JAVA SWING BASED –FOREX TRADING DATABASE MANAGEMENT– SQL CONNECTIVITY USING JDBC

A Report
Submitted in partial fulfillment of the Requirements
for the COURSE

DATABASE MANAGEMENT SYSTEMS

By

NAGASAI <1602-21-737-036>
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
'Forex Trading Database Management'
is a project work of **Nagasai** bearing roll no. 1602-21-737-036 who carried out this project under my supervision in the IV semesterfor
the academic year 2022- 2023

3

Signature
External Examiner

Signature
Internal Examiner

1602-21-737-036
Nagasai

ABSTRACT

The Forex trading market is a complex and highly dynamic environment where currencies are traded 24 hours a day across different time zones. As such, managing the vast amounts of data generated by Forex trading activities can be a challenging task for traders. A robust database management system can help ensure that trading data is properly organized, easily accessible, and securely stored.

This Forex trading database management project aims to design and implement a database management system that is tailored specifically to the needs of Forex traders and brokers. The system will be designed to handle large volumes of data, including real-time market data, trade orders, transaction history, and client information.

Overall, this Forex trading database management project aims to provide traders and brokers with a powerful tool for managing their trading activities. By providing a secure, scalable, and user-friendly database management system, the project aims to help traders and brokers optimize their trading strategies.

Requirement Analysis

List of Tables:

1. Currency
2. Client
3. Account
4. Trade
5. Transaction

List of Attributes with their Domain Types:

Currency

• currency_id	NOT NULL INT
• currency_name	NOT NULL VARCHAR2(20)
• exchange_rate	NOT NULL DECIMAL(10,4)

Client

• client_id	NOT NULL INT
• name	NOT NULL VARCHAR2(50)
• account_number	NOT NULL VARCHAR2(10)

Account

- account_number NOT NULL INT
- account_balance NOT NULL DECIMAL(10,2)
- client_id NOT NULL INT

Trade

- trade_id NOT NULL INT
- trade_type NOT NULL VARCHAR2(10)
- Currency_type NOT NULL VARCHAR(20)
- Trade_amount NOT NULL DECIMAL(10,4)
- time NOT NULL DATE
- Client_id NOT NULL INT

Transaction

- Transaction_id NOT NULL INT
- Transaction_type NOT NULL VARCHAR2(10)
- Currency_type NOT NULL VARCHAR(20)
- Transaction_amount NOT NULL DECIMAL(10,4)
- Transaction_date NOT NULL DATE
- Client_id NOT NULL INT

AIM AND PRIORITY OF THE PROJECT

To create a **Java GUI-based** desktop application for forex trading database management, the application aims to efficiently manage and organize forex trading data. It allows users to input and update information such as trade details, currency pairs, transaction history, etc., through user-friendly forms. The application utilizes JDBC connectivity to connect with the database, ensuring accurate and secure storage of trading data. It prioritizes data integrity, implements robust security measures, optimizes performance, and facilitates real-time updates.

ARCHITECTURE AND TECHNOLOGY

Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

Java SWING:

Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

DESIGN

Entity Relationship Diagram

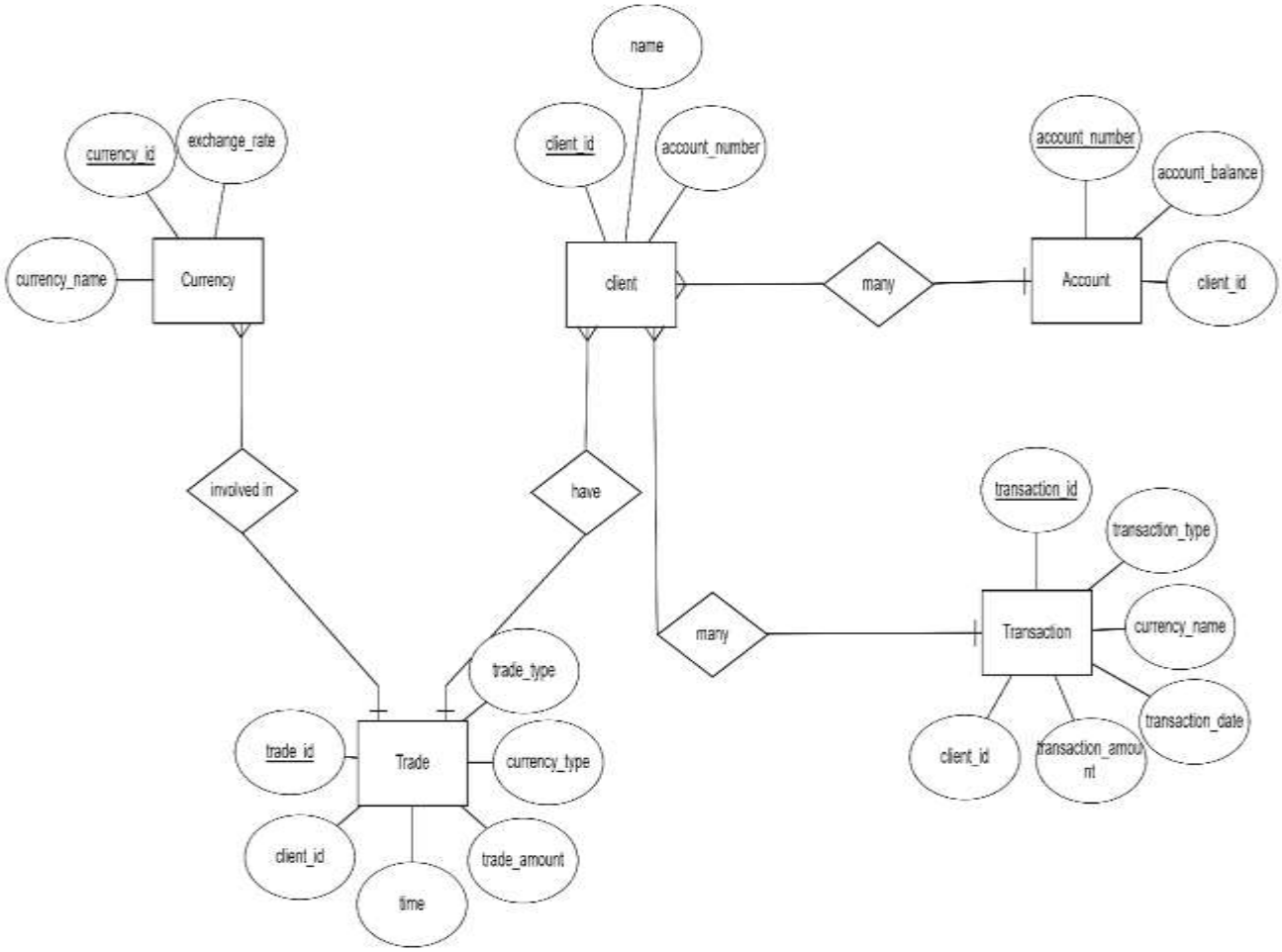


TABLE CREATED IN SQL:

1.Currency Table

```
SQL> CREATE TABLE Currency (
  2       currency_id INT PRIMARY KEY,
  3       currency_name VARCHAR(50),
  4       exchange_rate DECIMAL(10, 4) NOT NULL
  5  );

Table created.
```

2.Client Table

```
SQL> CREATE TABLE Client (
  2       client_id INT PRIMARY KEY,
  3       name VARCHAR(50) NOT NULL,
  4       account_number VARCHAR(10) NOT NULL
  5  );

Table created.
```

3.Account Table

```
SQL> CREATE TABLE Account (
  2     account_number VARCHAR(10) PRIMARY KEY,
  3     account_balance DECIMAL(18, 4) NOT NULL,
  4     client_id INT NOT NULL,
  5     FOREIGN KEY (client_id) REFERENCES Client(client_id)
  6 );

Table created.
```

4. Trade table

```
SQL> CREATE TABLE Trade (
  2     trade_id INT PRIMARY KEY,
  3     trade_type VARCHAR(10) NOT NULL,
  4     currency_type VARCHAR(7) NOT NULL,
  5     trade_amount DECIMAL(18, 4) NOT NULL,
  6     time DATE NOT NULL,
  7     client_id INT NOT NULL,
  8     FOREIGN KEY (client_id) REFERENCES Client(client_id)
  9 );

Table created.
```

5. Transaction table

```
SQL> CREATE TABLE Transaction (
  2     transaction_id INT PRIMARY KEY,
  3     transaction_type VARCHAR(10) NOT NULL,
  4     currency_name VARCHAR(50),
  5     transaction_amount DECIMAL(18, 4) NOT NULL,
  6     transaction_date DATE NOT NULL,
  7     client_id INT NOT NULL,
  8     FOREIGN KEY (client_id) REFERENCES Client(client_id)
  9 );

Table created.
```

Database Design

```
SQL> desc Client;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CLIENT_ID                                 NOT NULL NUMBER(38)
 NAME                                      NOT NULL VARCHAR2(50)
 ACCOUNT_NUMBER                            NOT NULL VARCHAR2(10)

SQL> desc Currency;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CURRENCY_ID                               NOT NULL NUMBER(38)
 CURRENCY_NAME                                      VARCHAR2(50)
 EXCHANGE_RATE                             NOT NULL NUMBER(10,4)

SQL> desc Account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NUMBER                            NOT NULL VARCHAR2(10)
 ACCOUNT_BALANCE                           NOT NULL NUMBER(18,4)
 CLIENT_ID                                 NOT NULL NUMBER(38)

SQL> desc Transaction;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRANSACTION_ID                            NOT NULL NUMBER(38)
 TRANSACTION_TYPE                          NOT NULL VARCHAR2(10)
 CURRENCY_NAME                                      VARCHAR2(50)
 TRANSACTION_AMOUNT                        NOT NULL NUMBER(18,4)
 TRANSACTION_DATE                          NOT NULL DATE
 CLIENT_ID                                 NOT NULL NUMBER(38)

SQL> desc Trade;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRADE_ID                                  NOT NULL NUMBER(38)
 TRADE_TYPE                                NOT NULL VARCHAR2(10)
 CURRENCY_TYPE                             NOT NULL VARCHAR2(7)
 TRADE_AMOUNT                              NOT NULL NUMBER(18,4)
 TIME                                      NOT NULL DATE
 CLIENT_ID                                 NOT NULL NUMBER(38)
```

DML Operations

1.INSERTING VALUES INTO CURRENCY TABLE:

```
SQL> INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate);
Enter value for currency_id: 1
Enter value for currency_name: USD
Enter value for exchange_rate: 1.000
old   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate)
new   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(1,'USD',1.000)

1 row created.

SQL> /
Enter value for currency_id: 2
Enter value for currency_name: INR
Enter value for exchange_rate: 0.0122
old   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate)
new   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(2,'INR',0.0122)

1 row created.

SQL> /
Enter value for currency_id: 3
Enter value for currency_name: GBP
Enter value for exchange_rate: 1.2492
old   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate)
new   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(3,'GBP',1.2492)

1 row created.

SQL> /
Enter value for currency_id: 4
Enter value for currency_name: JPY
Enter value for exchange_rate: 0.00728
old   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate)
new   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(4,'JPY',0.00728)

1 row created.

SQL> /
Enter value for currency_id: CHF
Enter value for currency_name: CHF
Enter value for exchange_rate: 1.1163
old   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate)
new   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(CHF,'CHF',1.1163)
INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(CHF,'CHF',1.1163)
                                                                     *
ERROR at line 1:
ORA-00984: column not allowed here


SQL> /
Enter value for currency_id: 5
Enter value for currency_name: CHF
Enter value for exchange_rate: 1.1163
old   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(&currency_id,'&currency_name',&exchange_rate)
new   1: INSERT INTO Currency(currency_id,currency_name,exchange_rate) VALUES(5,'CHF',1.1163)

1 row created.

SQL>
```

