, transactionType);

statement.setString(3, transactionAmount);

statement.setString(4, atmId);

statement.setString(5, transactionId);

statement.executeUpdate();

clearFields();

displayTransactions();

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select a transaction to modify.");

}

}

**private** **void** deleteTransaction() {

**int** selectedRow = tblTransactions.getSelectedRow();

**if** (selectedRow >= 0) {

String transactionId = tblTransactions.getValueAt(selectedRow, 0).toString();

**int** option = JOptionPane.*showConfirmDialog*(**this**, "Are you sure you want to delete this transaction?", "Confirmation", JOptionPane.***YES\_NO\_OPTION***);

**if** (option == JOptionPane.***YES\_OPTION***) {

**try** {

String query = "DELETE FROM transaction WHERE transaction\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, transactionId);

statement.executeUpdate();

clearFields();

displayTransactions();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select a transaction to delete.");

}

}

**private** **void** displayTransactions() {

**try** {

String query = "SELECT \* FROM transaction";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Transaction> transactions = **new** ArrayList<>();

**while** (resultSet.next()) {

String transactionId = resultSet.getString("transaction\_id");

String accountNo = resultSet.getString("account\_no");

String transactionType = resultSet.getString("transaction\_type");

String transactionAmount = resultSet.getString("transaction\_amount");

String atmId = resultSet.getString("atm\_id");

transactions.add(**new** Transaction(transactionId, accountNo, transactionType, transactionAmount, atmId));

}

DefaultTableModel model = **new** DefaultTableModel();

model.setColumnIdentifiers(**new** String[]{"Transaction ID", "Account No", "Transaction Type", "Transaction Amount", "ATM ID"});

**for** (Transaction transaction : transactions) {

model.addRow(**new** String[]{transaction.getTransactionId(), transaction.getAccountNo(), transaction.getTransactionType(), transaction.getTransactionAmount(), transaction.getAtmId()});

}

tblTransactions.setModel(model);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** selectTransaction() {

**int** selectedRow = tblTransactions.getSelectedRow();

**if** (selectedRow >= 0) {

String transactionId =tblTransactions.getValueAt(selectedRow, 0).toString();

String accountNo = tblTransactions.getValueAt(selectedRow, 1).toString();

String transactionType = tblTransactions.getValueAt(selectedRow, 2).toString();

String transactionAmount = tblTransactions.getValueAt(selectedRow, 3).toString();

String atmId = tblTransactions.getValueAt(selectedRow, 4).toString();

txtTransactionId.setText(transactionId);

txtAccountNo.setText(accountNo);

txtTransactionType.setText(transactionType);

txtTransactionAmount.setText(transactionAmount);

txtAtmId.setText(atmId);

}

}

**private** **void** clearFields() {

txtTransactionId.setText("");

txtAccountNo.setText("");

txtTransactionType.setText("");

txtTransactionAmount.setText("");

txtAtmId.setText("");

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(TransactionTableGUI::**new**);

}

**private** **class** Transaction {

**private** String transactionId;

**private** String accountNo;

**private** String transactionType;

**private** String transactionAmount;

**private** String atmId;

**public** Transaction(String transactionId, String accountNo, String transactionType, String transactionAmount, String atmId) {

**this**.transactionId = transactionId;

**this**.accountNo = accountNo;

**this**.transactionType = transactionType;

**this**.transactionAmount = transactionAmount;

**this**.atmId = atmId;

}

**public** String getTransactionId() {

**return** transactionId;

}

**public** String getAccountNo() {

**return** accountNo;

}

**public** String getTransactionType() {

**return** transactionType;

}

**public** String getTransactionAmount() {

**return** transactionAmount;

}

**public** String getAtmId() {

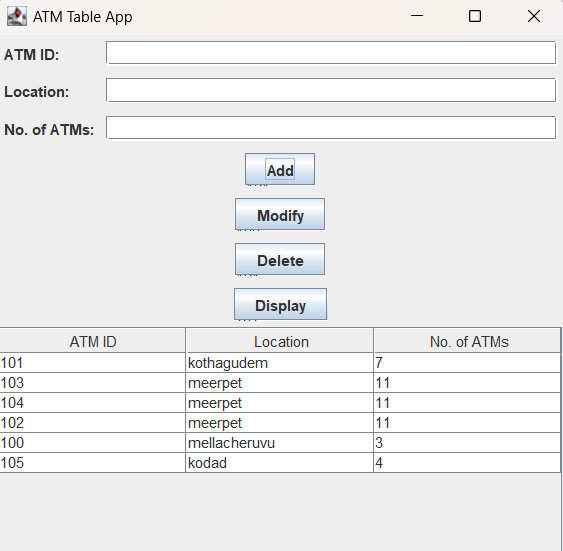
**return** atmId;

}

}

}

**ATM Page:**

****

**package** fingerprint;

**import** javax.swing.\*;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**import** java.sql.\*;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** AtmTableGUI **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JTextField txtAtmId, txtLocation, txtNoOfAtm;

**private** JTable tblAtms;

**private** JButton btnAdd, btnModify, btnDelete, btnDisplay;

**private** Connection connection;

**public** AtmTableGUI() {

initializeUI();

connectToDatabase();

displayAtms();