2. INSERTING VALUES INTO CLIENT TABLE:

```
SQL> INSERT INTO Client(client_id,name,account_number) VALUES(&client_id,'&name','&account_number');
Enter value for client_id: 1001
Enter value for name: Sai
Enter value for account_number: 123456
old   1: INSERT INTO Client(client_id,name,account_number) VALUES(&client_id,'&name','&account_number')
new   1: INSERT INTO Client(client_id,name,account_number) VALUES(1001,'Sai','123456')

1 row created.

SQL> /
Enter value for client_id: 1002
Enter value for name: Kohli
Enter value for account_number: 181718
old   1: INSERT INTO Client(client_id,name,account_number) VALUES(&client_id,'&name','&account_number')
new   1: INSERT INTO Client(client_id,name,account_number) VALUES(1002,'Kohli','181718')

1 row created.

SQL> /
Enter value for client_id: 1003
Enter value for name: John
Enter value for account_number: 112233
old   1: INSERT INTO Client(client_id,name,account_number) VALUES(&client_id,'&name','&account_number')
new   1: INSERT INTO Client(client_id,name,account_number) VALUES(1003,'John','112233')

1 row created.

SQL> /
Enter value for client_id: 1004
Enter value for name: Lee
Enter value for account_number: 123321
old   1: INSERT INTO Client(client_id,name,account_number) VALUES(&client_id,'&name','&account_number')
new   1: INSERT INTO Client(client_id,name,account_number) VALUES(1004,'Lee','123321')

1 row created.

SQL>
```

3. INSERTING VALUES INTO ACCOUNT TABLE:

```
SQL> INSERT INTO Account(account_number,account_balance,client_id) values(&account_number,&account_balance,&client_id);
Enter value for account_number: 123456
Enter value for account_balance: 10000.00
Enter value for client_id: 1001
old   1: INSERT INTO Account(account_number,account_balance,client_id) values(&account_number,&account_balance,&client_id)
new   1: INSERT INTO Account(account_number,account_balance,client_id) values(123456,10000.00,1001)

1 row created.

SQL> /
Enter value for account_number: 147852
Enter value for account_balance: 15000.00
Enter value for client_id: 1045
old   1: INSERT INTO Account(account_number,account_balance,client_id) values(&account_number,&account_balance,&client_id)
new   1: INSERT INTO Account(account_number,account_balance,client_id) values(147852,15000.00,1045)
INSERT INTO Account(account_number,account_balance,client_id) values(147852,15000.00,1045)
*
ERROR at line 1:
ORA-02291: integrity constraint (NAGASAI.SYS_C007002) violated - parent key not
found


SQL> /
Enter value for account_number: 181718
Enter value for account_balance: 25000.00
Enter value for client_id: 1002
old   1: INSERT INTO Account(account_number,account_balance,client_id) values(&account_number,&account_balance,&client_id)
new   1: INSERT INTO Account(account_number,account_balance,client_id) values(181718,25000.00,1002)

1 row created.

SQL> /
Enter value for account_number: 123321
Enter value for account_balance: 15000.00
Enter value for client_id: 1004
old   1: INSERT INTO Account(account_number,account_balance,client_id) values(&account_number,&account_balance,&client_id)
new   1: INSERT INTO Account(account_number,account_balance,client_id) values(123321,15000.00,1004)

1 row created.

SQL>
```

4. INSERTING VALUES INTO TRADE TABLE:

```
SQL> INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(&trade_id,'&trade_type','&currency_type',&trade_amount,'&time',&client_id);
Enter value for trade_id: 1
Enter value for trade_type: Buy
Enter value for currency_type: USD
Enter value for trade_amount: 4500.00
Enter value for time: 16-Apr-1004
Enter value for client_id: 1001
old   1: INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(&trade_id,'&trade_type','&currency_type',&trade_amount,'&time',&client_id)

new   1: INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(1,'Buy','USD',4500.00,'16-Apr-1004',1001)

1 row created.

SQL> /
Enter value for trade_id: 2
Enter value for trade_type: Sell
Enter value for currency_type: EUR
Enter value for trade_amount: 1000.00
Enter value for time: 28-Oct-1022
Enter value for client_id: 1002
old   1: INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(&trade_id,'&trade_type','&currency_type',&trade_amount,'&time',&client_id)

new   1: INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(2,'Sell','EUR',1000.00,'28-Oct-1022',1002)

1 row created.

SQL> /
Enter value for trade_id: 3
Enter value for trade_type: JPY
Enter value for currency_type: JPY
Enter value for trade_amount: 7000.00
Enter value for time: 01-05-2023
Enter value for client_id: 1002
old   1: INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(&trade_id,'&trade_type','&currency_type',&trade_amount,'&time',&client_id)

new   1: INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(3,'JPY','JPY',7000.00,'01-05-2023',1002)
INSERT INTO Trade(trade_id,trade_type,currency_type,trade_amount,time,client_id) VALUES(3,'JPY','JPY',7000.00,'01-05-2023',1002)
                                                                                                                          *
ERROR at line 1:
ORA-01843: not a valid month


SQL> /
Enter value for trade_id: 3
Enter value for trade_type: Buy
Enter value for currency_type: JPY
Enter value for trade_amount: 7000.00
Enter value for time: 25-Apr-2023
Enter value for client_id: 1002
```

5. INSERTING VALUES INTO TRANSACTION TABLE:

```
SQL> INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(&transaction_id,'&transaction_type','&curre
ncy_name',&transaction_amount,'&transaction_date',&client_id);
Enter value for transaction_id: 1
Enter value for transaction_type: Deposit
Enter value for currency_name: USD
Enter value for transaction_amount: 7500.00
Enter value for transaction_date: 10-Apr-2023
Enter value for client_id: 1001
old   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(&transaction_id,'&transaction_type','&c
urrency_name',&transaction_amount,'&transaction_date',&client_id)
new   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(1,'Deposit','USD',7500.00,'10-Apr-2023'
,1001)

1 row created.

SQL> /
Enter value for transaction_id: 2
Enter value for transaction_type: withdrawal
Enter value for currency_name: INR
Enter value for transaction_amount: 1500.00
Enter value for transaction_date: 3-Apr-2023
Enter value for client_id: 1002
old   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(&transaction_id,'&transaction_type','&c
urrency_name',&transaction_amount,'&transaction_date',&client_id)
new   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(2,'withdrawal','INR',1500.00,'3-Apr-2023
',1002)

1 row created.

SQL> /
Enter value for transaction_id: 3
Enter value for transaction_type: Deposit
Enter value for currency_name: EUR
Enter value for transaction_amount: 1000.00
Enter value for transaction_date: 10-Apr-2023
Enter value for client_id:
old   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(&transaction_id,'&transaction_type','&c
urrency_name',&transaction_amount,'&transaction_date',&client_id)
new   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(3,'Deposit','EUR',1000.00,'10-Apr-2023'
,)
INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(3,'Deposit','EUR',1000.00,'10-Apr-2023',)
                                                                                                                                                            *
ERROR at line 1:
ORA-00936: missing expression


SQL> /
Enter value for transaction_id: 3
Enter value for transaction_type: Deposit
Enter value for currency_name: EUR
Enter value for transaction_amount: 1000.00
Enter value for transaction_date: 10-Apr-2023
Enter value for client_id: 1001
old   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(&transaction_id,'&transaction_type','&c
urrency_name',&transaction_amount,'&transaction_date',&client_id)
new   1: INSERT INTO Transaction(transaction_id,transaction_type,currency_name,transaction_amount,transaction_date,client_id) VALUES(3,'Deposit','EUR',1000.00,'10-Apr-2023'
,1001)

1 row created.
```

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
{

        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

         // Connect to Oracle Database

         Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE"
,"nagasai","nagasai");

         Statement statement = con.createStatement()

         String query = "UPDATE SKILLS SET SS1=" +"'"+ jTextField3.getText() +"',SS2=" +"'"+
jTextField5.getText() +"',  AOI ="+" '"+ jTextField2.getText() +"' WHERE SID =+" + jTextField4.getText();

         ResultSet rs = statement.executeQuery(query);

         JOptionPane.showMessageDialog(new JFrame(), "Upadated Successfully", "INFORMATION",
JOptionPane.INFORMATION_MESSAGE);

         rs.close();

         statement.close();

         con.close(); }
```

Front-end Programs (User Interfaces) Home Page:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.net.URL;
import java.sql.*;

public class Interface extends JFrame {
    private static final String DB_URL =
  "jdbc:oracle:thin:@localhost:1521:xe";
    private static final String DB_USERNAME =
  "nagasai";
    private static final String DB_PASSWORD =
  "nagasai";

    private JButton adminButton;

    public Interface() {
        super("Java Application");

  setDefaultCloseOperation(JFrame.EXIT_ON_
  CLOSE);
        setSize(400, 300);

        initializeGUI();
    }

    private void initializeGUI() {
        // Create a panel with a background
  image
        JPanel panel = new JPanel() {
            @Override
            protected void
  paintComponent(Graphics g) {
                super.paintComponent(g);
                try {
```

```
// Load the background image from a URL
                URL imageURL = new URL("https://images.moneycontrol.com/static-
mcnews/2021/05/wallstreet-bull_04022021-770x433.jpg?impolicy=website&width=770&height=431");
                Image image = new ImageIcon(imageURL).getImage();
                g.drawImage(image, 0, 0, getWidth(), getHeight(), this);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    };

    // Set panel layout
    panel.setLayout(new FlowLayout());

    // Create the buttons
    JButton button1 = new JButton("Button 1");
    JButton button2 = new JButton("Button 2");

    // Rename Button 1 to Admin
    adminButton = button1;
    adminButton.setText("Admin");

    // Add action listeners to the buttons
    adminButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            openAdminInterface();
        }
    });

    button2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            openUserInterface();
        }
    });

    // Add the buttons to the panel
    panel.add(adminButton);
    panel.add(button2);

    // Add the panel to the frame
    getContentPane().add(panel, BorderLayout.CENTER);
  }
```

```
private void openUserInterface() {
            JFrame adminFrame = new JFrame("Admin Interface");
   adminFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
   adminFrame.setSize(400, 300);
}

private void openAdminInterface() {
   JFrame adminFrame = new JFrame("Admin Interface");
   adminFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
   adminFrame.setSize(400, 300);

   // Create the menu bar
   JMenuBar menuBar = new JMenuBar();

   // Create the "Clients" menu
   JMenu clientsMenu = new JMenu("Clients");

   // Create the menu items for the "Clients" menu
   JMenuItem viewClientsItem = new JMenuItem("View Clients");
   JMenuItem addClientsItem = new JMenuItem("Add Clients");
   JMenuItem updateClientsItem = new JMenuItem("Update Clients");
   JMenuItem deleteClientsItem = new JMenuItem("Delete Clients");

   // Add action listeners to the menu items
   viewClientsItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
         displayClientTable();
      }
   });

   addClientsItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
         addClient();
      }
   });

   updateClientsItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
         updateClient();
      }
   });
```

```
deleteClientsItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            deleteClient();
        }
    });

    // Add the menu items to the "Clients" menu
    clientsMenu.add(viewClientsItem);
    clientsMenu.add(addClientsItem);
    clientsMenu.add(updateClientsItem);
    clientsMenu.add(deleteClientsItem);

    // Create the "Transactions" menu
    JMenu transactionsMenu = new JMenu("Transactions");

    // Create the menu items for the "Transactions" menu
    JMenuItem viewTransactionsItem = new JMenuItem("View Transactions");
    JMenuItem addTransactionsItem = new JMenuItem("Add Transactions");
    JMenuItem updateTransactionsItem = new JMenuItem("Update Transactions");
    JMenuItem deleteTransactionsItem = new JMenuItem("Delete Transactions");

    // Add action listeners to the menu items
    viewTransactionsItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayTransactionTable();
        }
    });

    addTransactionsItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            addTransaction();
        }
    });

    updateTransactionsItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            updateTransaction();
        }
    });

    deleteTransactionsItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            deleteTransaction();
        }
    });
```

```
// Add the menu items to the "Transactions" menu
    transactionsMenu.add(viewTransactionsItem);
    transactionsMenu.add(addTransactionsItem);
    transactionsMenu.add(updateTransactionsItem);
    transactionsMenu.add(deleteTransactionsItem);

    // Create the "Trades" menu
    JMenu tradesMenu = new JMenu("Trades");

    // Create the menu items for the "Trades" menu
    JMenuItem viewTradesItem = new JMenuItem("View Trades");
    JMenuItem addTradesItem = new JMenuItem("Add Trades");
    JMenuItem updateTradesItem = new JMenuItem("Update Trades");
    JMenuItem deleteTradesItem = new JMenuItem("Delete Trades");

    // Add action listeners to the menu items
    viewTradesItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        displayTradeTable();
      }
    });

    addTradesItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        addTrade();
      }
    });

    updateTradesItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        updateTrade();
      }
    });

    deleteTradesItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        deleteTrade();
      }
    });

    // Add the menu items to the "Trades" menu
    tradesMenu.add(viewTradesItem);
    tradesMenu.add(addTradesItem);
    tradesMenu.add(updateTradesItem);
    tradesMenu.add(deleteTradesItem);
```

```
// Create the "Currency" menu
    JMenu currencyMenu = new JMenu("Currency");

    // Create the menu items for the "Currency" menu
    JMenuItem viewCurrencyItem = new JMenuItem("View Currency");
    JMenuItem addCurrencyItem = new JMenuItem("Add Currency");
    JMenuItem updateCurrencyItem = new JMenuItem("Update Currency");
    JMenuItem deleteCurrencyItem = new JMenuItem("Delete Currency");

    // Add action listeners to the menu items
    viewCurrencyItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        displayCurrencyTable();
      }
    });

    addCurrencyItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        addCurrency();
      }
    });

    updateCurrencyItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        updateCurrency();
      }
    });

    deleteCurrencyItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        deleteCurrency();
      }
    });
    currencyMenu.add(viewCurrencyItem);
    currencyMenu.add(addCurrencyItem);
    currencyMenu.add(updateCurrencyItem);
    currencyMenu.add(deleteCurrencyItem);

    // Create the "Accounts" menu
    JMenu accountsMenu = new JMenu("Accounts");

    // Create the menu items for the "Accounts" menu
    JMenuItem viewAccountsItem = new JMenuItem("View Accounts");
    JMenuItem addAccountsItem = new JMenuItem("Add Account");
    JMenuItem updateAccountsItem = new JMenuItem("Update Account");
    JMenuItem deleteAccountsItem = new JMenuItem("Delete Account");
```

```
    // Add action listeners to the menu items
        viewAccountsItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayAccountTable();
            }
        });

        addAccountsItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                addAccount();
            }
        });

        updateAccountsItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                updateAccount();
            }
        });

        deleteAccountsItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                deleteAccount();
            }
        });

        // Add the menu items to the "Accounts" menu
        accountsMenu.add(viewAccountsItem);
        accountsMenu.add(addAccountsItem);
        accountsMenu.add(updateAccountsItem);
        accountsMenu.add(deleteAccountsItem);

        // Add the menus to the menu bar
        menuBar.add(clientsMenu);
        menuBar.add(transactionsMenu);
        menuBar.add(tradesMenu);
        menuBar.add(currencyMenu);
        menuBar.add(accountsMenu);

        // Set the menu bar on the admin frame
        adminFrame.setJMenuBar(menuBar);

        adminFrame.setVisible(true);
    }
```

```
    private void displayCurrencyTable() {
        JFrame currencyFrame = new JFrame("Currency Table");
        currencyFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        currencyFrame.setSize(400, 300);

        // Create a JTextArea to display the Currency table rows
        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setPreferredSize(new Dimension(380, 250)); // Adjust the preferred size as needed

        // Establish a database connection
        try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
    DB_PASSWORD)) {
            // Create a statement
            Statement statement = connection.createStatement();

            // Execute the SELECT query on the Currency table
            ResultSet resultSet = statement.executeQuery("SELECT * FROM Currency");

            // Iterate through the result set and append the rows to the text area
            while (resultSet.next()) {
                int currencyId = resultSet.getInt("CURRENCY_ID");
                String currencyName = resultSet.getString("CURRENCY_NAME");
                double exchangeRate = resultSet.getDouble("EXCHANGE_RATE");

                textArea.append("CURRENCY_ID: " + currencyId + "\n");
                textArea.append("CURRENCY_NAME: " + currencyName + "\n");
                textArea.append("EXCHANGE_RATE: " + exchangeRate + "\n\n");
            }
            resultSet.close();
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        currencyFrame.getContentPane().add(new JScrollPane(textArea));

        currencyFrame.setVisible(true);
    }

    private void addCurrency() {
        JFrame addCurrencyFrame = new JFrame("Add Currency");
        addCurrencyFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        addCurrencyFrame.setSize(400, 300);
```

```
 // Create labels and text fields for currency information
      JLabel currencyIdLabel = new JLabel("Currency ID:");
      JTextField currencyIdField = new JTextField(10);
      JLabel currencyNameLabel = new JLabel("Currency Name:");
      JTextField currencyNameField = new JTextField(10);
      JLabel exchangeRateLabel = new JLabel("Exchange Rate:");
      JTextField exchangeRateField = new JTextField(10);

      // Create a button to add the currency
      JButton addButton = new JButton("Add");

      // Set the layout for the add currency panel
      JPanel addCurrencyPanel = new JPanel(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns with 10px
horizontal and vertical gaps

      // Add components to the add currency panel
      addCurrencyPanel.add(currencyIdLabel);
      addCurrencyPanel.add(currencyIdField);
      addCurrencyPanel.add(currencyNameLabel);
      addCurrencyPanel.add(currencyNameField);
      addCurrencyPanel.add(exchangeRateLabel);
      addCurrencyPanel.add(exchangeRateField);
      addCurrencyPanel.add(addButton);

      // Add an action listener to the add button
      addButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            // Get the input values
            int currencyId = Integer.parseInt(currencyIdField.getText());
            String currencyName = currencyNameField.getText();
            double exchangeRate = Double.parseDouble(exchangeRateField.getText());

            // Insert the currency into the Currency table
            try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD)) {
               // Create a prepared statement
               PreparedStatement statement = connection.prepareStatement("INSERT INTO Currency
(CURRENCY_ID, CURRENCY_NAME, EXCHANGE_RATE) VALUES (?, ?, ?)");
               statement.setInt(1, currencyId);
               statement.setString(2, currencyName);
               statement.setDouble(3, exchangeRate);
               int rowsAffected = statement.executeUpdate();
               JOptionPane.showMessageDialog(addCurrencyFrame, "Currency added successfully. Rows
affected: " + rowsAffected);
```

```
// Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the add currency panel to the add currency frame
    addCurrencyFrame.getContentPane().add(addCurrencyPanel);

    addCurrencyFrame.setVisible(true);
  }

  private void updateCurrency() {
    JFrame updateCurrencyFrame = new JFrame("Update Currency");
    updateCurrencyFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    updateCurrencyFrame.setSize(400, 300);

    // Create labels and text fields for update information
    JLabel currencyIdLabel = new JLabel("Currency ID to update:");
    JTextField currencyIdField = new JTextField(10);
    JLabel exchangeRateLabel = new JLabel("Updated Exchange Rate:");
    JTextField exchangeRateField = new JTextField(10);

    // Create a button to update the currency
    JButton updateButton = new JButton("Update");

    // Set the layout for the update currency panel
    JPanel updateCurrencyPanel = new JPanel(new GridLayout(3, 2, 10, 10)); // 3 rows, 2 columns with
10px horizontal and vertical gaps

    // Add components to the update currency panel
    updateCurrencyPanel.add(currencyIdLabel);
    updateCurrencyPanel.add(currencyIdField);
    updateCurrencyPanel.add(exchangeRateLabel);
    updateCurrencyPanel.add(exchangeRateField);
    updateCurrencyPanel.add(updateButton);

    // Add an action listener to the update button
    updateButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Get the input values
            int currencyId = Integer.parseInt(currencyIdField.getText());
            double updatedExchangeRate = Double.parseDouble(exchangeRateField.getText());
```

```
 // Update the exchange rate in the Currency table
            try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD)) {
                // Create a prepared statement
                PreparedStatement statement = connection.prepareStatement("UPDATE Currency SET
EXCHANGE_RATE = ? WHERE CURRENCY_ID = ?");
                statement.setDouble(1, updatedExchangeRate);
                statement.setInt(2, currencyId);
int rowsAffected = statement.executeUpdate();

JOptionPane.showMessageDialog(updateCurrencyFrame, "Currency updated successfully. Rows affected:
" + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the update currency panel to the update currency frame
    updateCurrencyFrame.getContentPane().add(updateCurrencyPanel);
    updateCurrencyFrame.setVisible(true);
  }

  private void deleteCurrency() {
    JFrame deleteCurrencyFrame = new JFrame("Delete Currency");
    deleteCurrencyFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    deleteCurrencyFrame.setSize(400, 300);

    // Create labels and text fields for delete information
    JLabel currencyIdLabel = new JLabel("Currency ID to delete:");
    JTextField currencyIdField = new JTextField(10);

    // Create a button to delete the currency
    JButton deleteButton = new JButton("Delete");

    // Set the layout for the delete currency panel
    JPanel deleteCurrencyPanel = new JPanel(new GridLayout(2, 2, 10, 10)); // 2 rows, 2 columns with
10px horizontal and vertical gaps

    // Add components to the delete currency panel
    deleteCurrencyPanel.add(currencyIdLabel);
    deleteCurrencyPanel.add(currencyIdField);
    deleteCurrencyPanel.add(deleteButton);
```

```
 // Add an action listener to the delete button
      deleteButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            // Get the input value
            int currencyId = Integer.parseInt(currencyIdField.getText());

            // Delete the currency from the Currency table
            try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD)) {
               // Create a prepared statement
               PreparedStatement statement = connection.prepareStatement("DELETE FROM Currency
WHERE CURRENCY_ID = ?");
               statement.setInt(1, currencyId);

               // Execute the statement
               int rowsAffected = statement.executeUpdate();

               // Display a message indicating the success of the deletion
               JOptionPane.showMessageDialog(deleteCurrencyFrame, "Currency deleted successfully. Rows
affected: " + rowsAffected);

               // Close the statement
               statement.close();
            } catch (SQLException ex) {
               ex.printStackTrace();
            }
         }
      });

      // Add the delete currency panel to the delete currency frame
      deleteCurrencyFrame.getContentPane().add(deleteCurrencyPanel);

      deleteCurrencyFrame.setVisible(true);
   }


   private void displayClientTable() {
      JFrame clientFrame = new JFrame("Client Table");
      clientFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      clientFrame.setSize(400, 300);

      // Create a JTextArea to display the Client table rows
      JTextArea textArea = new JTextArea();
      textArea.setEditable(false);
      textArea.setLineWrap(true);
      textArea.setWrapStyleWord(true);
      textArea.setPreferredSize(new Dimension(380, 250)); // Adjust the preferred size as needed
```

```
// Establish a database connection
    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD)) {
        // Create a statement
        Statement statement = connection.createStatement();

        // Execute the SELECT query on the Client table
        ResultSet resultSet = statement.executeQuery("SELECT * FROM Client");

        // Iterate through the result set and append the rows to the text area
        while (resultSet.next()) {
            int clientId = resultSet.getInt("CLIENT_ID");
            String name = resultSet.getString("NAME");
            String accountNumber = resultSet.getString("ACCOUNT_NUMBER");

            textArea.append("CLIENT_ID: " + clientId + "\n");
            textArea.append("NAME: " + name + "\n");
            textArea.append("ACCOUNT_NUMBER: " + accountNumber + "\n\n");
        }

        // Close the result set and statement
        resultSet.close();
        statement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    // Add the text area to the clientFrame
    clientFrame.getContentPane().add(new JScrollPane(textArea));

    clientFrame.setVisible(true);
}

private void addClient() {
    JFrame addClientFrame = new JFrame("Add Client");
    addClientFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    addClientFrame.setSize(400, 300);

    // Create labels and text fields for client information
    JLabel clientIdLabel = new JLabel("Client ID:");
    JTextField clientIdField = new JTextField(10);
    JLabel clientNameLabel = new JLabel("Client Name:");
    JTextField clientNameField = new JTextField(10);
    JLabel accountNumberLabel = new JLabel("Account Number:");
    JTextField accountNumberField = new JTextField(10);
```

```
// Create a button to add the client
    JButton addButton = new JButton("Add");

    // Set the layout for the add client panel
    JPanel addClientPanel = new JPanel(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns with 10px
horizontal and vertical gaps

    // Add components to the add client panel
    addClientPanel.add(clientIdLabel);
    addClientPanel.add(clientIdField);
    addClientPanel.add(clientNameLabel);
    addClientPanel.add(clientNameField);
    addClientPanel.add(accountNumberLabel);
    addClientPanel.add(accountNumberField);
//      addClientPanel.add(emailLabel);
//      addClientPanel.add(emailField);
    addClientPanel.add(addButton);

    // Add an action listener to the add button
    addButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Get the input values
            int clientId = Integer.parseInt(clientIdField.getText());
            String clientName = clientNameField.getText();
            String accountNumber = accountNumberField.getText();
//              String email = emailField.getText();

            // Insert the client into the Client table
            try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD)) {
                // Create a prepared statement
                PreparedStatement statement = connection.prepareStatement("INSERT INTO Client
(CLIENT_ID, CLIENT_NAME, EMAIL) VALUES (?, ?, ?)");
                statement.setInt(1, clientId);
                statement.setString(2, clientName);
                statement.setString(3, accountNumber);
//                  statement.setString(3, email);

                // Execute the statement
                int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the insertion
                JOptionPane.showMessageDialog(addClientFrame, "Client added successfully. Rows affected:
" + rowsAffected);

                // Close the statement
                statement.close();[
```

```
catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the add client panel to the add client frame
    addClientFrame.getContentPane().add(addClientPanel);

    addClientFrame.setVisible(true);
  }

  private void updateClient() {
    JFrame updateClientFrame = new JFrame("Update Client");
    updateClientFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    updateClientFrame.setSize(400, 300);

    // Create labels and text fields for update information
    JLabel clientIdLabel = new JLabel("Client ID to update:");
    JTextField clientIdField = new JTextField(10);
    JLabel emailLabel = new JLabel("Updated Email:");
    JTextField emailField = new JTextField(10);

    // Create a button to update the client
    JButton updateButton = new JButton("Update");