}

**private** **void** initializeUI() {

txtAtmId = **new** JTextField();

txtLocation = **new** JTextField();

txtNoOfAtm = **new** JTextField();

tblAtms = **new** JTable();

tblAtms.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

tblAtms.getSelectionModel().addListSelectionListener(e -> selectAtm());

JScrollPane scrollPane = **new** JScrollPane(tblAtms);

btnAdd = **new** JButton("Add");

btnModify = **new** JButton("Modify");

btnDelete = **new** JButton("Delete");

btnDisplay = **new** JButton("Display");

JPanel panel = **new** JPanel(**new** GridBagLayout());

GridBagConstraints gbc = **new** GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.***WEST***;

gbc.insets = **new** Insets(5, 5, 5, 5);

panel.add(**new** JLabel("ATM ID:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Location:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("No. of ATMs:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.***HORIZONTAL***;

gbc.weightx = 1;

panel.add(txtAtmId, gbc);

gbc.gridy++;

panel.add(txtLocation, gbc);

gbc.gridy++;

panel.add(txtNoOfAtm, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.***NONE***;

gbc.anchor = GridBagConstraints.***CENTER***;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(**new** BorderLayout());

add(panel, BorderLayout.***NORTH***);

add(scrollPane, BorderLayout.***CENTER***);

btnAdd.addActionListener(e -> insertAtm());

btnModify.addActionListener(e -> modifyAtm());

btnDelete.addActionListener(e -> deleteAtm());

btnDisplay.addActionListener(e -> displayAtms());

setTitle("ATM Table App");

pack();

setLocationRelativeTo(**null**);

setVisible(**true**);

}

**private** **void** connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "nagaraju";

String password = "nagaraju";

**try** {

connection = DriverManager.*getConnection*(url, username, password);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** insertAtm() {

String atmId = txtAtmId.getText();

String location = txtLocation.getText();

String noOfAtm = txtNoOfAtm.getText();

**try** {

String query = "INSERT INTO atm (atm\_id, location, no\_of\_atm) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, atmId);

statement.setString(2, location);

statement.setString(3, noOfAtm);

statement.executeUpdate();

clearFields();

displayAtms();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** modifyAtm() {

**int** selectedRow = tblAtms.getSelectedRow();

**if** (selectedRow >= 0) {

String atmId = txtAtmId.getText();

String location = txtLocation.getText();

String noOfAtm = txtNoOfAtm.getText();

**try** {

String query = "UPDATE atm SET location=?, no\_of\_atm=? WHERE atm\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, location);

statement.setString(2, noOfAtm);

statement.setString(3, atmId);

statement.executeUpdate();

clearFields();

displayAtms();

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select an ATM to modify.");

}

}

**private** **void** deleteAtm() {

**int** selectedRow = tblAtms.getSelectedRow();

**if** (selectedRow >= 0) {

String atmId = tblAtms.getValueAt(selectedRow, 0).toString();

**int** option = JOptionPane.*showConfirmDialog*(**this**, "Are you sure you want to delete this ATM?", "Confirmation", JOptionPane.***YES\_NO\_OPTION***);

**if** (option == JOptionPane.***YES\_OPTION***) {

**try** {

String query = "DELETE FROM atm WHERE atm\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, atmId);

statement.executeUpdate();

clearFields();

displayAtms();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select an ATM to delete.");

}

}

**private** **void** displayAtms() {

**try** {

String query = "SELECT \* FROM atm";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Atm> atms = **new** ArrayList<>();

**while** (resultSet.next()) {

String atmId = resultSet.getString("atm\_id");

String location = resultSet.getString("location");

String noOfAtm = resultSet.getString("no\_of\_atm");

atms.add(**new** Atm(atmId, location, noOfAtm));

}

DefaultTableModel model = **new** DefaultTableModel();

model.setColumnIdentifiers(**new** String[]{"ATM ID", "Location", "No. of ATMs"});

**for** (Atm atm : atms) {

model.addRow(**new** String[]{atm.getAtmId(), atm.getLocation(), atm.getNoOfAtm()});

}

tblAtms.setModel(model);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** selectAtm() {

**int** selectedRow = tblAtms.getSelectedRow();

**if** (selectedRow >= 0) {

String atmId = tblAtms.getValueAt(selectedRow, 0).toString();

String location = tblAtms.getValueAt(selectedRow, 1).toString();

String noOfAtm = tblAtms.getValueAt(selectedRow, 2).toString();

txtAtmId.setText(atmId);

txtLocation.setText(location);

txtNoOfAtm.setText(noOfAtm);

}

}

**private** **void** clearFields() {

txtAtmId.setText("");

txtLocation.setText("");

txtNoOfAtm.setText("");

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(AtmTableGUI::**new**);