    // Set the layout for the update client panel
    JPanel updateClientPanel = new JPanel(new GridLayout(3, 2, 10, 10)); // 3 rows, 2
columns with 10px horizontal and vertical gaps

    // Add components to the update client panel
    updateClientPanel.add(clientIdLabel);
    updateClientPanel.add(clientIdField);
    updateClientPanel.add(emailLabel);
    updateClientPanel.add(emailField);
    updateClientPanel.add(updateButton);

    // Add an action listener to the update button
```

```
int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the update
                JOptionPane.showMessageDialog(updateClientFrame, "Client updated
successfully. Rows affected: " + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the update client panel to the update client frame
    updateClientFrame.getContentPane().add(updateClientPanel);

    updateClientFrame.setVisible(true);
}

private void deleteClient() {
    JFrame deleteClientFrame = new JFrame("Delete Client");
    deleteClientFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    deleteClientFrame.setSize(400, 300);

    // Create labels and text fields for delete information
    JLabel clientIdLabel = new JLabel("Client ID to delete:");
    JTextField clientIdField = new JTextField(10);

    // Create a button to delete the client
    JButton deleteButton = new JButton("Delete");

    // Set the layout for the delete client panel
    JPanel deleteClientPanel = new JPanel(new GridLayout(2, 2, 10, 10)); // 2 rows, 2
columns with 10px horizontal and vertical gaps

    deleteClientPanel.add(clientIdLabel);
    deleteClientPanel.add(clientIdField);
    deleteClientPanel.add(deleteButton);
```

```
deleteButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Get the input value
            int clientId = Integer.parseInt(clientIdField.getText());

            // Delete the client from the Client table
            try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                // Create a prepared statement
                PreparedStatement statement = connection.prepareStatement("DELETE FROM
Client WHERE CLIENT_ID = ?");
                statement.setInt(1, clientId);

                // Execute the statement
                int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the deletion
                JOptionPane.showMessageDialog(deleteClientFrame, "Client deleted
successfully. Rows affected: " + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the delete client panel to the delete client frame
    deleteClientFrame.getContentPane().add(deleteClientPanel);

    deleteClientFrame.setVisible(true);
}

private void displayTransactionTable() {
    JFrame transactionFrame = new JFrame("Transaction Table");
    transactionFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    transactionFrame.setSize(400, 300);
```

```
JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setPreferredSize(new Dimension(380, 250)); // Adjust the preferred size as
needed

        // Establish a database connection
        try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
            // Create a statement
            Statement statement = connection.createStatement();

            // Execute the SELECT query on the Transaction table
            ResultSet resultSet = statement.executeQuery("SELECT * FROM Transaction");

            // Iterate through the result set and append the rows to the text area
            while (resultSet.next()) {
                int transactionId = resultSet.getInt("TRANSACTION_ID");
                String transactionType = resultSet.getString("TRANSACTION_TYPE");
                String currencyName = resultSet.getString("CURRENCY_NAME");
                double transactionAmount = resultSet.getDouble("TRANSACTION_AMOUNT");
                Date transactionDate = resultSet.getDate("TRANSACTION_DATE");
                int clientId = resultSet.getInt("CLIENT_ID");

                textArea.append("TRANSACTION_ID: " + transactionId + "\n");
                textArea.append("TRANSACTION_TYPE: " + transactionType + "\n");
                textArea.append("CURRENCY_NAME: " + currencyName + "\n");
                textArea.append("TRANSACTION_AMOUNT: " + transactionAmount + "\n");
                textArea.append("TRANSACTION_DATE: " + transactionDate + "\n");
                textArea.append("CLIENT_ID: " + clientId + "\n\n");
            }

            // Close the result set and statement
            resultSet.close();
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
transactionFrame.getContentPane().add(new JScrollPane(textArea));

        transactionFrame.setVisible(true);
    }

    private void addTransaction() {
        JFrame addTransactionFrame = new JFrame("Add Transaction");
        addTransactionFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        addTransactionFrame.setSize(400, 300);

        // Create labels and text fields for transaction information
        JLabel transactionIdLabel = new JLabel("Transaction ID:");
        JTextField transactionIdField = new JTextField(10);
        JLabel transactionTypeLabel = new JLabel("Transaction Type:");
        JTextField transactionTypeField = new JTextField(10);
        JLabel amountLabel = new JLabel("Amount:");
        JTextField amountField = new JTextField(10);

        // Create a button to add the transaction
        JButton addButton = new JButton("Add");

        // Set the layout for the add transaction panel
        JPanel addTransactionPanel = new JPanel(new GridLayout(4, 2, 10, 10)); // 4 rows, 2
columns with 10px horizontal and vertical gaps

        // Add components to the add transaction panel
        addTransactionPanel.add(transactionIdLabel);
        addTransactionPanel.add(transactionIdField);
        addTransactionPanel.add(transactionTypeLabel);
        addTransactionPanel.add(transactionTypeField);
        addTransactionPanel.add(amountLabel);
        addTransactionPanel.add(amountField);
        addTransactionPanel.add(addButton);

        // Add an action listener to the add button
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
int transactionId = Integer.parseInt(transactionIdField.getText());
            String transactionType = transactionTypeField.getText();
            double amount = Double.parseDouble(amountField.getText());

            // Insert the transaction into the Transaction table
            try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                // Create a prepared statement
                PreparedStatement statement = connection.prepareStatement("INSERT INTO
Transaction (TRANSACTION_ID, TRANSACTION_TYPE, AMOUNT) VALUES (?, ?, ?)");
                statement.setInt(1, transactionId);
                statement.setString(2, transactionType);
                statement.setDouble(3, amount);

                // Execute the statement
                int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the insertion
                JOptionPane.showMessageDialog(addTransactionFrame, "Transaction added
successfully. Rows affected: " + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the add transaction panel to the add transaction frame
    addTransactionFrame.getContentPane().add(addTransactionPanel);

    addTransactionFrame.setVisible(true);
  }

  private void updateTransaction() {
    JFrame updateTransactionFrame = new JFrame("Update Transaction");
    updateTransactionFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
 JLabel transactionIdLabel = new JLabel("Transaction ID to update:");
        JTextField transactionIdField = new JTextField(10);
        JLabel amountLabel = new JLabel("Updated Amount:");
        JTextField amountField = new JTextField(10);

        // Create a button to update the transaction
        JButton updateButton = new JButton("Update");

        // Set the layout for the update transaction panel
        JPanel updateTransactionPanel = new JPanel(new GridLayout(3, 2, 10, 10)); // 3 rows, 2
columns with 10px horizontal and vertical gaps

        // Add components to the update transaction panel
        updateTransactionPanel.add(transactionIdLabel);
        updateTransactionPanel.add(transactionIdField);
        updateTransactionPanel.add(amountLabel);
        updateTransactionPanel.add(amountField);
        updateTransactionPanel.add(updateButton);

        // Add an action listener to the update button
        updateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Get the input values
                int transactionId = Integer.parseInt(transactionIdField.getText());
                double updatedAmount = Double.parseDouble(amountField.getText());

                // Update the amount in the Transaction table
                try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                    // Create a prepared statement
                    PreparedStatement statement = connection.prepareStatement("UPDATE
Transaction SET AMOUNT = ? WHERE TRANSACTION_ID = ?");
                    statement.setDouble(1, updatedAmount);
                    statement.setInt(2, transactionId);

                    // Execute the statement
                    int rowsAffected = statement.executeUpdate();
```

```
 JOptionPane.showMessageDialog(updateTransactionFrame, "Transaction updated
successfully. Rows affected: " + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the update transaction panel to the update transaction frame
    updateTransactionFrame.getContentPane().add(updateTransactionPanel);

    updateTransactionFrame.setVisible(true);
}

private void deleteTransaction() {
    JFrame deleteTransactionFrame = new JFrame("Delete Transaction");
    deleteTransactionFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    deleteTransactionFrame.setSize(400, 300);

    // Create labels and text fields for delete information
    JLabel transactionIdLabel = new JLabel("Transaction ID to delete:");
    JTextField transactionIdField = new JTextField(10);

    // Create a button to delete the transaction
    JButton deleteButton = new JButton("Delete");

    // Set the layout for the delete transaction panel
    JPanel deleteTransactionPanel = new JPanel(new GridLayout(2, 2, 10, 10)); // 2 rows, 2
columns with 10px horizontal and vertical gaps

    // Add components to the delete transaction panel
    deleteTransactionPanel.add(transactionIdLabel);
    deleteTransactionPanel.add(transactionIdField);
    deleteTransactionPanel.add(deleteButton);
```

```
 deleteButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Get the input value
                int transactionId = Integer.parseInt(transactionIdField.getText());

                // Delete the transaction from the Transaction table
                try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                    // Create a prepared statement
                    PreparedStatement statement = connection.prepareStatement("DELETE FROM
Transaction WHERE TRANSACTION_ID = ?");
                    statement.setInt(1, transactionId);

                    // Execute the statement
                    int rowsAffected = statement.executeUpdate();

                    // Display a message indicating the success of the deletion
                    JOptionPane.showMessageDialog(deleteTransactionFrame, "Transaction deleted
successfully. Rows affected: " + rowsAffected);

                    // Close the statement
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        });

        // Add the delete transaction panel to the delete transaction frame
        deleteTransactionFrame.getContentPane().add(deleteTransactionPanel);

        deleteTransactionFrame.setVisible(true);
    }

    private void displayTradeTable() {
        JFrame tradeFrame = new JFrame("Trade Table");
        tradeFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        tradeFrame.setSize(400, 300);
```

```
 JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setPreferredSize(new Dimension(380, 250)); // Adjust the preferred size as
needed

        // Establish a database connection
        try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
            // Create a statement
            Statement statement = connection.createStatement();

            // Execute the SELECT query on the Trade table
            ResultSet resultSet = statement.executeQuery("SELECT * FROM Trade");

            // Iterate through the result set and append the rows to the text area
            while (resultSet.next()) {
                int tradeId = resultSet.getInt("TRADE_ID");
                String tradeType = resultSet.getString("TRADE_TYPE");
                String currencyType = resultSet.getString("CURRENCY_TYPE");
                double tradeAmount = resultSet.getDouble("TRADE_AMOUNT");
                Date time = resultSet.getDate("TIME");
                int clientId = resultSet.getInt("CLIENT_ID");

                textArea.append("TRADE_ID: " + tradeId + "\n");
                textArea.append("TRADE_TYPE: " + tradeType + "\n");
                textArea.append("CURRENCY_TYPE: " + currencyType + "\n");
                textArea.append("TRADE_AMOUNT: " + tradeAmount + "\n");
                textArea.append("TIME: " + time + "\n");
                textArea.append("CLIENT_ID: " + clientId + "\n\n");
            }
            resultSet.close();
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
 tradeFrame.getContentPane().add(new JScrollPane(textArea));

        tradeFrame.setVisible(true);
    }

    private void addTrade() {
        JFrame addTradeFrame = new JFrame("Add Trade");
        addTradeFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        addTradeFrame.setSize(400, 300);

        // Create labels and text fields for trade information
        JLabel tradeIdLabel = new JLabel("Trade ID:");
        JTextField tradeIdField = new JTextField(10);
        JLabel tradeTypeLabel = new JLabel("Trade Type:");
        JTextField tradeTypeField = new JTextField(10);
        JLabel quantityLabel = new JLabel("Quantity:");
        JTextField quantityField = new JTextField(10);

        // Create a button to add the trade
        JButton addButton = new JButton("Add");

        // Set the layout for the add trade panel
        JPanel addTradePanel = new JPanel(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns
with 10px horizontal and vertical gaps

        // Add components to the add trade panel
        addTradePanel.add(tradeIdLabel);
        addTradePanel.add(tradeIdField);
        addTradePanel.add(tradeTypeLabel);
        addTradePanel.add(tradeTypeField);
        addTradePanel.add(quantityLabel);
        addTradePanel.add(quantityField);
        addTradePanel.add(addButton);
```

```
 addButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
             // Get the input values
             int tradeId = Integer.parseInt(tradeIdField.getText());
             String tradeType = tradeTypeField.getText();
             int quantity = Integer.parseInt(quantityField.getText());

             // Insert the trade into the Trade table
             try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                // Create a prepared statement
                PreparedStatement statement = connection.prepareStatement("INSERT INTO
Trade (TRADE_ID, TRADE_TYPE, QUANTITY) VALUES (?, ?, ?)");
                statement.setInt(1, tradeId);
                statement.setString(2, tradeType);
                statement.setInt(3, quantity);

                // Execute the statement
                int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the insertion
                JOptionPane.showMessageDialog(addTradeFrame, "Trade added successfully.
Rows affected: " + rowsAffected);
                  statement.close();
             } catch (SQLException ex) {
                ex.printStackTrace();
             }
          }
       });
       addTradeFrame.getContentPane().add(addTradePanel);

       addTradeFrame.setVisible(true);
    }
    private void updateTrade() {
       JFrame updateTradeFrame = new JFrame("Update Trade");
       updateTradeFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
       updateTradeFrame.setSize(400, 300);
```

```
 JLabel tradeIdLabel = new JLabel("Trade ID to update:");
        JTextField tradeIdField = new JTextField(10);
        JLabel quantityLabel = new JLabel("Updated Quantity:");
        JTextField quantityField = new JTextField(10);
        JButton updateButton = new JButton("Update");
        JPanel updateTradePanel = new JPanel(new GridLayout(3, 2, 10, 10)); // 3 rows, 2
columns with 10px horizontal and vertical gaps

        updateTradePanel.add(tradeIdLabel);
        updateTradePanel.add(tradeIdField);
        updateTradePanel.add(quantityLabel);
        updateTradePanel.add(quantityField);
        updateTradePanel.add(updateButton);

        // Add an action listener to the update button
        updateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Get the input values
                int tradeId = Integer.parseInt(tradeIdField.getText());
                int updatedQuantity = Integer.parseInt(quantityField.getText());

                // Update the quantity in the Trade table
                try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                    // Create a prepared statement
                    PreparedStatement statement = connection.prepareStatement("UPDATE Trade
SET QUANTITY = ? WHERE TRADE_ID = ?");
                    statement.setInt(1, updatedQuantity);
                    statement.setInt(2, tradeId);
                      int rowsAffected = statement.executeUpdate();
                      JOptionPane.showMessageDialog(updateTradeFrame, "Trade updated
successfully. Rows affected: " + rowsAffected);
                      statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        });
```

```
 updateTradeFrame.getContentPane().add(updateTradePanel);

        updateTradeFrame.setVisible(true);
    }

    private void deleteTrade() {
        JFrame deleteTradeFrame = new JFrame("Delete Trade");
        deleteTradeFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        deleteTradeFrame.setSize(400, 300);

        // Create labels and text fields for delete information
        JLabel tradeIdLabel = new JLabel("Trade ID to delete:");
        JTextField tradeIdField = new JTextField(10);

        // Create a button to delete the trade
        JButton deleteButton = new JButton("Delete");

        // Set the layout for the delete trade panel
        JPanel deleteTradePanel = new JPanel(new GridLayout(2, 2, 10, 10)); // 2 rows, 2
columns with 10px horizontal and vertical gaps

        // Add components to the delete trade panel
        deleteTradePanel.add(tradeIdLabel);
        deleteTradePanel.add(tradeIdField);
        deleteTradePanel.add(deleteButton);

        // Add an action listener to the delete button
        deleteButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Get the input value
                int tradeId = Integer.parseInt(tradeIdField.getText());

                // Delete the trade from the Trade table
                try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                    // Create a prepared statement
                    PreparedStatement statement = connection.prepareStatement("DELETE FROM
Trade WHERE TRADE_ID = ?");
                    statement.setInt(1, tradeId);
```

```
 int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the deletion
                JOptionPane.showMessageDialog(deleteTradeFrame, "Trade deleted successfully.
Rows affected: " + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the delete trade panel to the delete trade frame
    deleteTradeFrame.getContentPane().add(deleteTradePanel);

    deleteTradeFrame.setVisible(true);
  }



  private void displayAccountTable() {
    JFrame accountFrame = new JFrame("Account Table");
    accountFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    accountFrame.setSize(400, 300);

    // Create a JTextArea to display the Account table rows
    JTextArea textArea = new JTextArea();
    textArea.setEditable(false);
    textArea.setLineWrap(true);
    textArea.setWrapStyleWord(true);
    textArea.setPreferredSize(new Dimension(380, 250)); // Adjust the preferred size as
needed
```

```
 try (Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD)) {
            // Create a statement
            Statement statement = connection.createStatement();

            // Execute the SELECT query on the Account table
            ResultSet resultSet = statement.executeQuery("SELECT * FROM Account");

            // Iterate through the result set and append the rows to the text area
            while (resultSet.next()) {
                String accountNumber = resultSet.getString("ACCOUNT_NUMBER");
                double balance = resultSet.getDouble("ACCOUNT_BALANCE");
                int clientId = resultSet.getInt("CLIENT_ID");

                textArea.append("ACCOUNT_TYPE: " + accountNumber + "\n");
                textArea.append("BALANCE: " + balance + "\n");
                textArea.append("CLIENT_ID: " + clientId + "\n\n");
            }
            // Close the result set and statement
            resultSet.close();
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        // Add the text area to the accountFrame
        accountFrame.getContentPane().add(new JScrollPane(textArea));

        accountFrame.setVisible(true);
    }
    private void addAccount() {
        JFrame addAccountFrame = new JFrame("Add Account");
        addAccountFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        addAccountFrame.setSize(400, 300);

        // Create labels and text fields for account information
        JLabel accountIdLabel = new JLabel("Account ID:");
        JTextField accountIdField = new JTextField(10);
```