}

**private** **class** Atm {

**private** String atmId;

**private** String location;

**private** String noOfAtm;

**public** Atm(String atmId, String location, String noOfAtm) {

**this**.atmId = atmId;

**this**.location = location;

**this**.noOfAtm = noOfAtm;

}

**public** String getAtmId() {

**return** atmId;

}

**public** String getLocation() {

**return** location;

}

**public** String getNoOfAtm() {

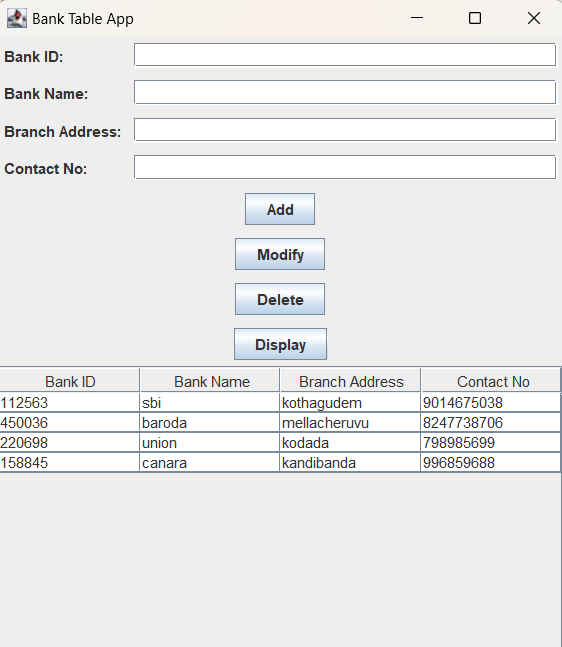
**return** noOfAtm;

}

}

}

**Bank page:**

****

**package** fingerprint;

**import** javax.swing.\*;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**import** java.sql.\*;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** BankTableGUI **extends** JFrame {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JTextField txtBankId, txtBankName, txtBranchAddress, txtContactNo;

**private** JTable tblBanks;

**private** JButton btnAdd, btnModify, btnDelete, btnDisplay;

**private** Connection connection;

**public** BankTableGUI() {

initializeUI();

connectToDatabase();

displayBanks();

}

**private** **void** initializeUI() {

txtBankId = **new** JTextField();

txtBankName = **new** JTextField();

txtBranchAddress = **new** JTextField();

txtContactNo = **new** JTextField();

tblBanks = **new** JTable();

tblBanks.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

tblBanks.getSelectionModel().addListSelectionListener(e -> selectBank());

JScrollPane scrollPane = **new** JScrollPane(tblBanks);

btnAdd = **new** JButton("Add");

btnModify = **new** JButton("Modify");

btnDelete = **new** JButton("Delete");

btnDisplay = **new** JButton("Display");

JPanel panel = **new** JPanel(**new** GridBagLayout());

GridBagConstraints gbc = **new** GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.***WEST***;

gbc.insets = **new** Insets(5, 5, 5, 5);

panel.add(**new** JLabel("Bank ID:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Bank Name:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Branch Address:"), gbc);

gbc.gridy++;

panel.add(**new** JLabel("Contact No:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.***HORIZONTAL***;

gbc.weightx = 1;

panel.add(txtBankId, gbc);

gbc.gridy++;

panel.add(txtBankName, gbc);

gbc.gridy++;

panel.add(txtBranchAddress, gbc);

gbc.gridy++;

panel.add(txtContactNo, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.***NONE***;

gbc.anchor = GridBagConstraints.***CENTER***;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(**new** BorderLayout());

add(panel, BorderLayout.***NORTH***);

add(scrollPane, BorderLayout.***CENTER***);

btnAdd.addActionListener(e -> insertBank());

btnModify.addActionListener(e -> modifyBank());

btnDelete.addActionListener(e -> deleteBank());

btnDisplay.addActionListener(e -> displayBanks());

setTitle("Bank Table App");

pack();

setLocationRelativeTo(**null**);

setVisible(**true**);

}

**private** **void** connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "nagaraju";

String password = "nagaraju";

**try** {

connection = DriverManager.*getConnection*(url, username, password);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** insertBank() {

String bankId = txtBankId.getText();

String bankName = txtBankName.getText();

String branchAddress = txtBranchAddress.getText();

String contactNo = txtContactNo.getText();

**try** {

String query = "INSERT INTO bank (bank\_id, bank\_name, branch\_address, contact\_no) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, bankId);

statement.setString(2, bankName);

statement.setString(3, branchAddress);

statement.setString(4, contactNo);

statement.executeUpdate();

clearFields();

displayBanks();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** modifyBank() {

**int** selectedRow = tblBanks.getSelectedRow();

**if** (selectedRow >= 0) {

String bankId = txtBankId.getText();

String bankName = txtBankName.getText();

String branchAddress = txtBranchAddress.getText();

String contactNo = txtContactNo.getText();

**try** {

String query = "UPDATE bank SET bank\_name=?, branch\_address=?, contact\_no=? WHERE bank\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, bankName);

statement.setString(2, branchAddress);

statement.setString(3, contactNo);

statement.setString(4, bankId);

statement.executeUpdate();

clearFields();

displayBanks();

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select a bank to modify.");

}

}

**private** **void** deleteBank() {

**int** selectedRow = tblBanks.getSelectedRow();

**if** (selectedRow >= 0) {

String bankId = tblBanks.getValueAt(selectedRow, 0).toString();

**int** option = JOptionPane.*showConfirmDialog*(**this**, "Are you sure you want to delete this bank?", "Confirmation", JOptionPane.***YES\_NO\_OPTION***);

**if** (option == JOptionPane.***YES\_OPTION***) {

**try** {

String query = "DELETE FROM bank WHERE bank\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, bankId);

statement.executeUpdate();

clearFields();

displayBanks();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

} **else** {

JOptionPane.*showMessageDialog*(**this**, "Please select a bank to delete.");

}

}

**private** **void** displayBanks() {

**try** {

String query = "SELECT \* FROM bank";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Bank> banks = **new** ArrayList<>();

**while** (resultSet.next()) {

String bankId = resultSet.getString("bank\_id");

String bankName = resultSet.getString("bank\_name");

String branchAddress = resultSet.getString("branch\_address");

String contactNo = resultSet.getString("contact\_no");

banks.add(**new** Bank(bankId, bankName, branchAddress, contactNo));

}

DefaultTableModel model = **new** DefaultTableModel();

model.setColumnIdentifiers(**new** String[]{"Bank ID", "Bank Name", "Branch Address", "Contact No"});

**for** (Bank bank : banks) {

model.addRow(**new** String[]{bank.getBankId(), bank.getBankName(), bank.getBranchAddress(), bank.getContactNo()});

}

tblBanks.setModel(model);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** selectBank() {

**int** selectedRow = tblBanks.getSelectedRow();

**if** (selectedRow >= 0) {

String bankId = tblBanks.getValueAt(selectedRow, 0).toString();

String bankName = tblBanks.getValueAt(selectedRow, 1).toString();

String branchAddress = tblBanks.getValueAt(selectedRow, 2).toString();

String contactNo = tblBanks.getValueAt(selectedRow, 3).toString();

txtBankId.setText(bankId);

txtBankName.setText(bankName);

txtBranchAddress.setText(branchAddress);

txtContactNo.setText(contactNo);

}

}

**private** **void** clearFields() {

txtBankId.setText("");

txtBankName.setText("");

txtBranchAddress.setText("");

txtContactNo.setText("");

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(BankTableGUI::**new**);