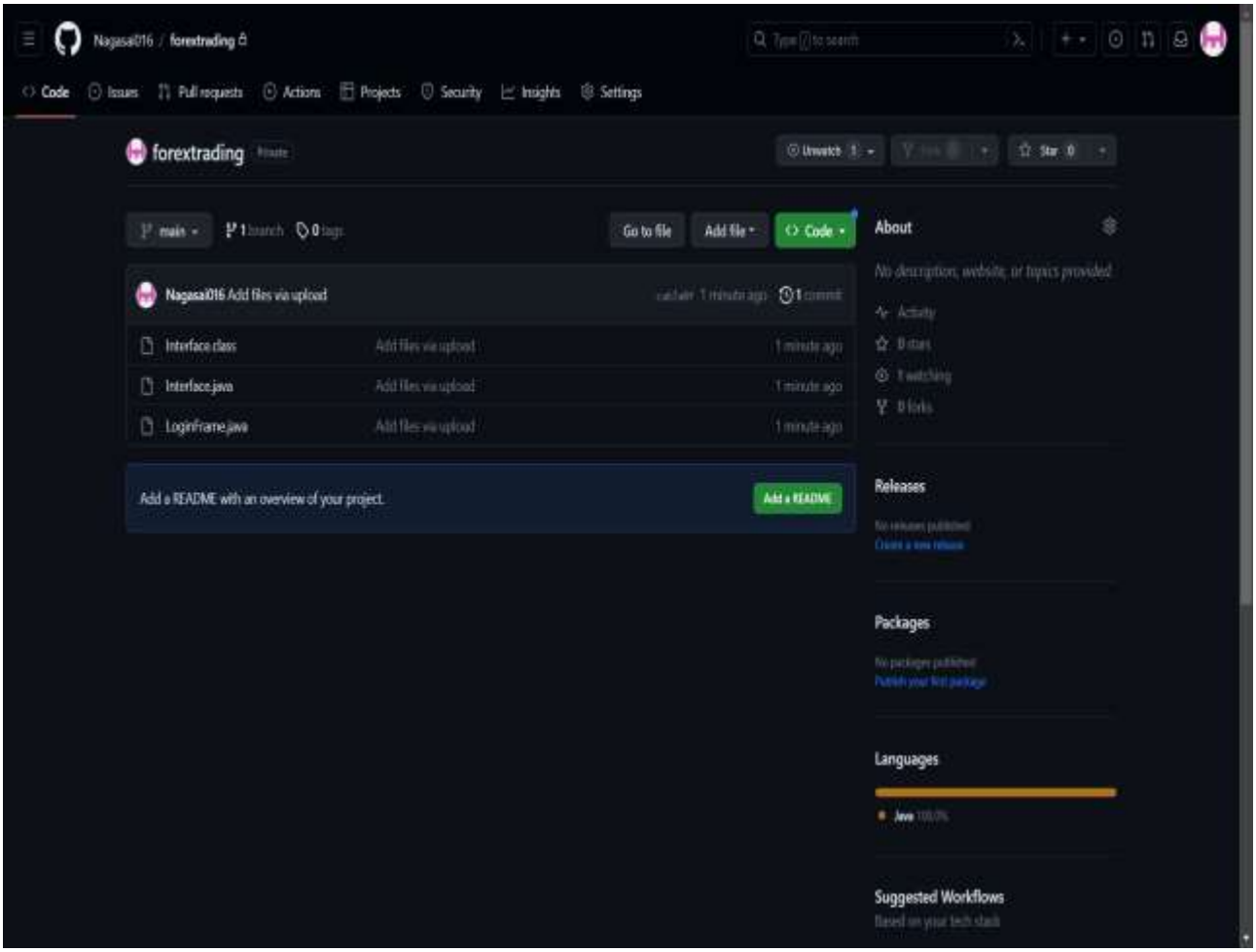
```
 JLabel accountTypeLabel = new JLabel("Account Type:");
        JTextField accountTypeField = new JTextField(10);
        JLabel balanceLabel = new JLabel("Balance:");
        JTextField balanceField = new JTextField(10);
        JLabel clientIdLabel = new JLabel("Client ID:");
        JTextField clientIdField = new JTextField(10);

        // Create a button to add the account
        JButton addButton = new JButton("Add");
        // Set the layout for the add account panel
        JPanel addAccountPanel = new JPanel(new GridLayout(5, 2, 10, 10)); // 5 rows, 2
columns with 10px horizontal and vertical gaps
        addAccountPanel.add(accountIdLabel);
        addAccountPanel.add(accountIdField);
        addAccountPanel.add(accountTypeLabel);
        addAccountPanel.add(accountTypeField);
        addAccountPanel.add(balanceLabel);
        addAccountPanel.add(balanceField);
        addAccountPanel.add(clientIdLabel);
        addAccountPanel.add(clientIdField);
        addAccountPanel.add(addButton); addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Get the input values
                int accountNumber = Integer.parseInt(accountNumberField.getText());
                double accountBalance = Double.parseDouble(accountBalanceField.getText());
                int clientId = Integer.parseInt(clientIdField.getText());

                // Insert the account into the Account table
                try (Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD)) {
                    // Create a prepared statement
                    PreparedStatement statement = connection.prepareStatement("INSERT INTO
Account (ACCOUNT_NUMBER, ACCOUNT_BALANCE, CLIENT_ID) VALUES (?, ?,
?)");
                    statement.setInt(1, accountNumber);
                    statement.setDouble(2, accountBalance);
                    statement.setInt(3, clientId);
```

```
 // Execute the statement
                int rowsAffected = statement.executeUpdate();

                // Display a message indicating the success of the insertion
                JOptionPane.showMessageDialog(addAccountFrame, "Account added
successfully. Rows affected: " + rowsAffected);

                // Close the statement
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    });

    // Add the add account panel to the add account frame
    addAccountFrame.getContentPane().add(addAccountPanel);

    addAccountFrame.setVisible(true);
  }
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            Interface forex = new Interface();
            forex.setVisible(true);
        }
    });
  }
}
```