}

**private** **class** Bank {

**private** String bankId;

**private** String bankName;

**private** String branchAddress;

**private** String contactNo;

**public** Bank(String bankId, String bankName, String branchAddress, String contactNo) {

**this**.bankId = bankId;

**this**.bankName = bankName;

**this**.branchAddress = branchAddress;

**this**.contactNo = contactNo;

}

**public** String getBankId() {

**return** bankId;

}

**public** String getBankName() {

**return** bankName;

}

**public** String getBranchAddress() {

**return** branchAddress;

}

**public** String getContactNo() {

**return** contactNo;

}

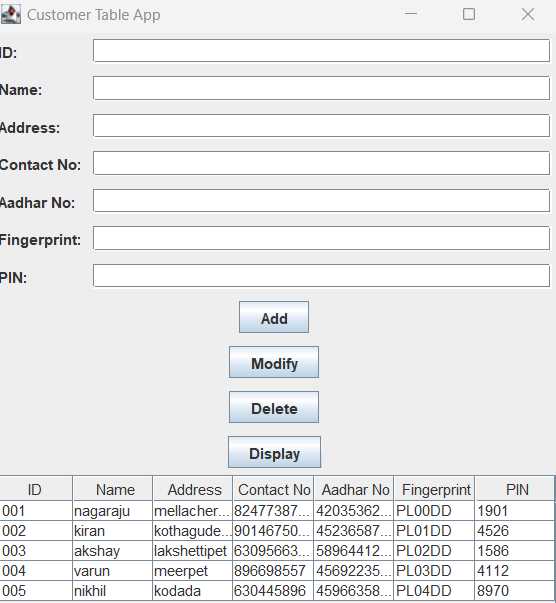
}

}

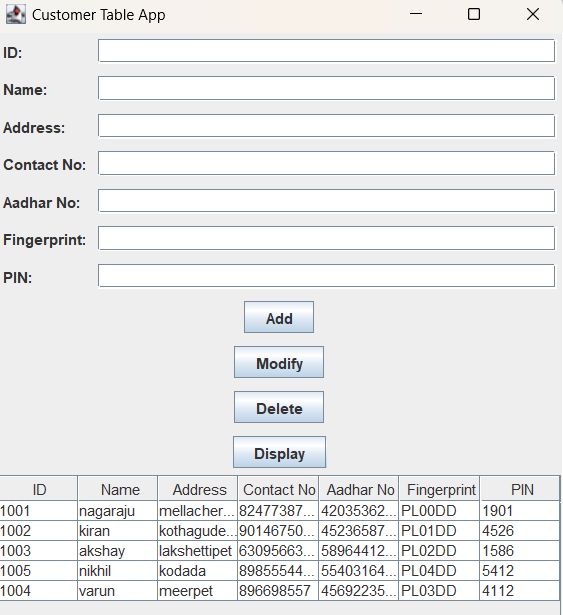
**TESTING**

**Customer page:**

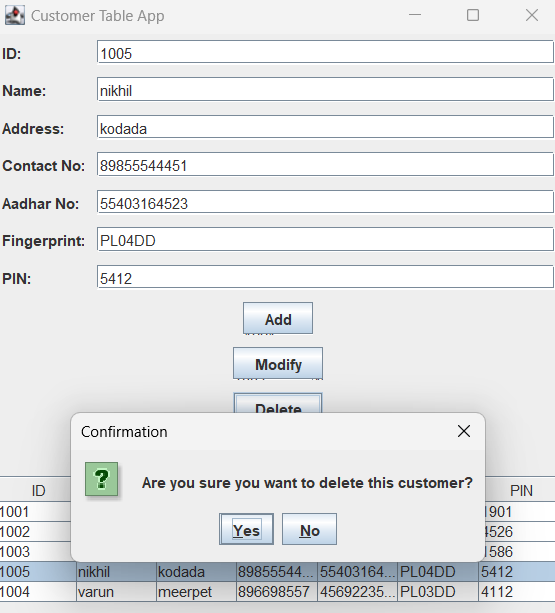
Before insertion



After modify**:**

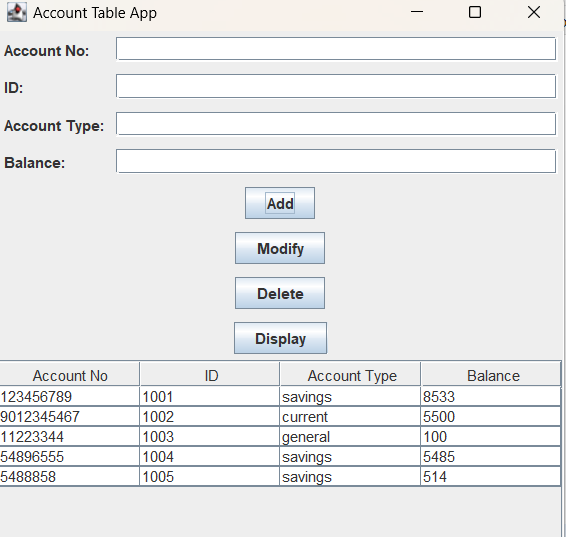
****

**Deletion:**

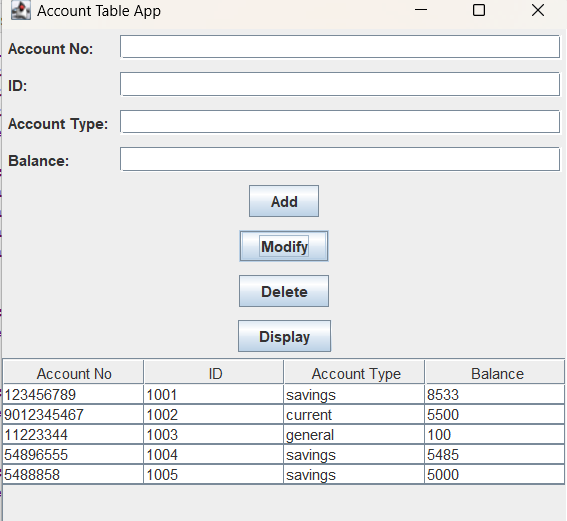
****

**Account Table:**

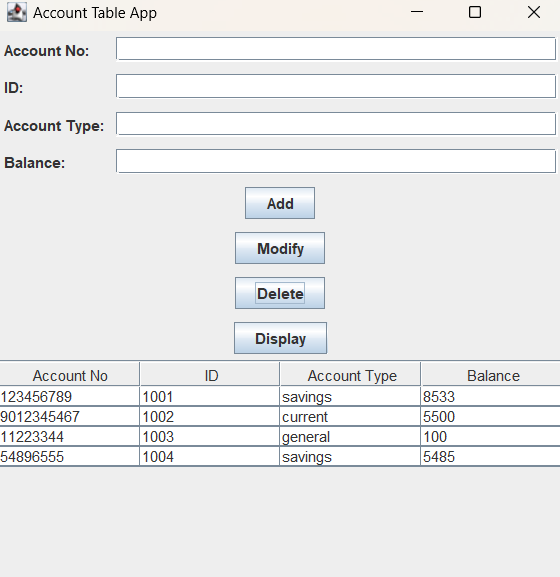
Before insertion:



Modify:

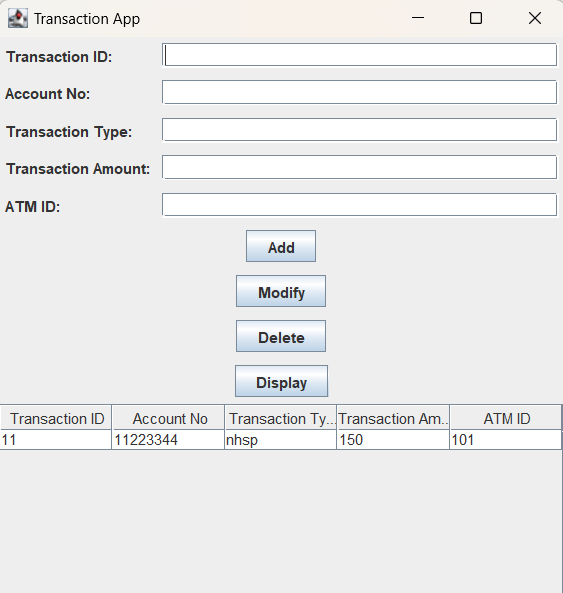


Deletion:

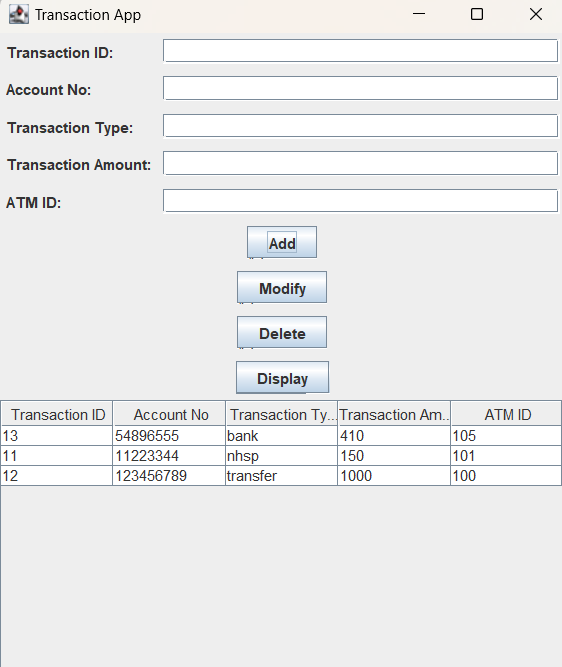


**Transaaction table:**

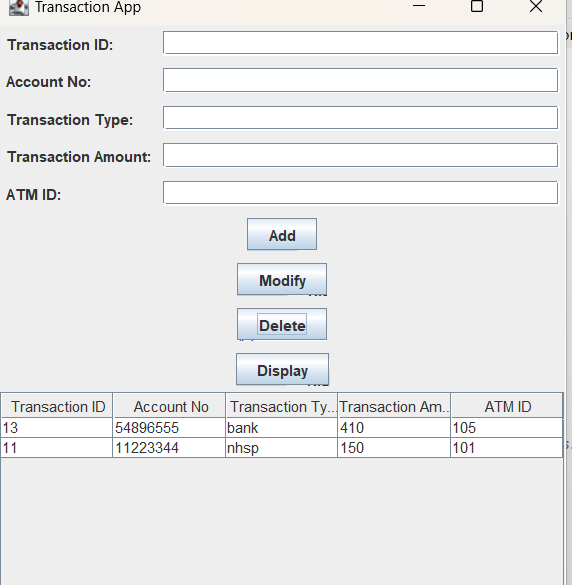
Before Insertion:



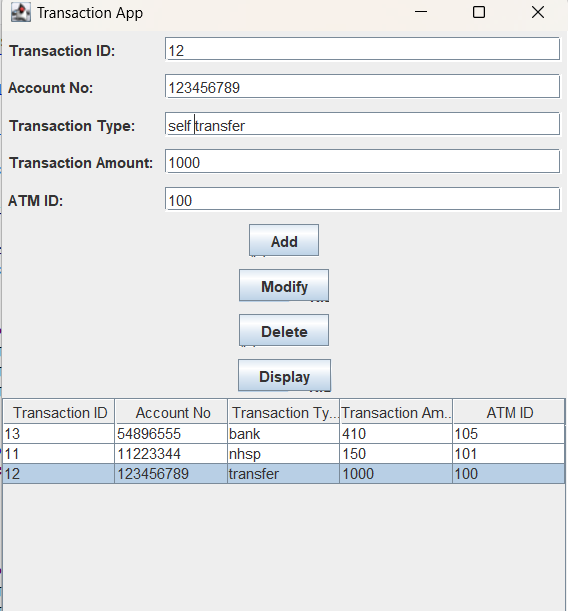
After Insertion:



Delete:

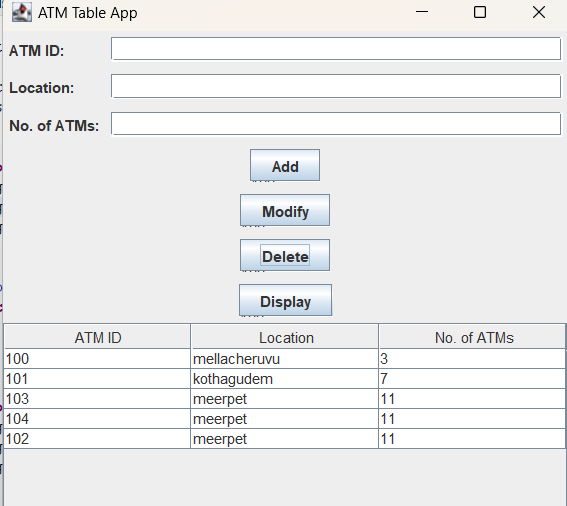


Modify:

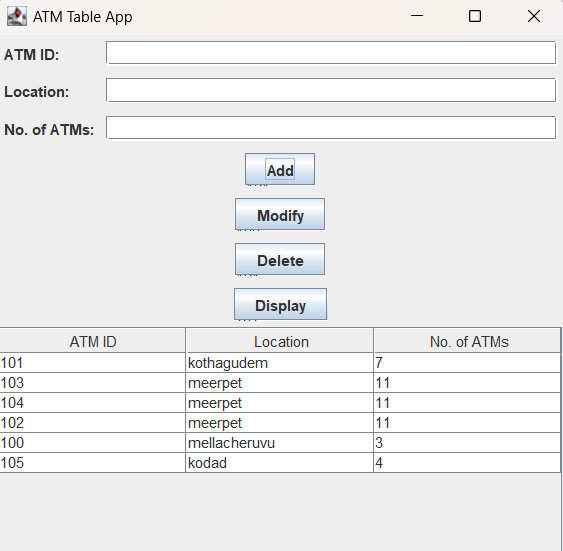


**Atm page:**

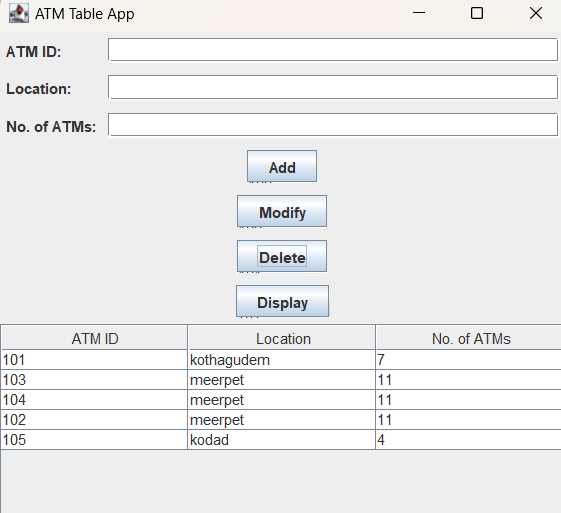
Before Insertion:



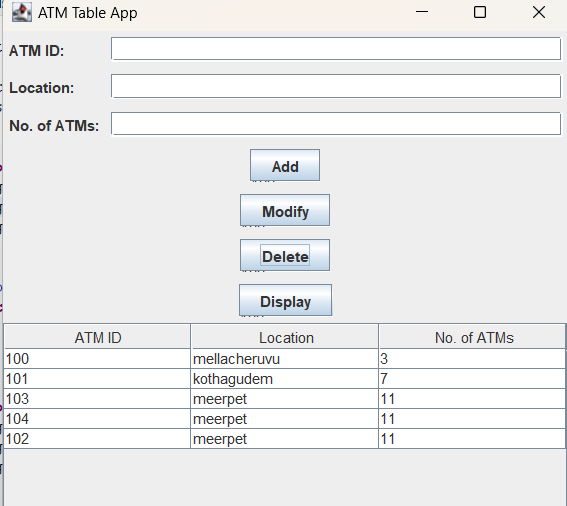
After insert:



Modify:

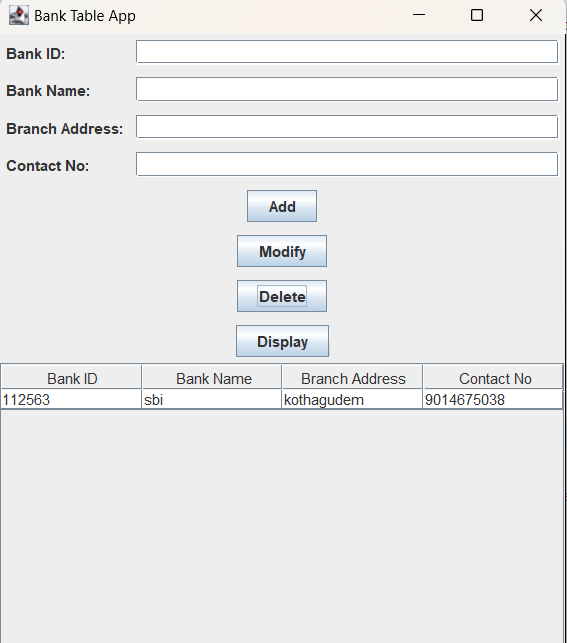


Delete:

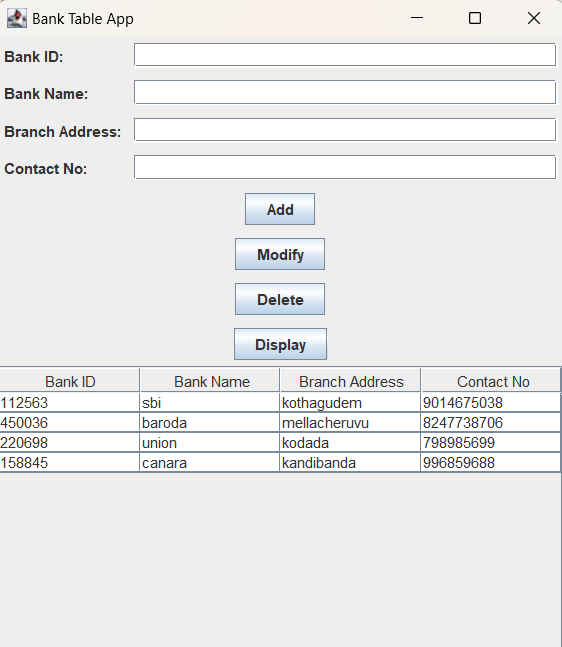


**Bank page:**

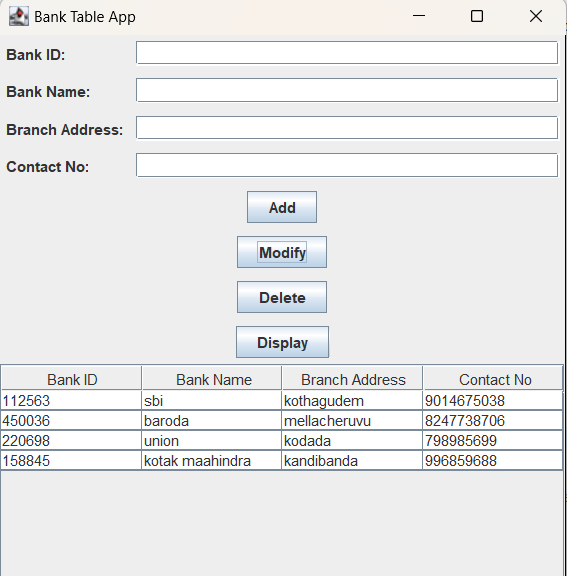
Before insertion



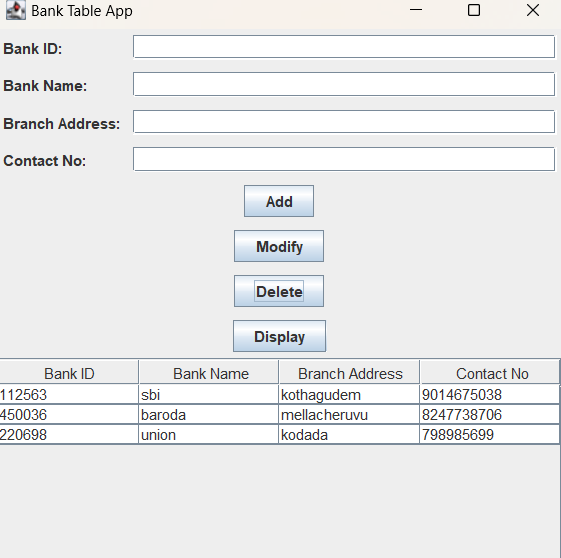
After insert:



Modify:



Delete:



**Result:**

I have successfully completed my DBMS project’finger print based ATM system’.

**DISCUSSION AND FUTURE WORK:**

**Benefits:**

a.Enhanced Security: Fingerprint-based authentication adds an additional layer of security compared to traditional PIN-based systems. Each person's fingerprint is unique, making it difficult for unauthorized individuals to access the account.

b.Convenience: Users don't need to carry physical cards or remember PINs, reducing the risk of card loss or PIN theft. The authentication process becomes quicker and more convenient as users only need their fingerprint to access the ATM.

c.Aadhaar Integration: Aadhaar card linking provides a reliable source of biometric data for authentication. Leveraging the existing Aadhaar infrastructure can streamline the integration process.

**Future Work:**

a.System Optimization: Future work can focus on refining the fingerprint recognition algorithms to enhance accuracy and speed. Improving the system's performance will ensure smoother user experiences and minimize false rejections or acceptances.

b.Scalability and Integration: Integrating fingerprint-based ATM systems with existing banking infrastructure and networks requires careful planning. Future work involves addressing scalability challenges to accommodate a large user base and seamless integration with different banks' systems.

c.Security Enhancements: While fingerprint-based authentication is more secure, it is not foolproof. Future work should continue to explore additional security measures, such as multi-factor authentication, to strengthen the system against potential vulnerabilities.

d.User Education and Acceptance: Introducing a new authentication method requires user education and awareness. Future work involves conducting awareness campaigns and providing clear instructions to users on how to use the fingerprint-based ATM system securely.

e.Regulatory Compliance: The implementation of fingerprint-based ATM systems needs to align with relevant regulatory guidelines and data protection laws. Future work should ensure compliance with privacy regulations and address concerns related to data storage and protection.

f.Accessibility Considerations: Future work should focus on designing the system with accessibility in mind. This includes accommodating users with disabilities and providing alternative authentication methods for individuals whose fingerprints may not be suitable for scanning.

In summary, a fingerprint-based ATM system using Aadhaar card linking offers improved security and convenience. Future work should concentrate on system optimization, scalability, security enhancements, user education, regulatory compliance, and accessibility considerations to further develop and refine such a system.

**References:**

● https://docs.oracle.com/javase/7/docs/api/

● <https://www.javatpoint.com/java-swing>

● <https://stackoverflow.com/>

**Git-Hub LINK:**

https://github.com/nagarajuhalavath/FINGERPRINTBASED-ATM-SYSTEM