GitHub Links and Folder Structure

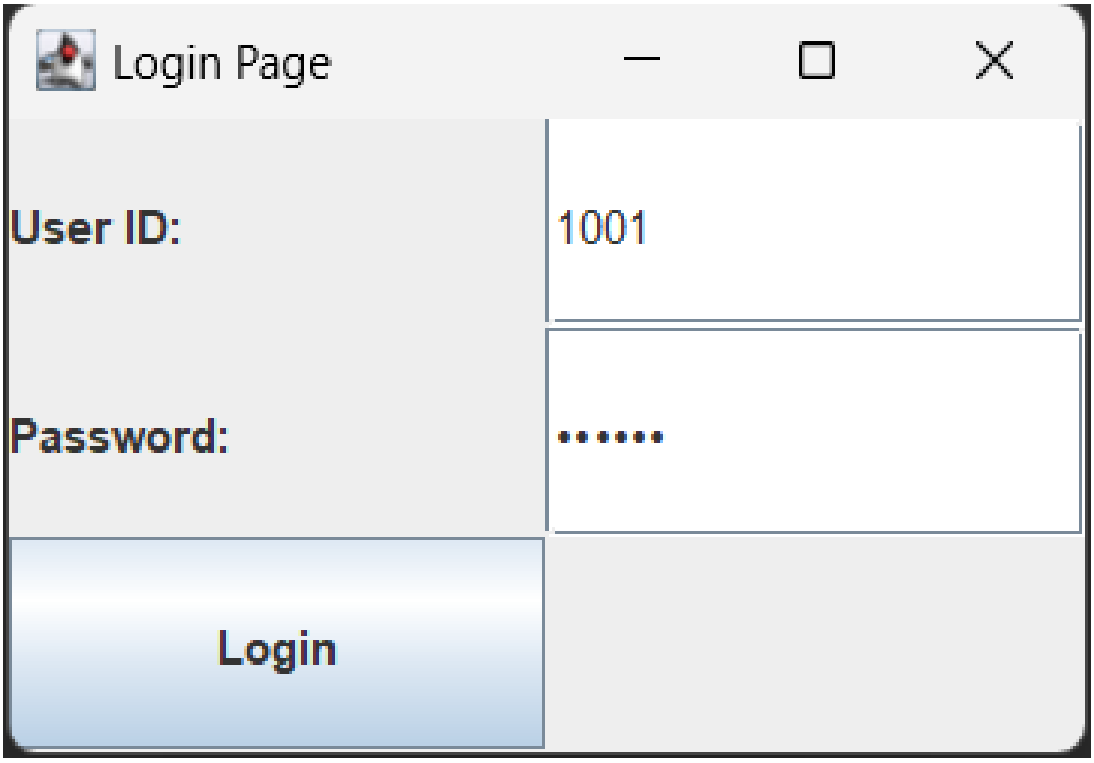
Link: https://github.com/Nagasai016/forextrading

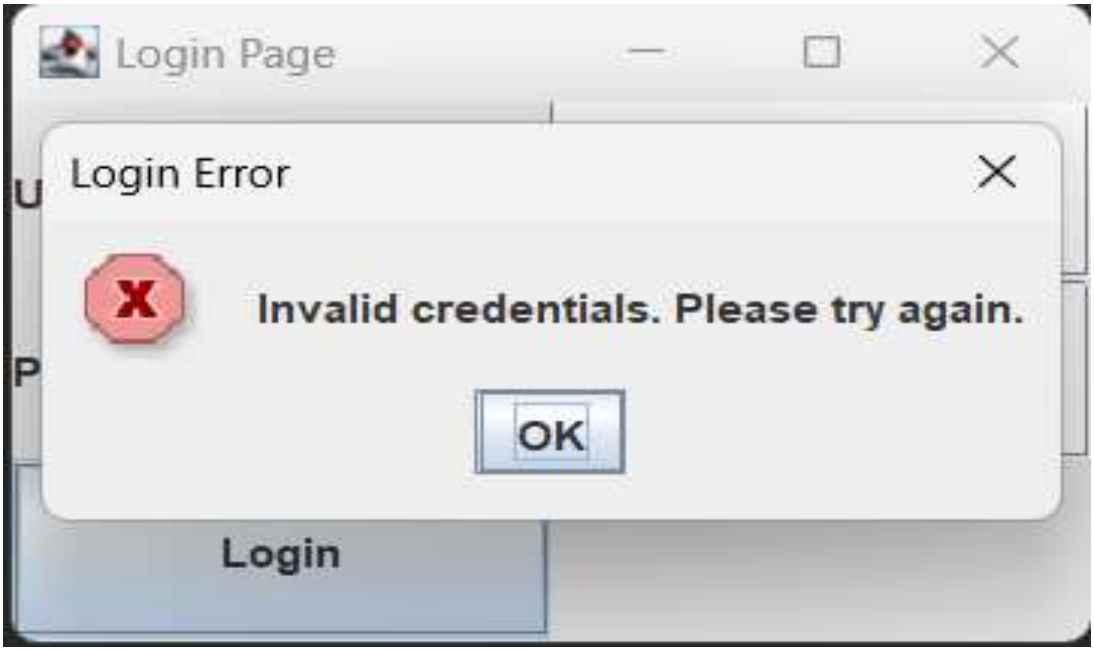
Folder Structure:



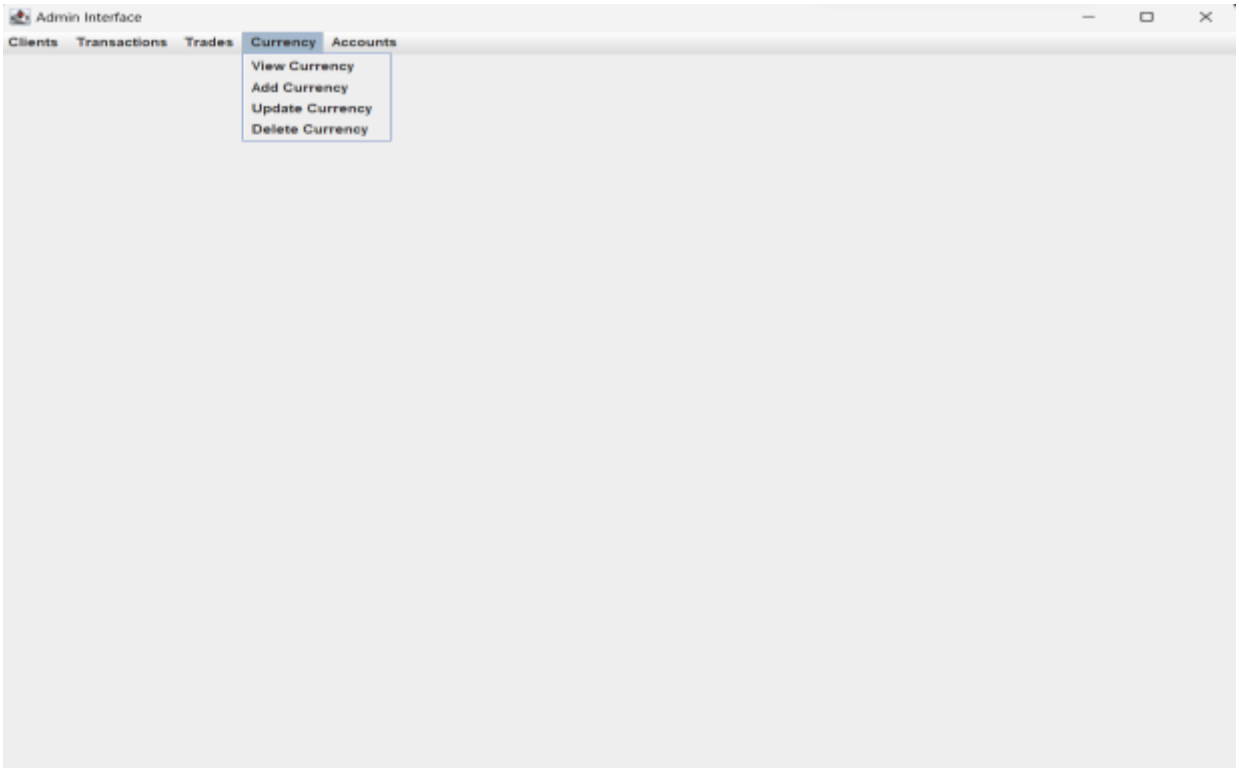
TESTING

INTRODUCTION PAGE:

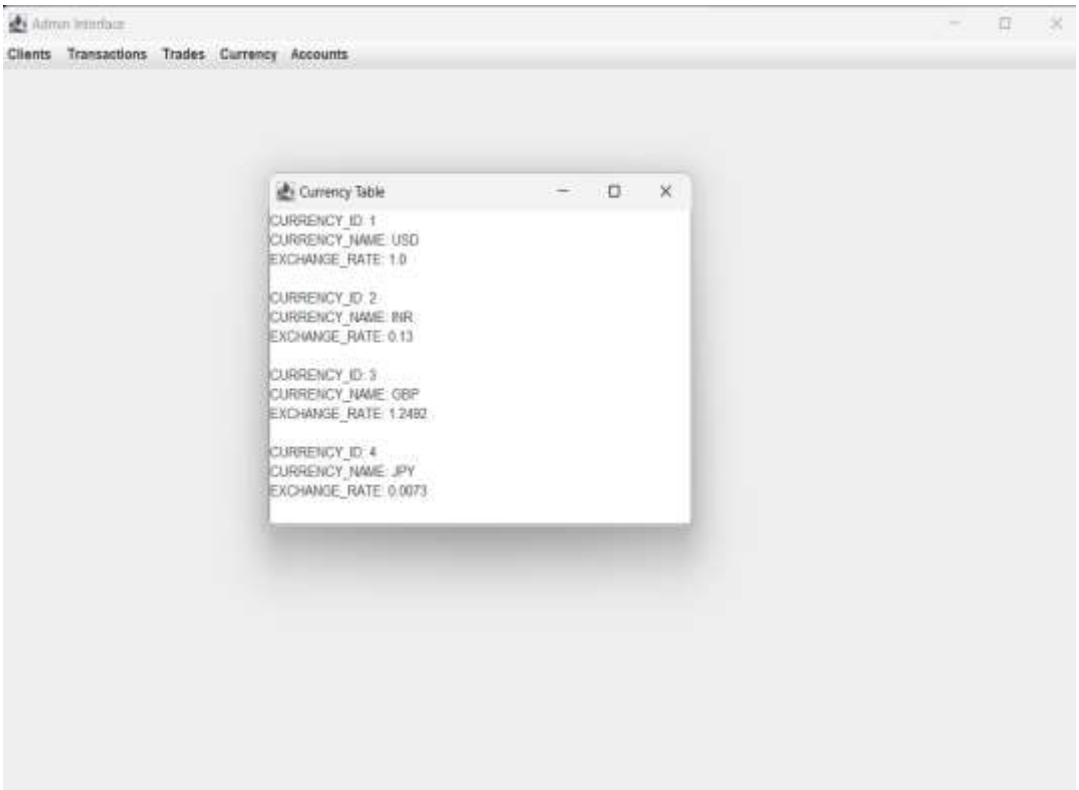

Login Page
User ID:
1001
Password:
Login

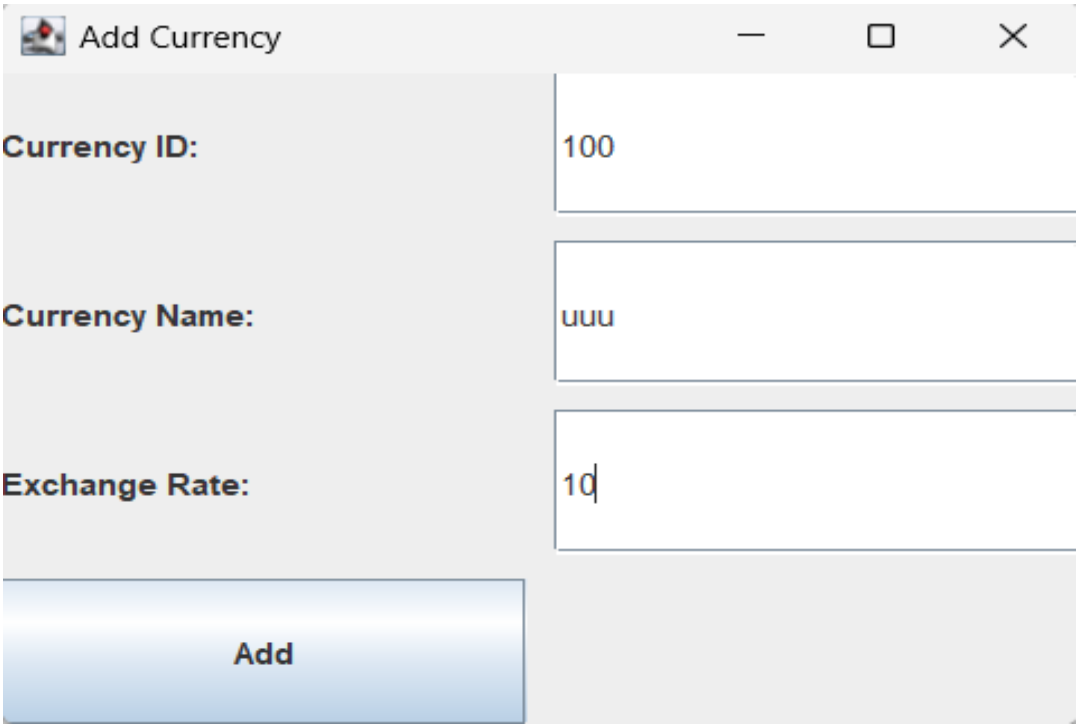

Login Page
Login Error
Invalid credentials. Please try again.
OK
Login

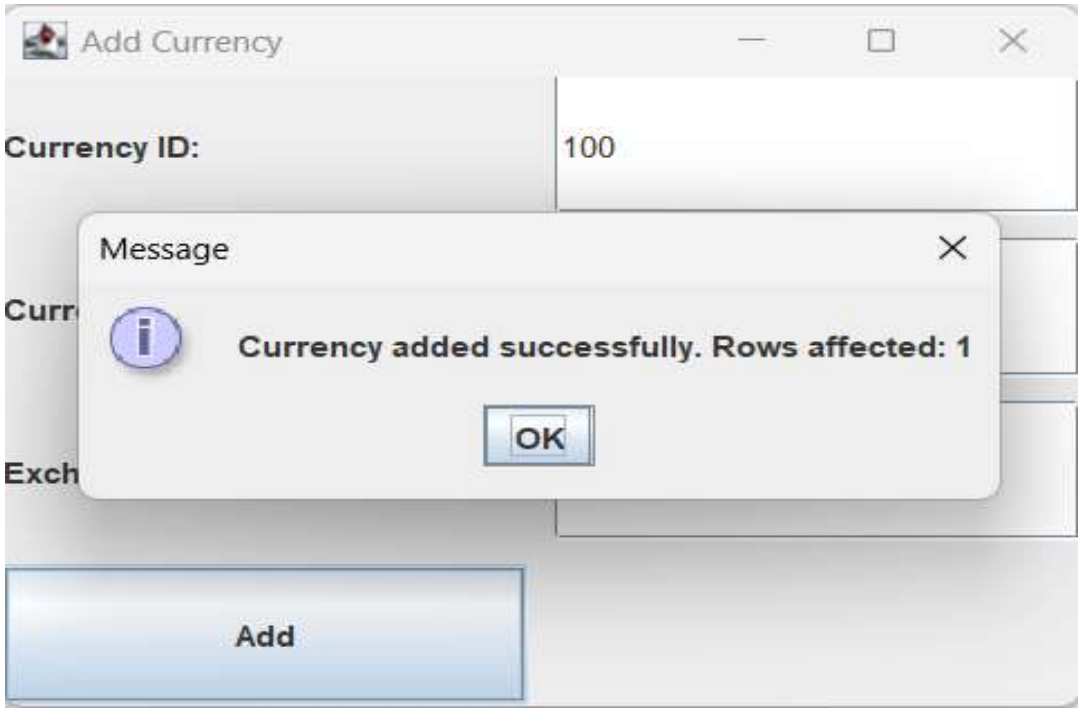
FOREX TRADING DATABASE MANAGEMENT

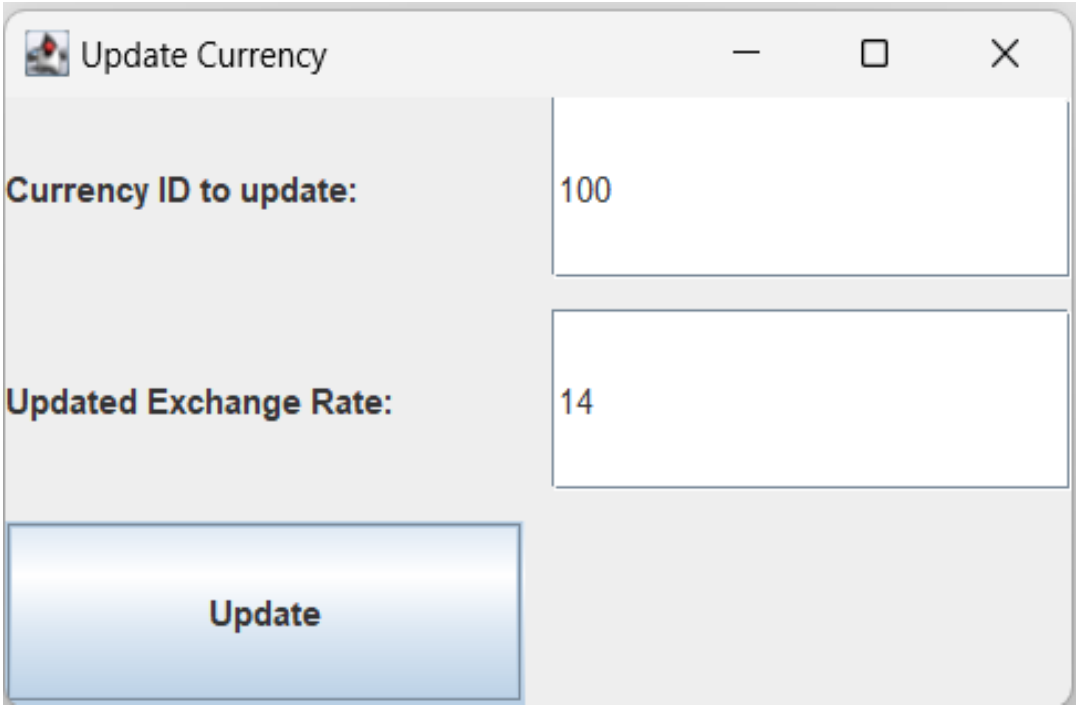


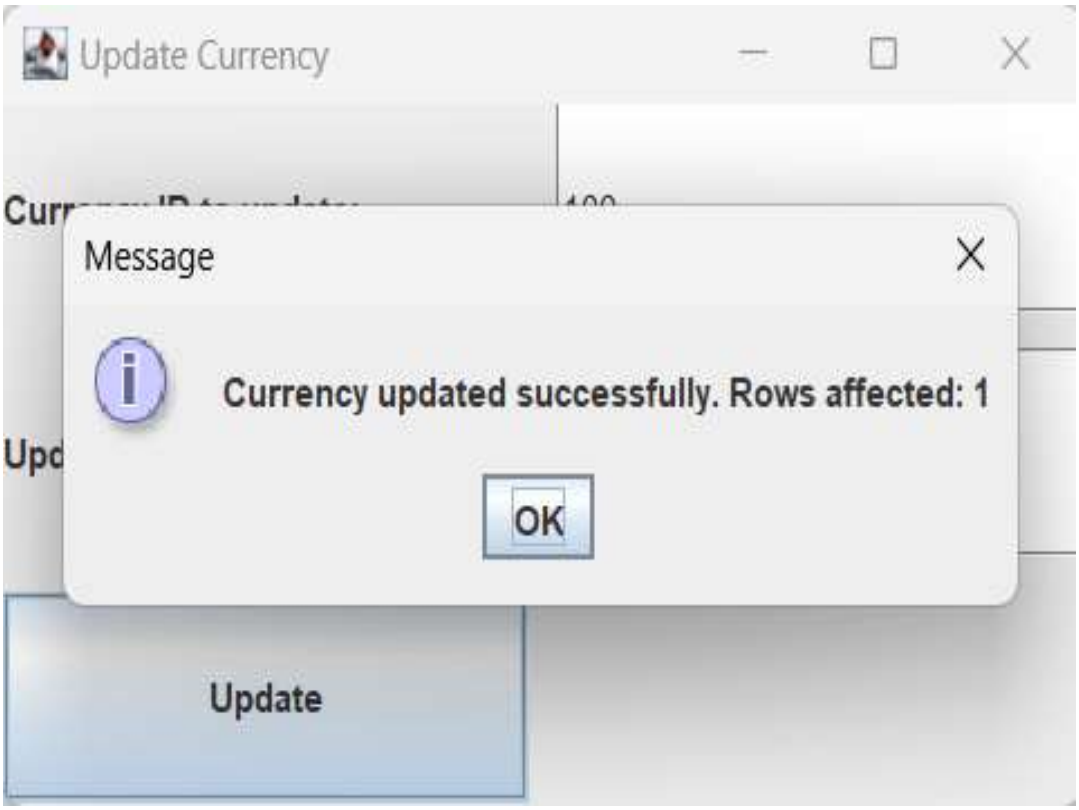
1602-21-737-036
Nagasai

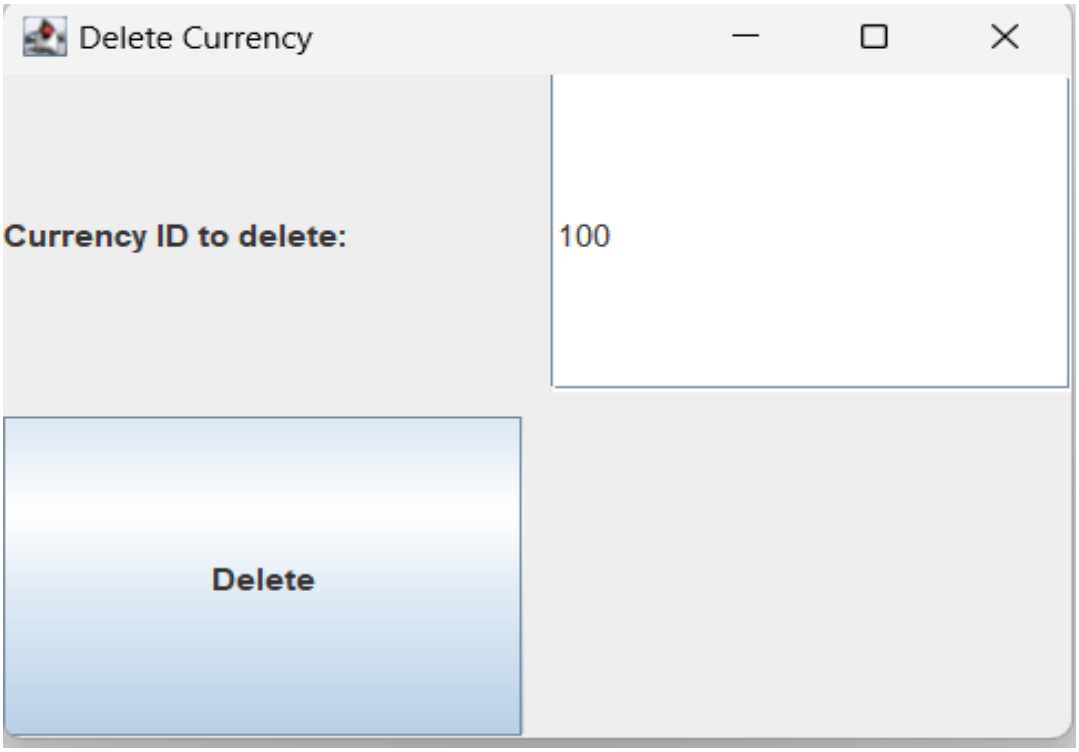

Admin Interface
Clients
Transactions
Trades
Currency
Accounts
Currency Table
CURRENCY_ID: 1
CURRENCY_NAME: USD
EXCHANGE_RATE: 1.0
CURRENCY_ID: 2
CURRENCY_NAME: INR
EXCHANGE_RATE: 0.13
CURRENCY_ID: 3
CURRENCY_NAME: GBP
EXCHANGE_RATE: 1.2492
CURRENCY_ID: 4
CURRENCY_NAME: JPY
EXCHANGE_RATE: 0.0073

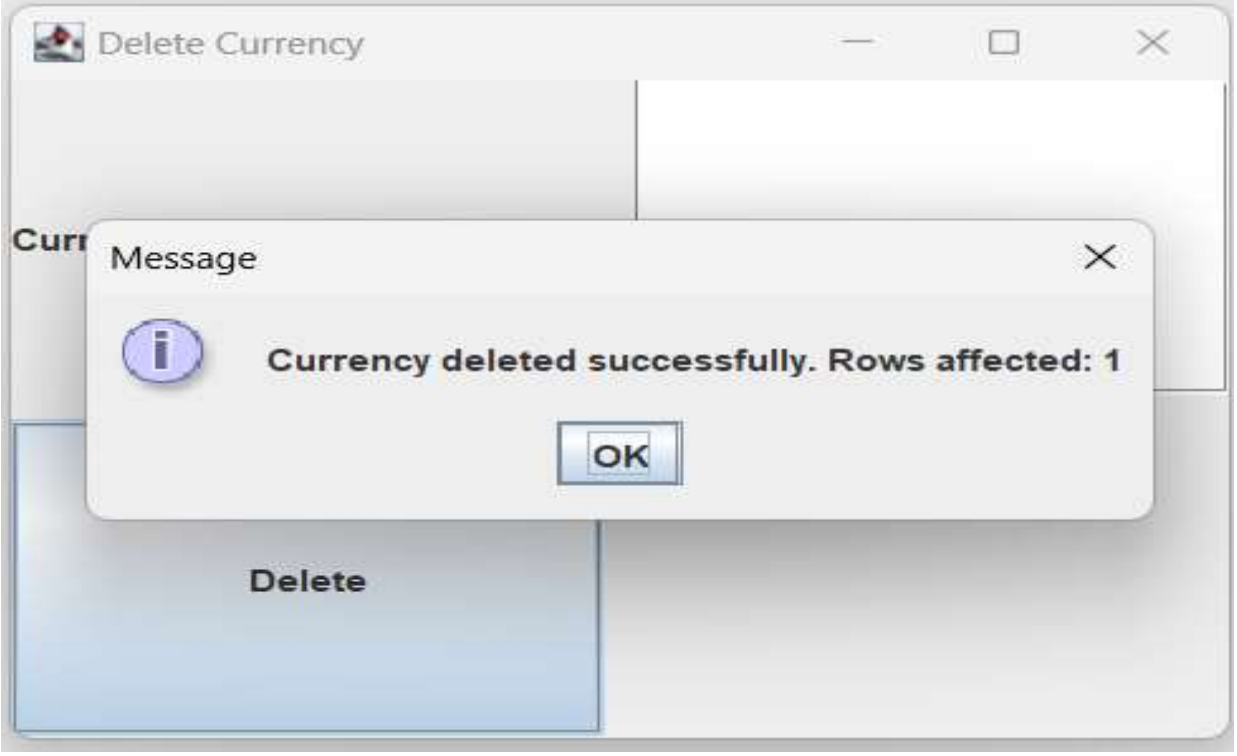

Add Currency
Currency ID:
100
Currency Name:
uuu
Exchange Rate:
10
Add


Add Currency
Currency ID:
100
Message
Currency added successfully. Rows affected: 1
OK
Add


Update Currency
Currency ID to update:
100
Updated Exchange Rate:
14
Update


Update Currency
Message
Currency updated successfully. Rows affected: 1
OK
Update


Delete Currency
Currency ID to delete:
100
Delete


Delete Currency
Curr
Message
Currency deleted successfully. Rows affected: 1
OK
Delete

RESULTS

I have successfully completed the mini-project **“FOREX TRADING DATABASE MANAGEMENT ”** .

DISCUSSION AND FUTURE WORK

This project focuses on forex trading database management and aims to provide a solid foundation for organizing and analyzing trading data. The current implementation has a basic user interface, but the future scope involves enhancing the user experience and incorporating additional features.

One potential improvement is to integrate graphical elements into the user interface, making it more visually appealing and intuitive. Graphs, charts, and visual representations of data can help users analyze trends, track performance, and make informed trading decisions.

Another valuable addition would be to allow users to upload their trading data, such as trade history, account statements, and performance metrics. By integrating this functionality, the system can provide more accurate insights and personalized suggestions based on the user's specific trading data.

Furthermore, incorporating a feedback system would be beneficial. This feature would enable users to provide their valuable input and suggestions, helping to improve the application over time. Additionally, making the feedback publicly viewable can foster transparency and credibility, attracting more users to utilize the app.

Overall, the future scope for the forex trading database management project includes enhancing the user interface with graphics, allowing users to upload trading data for personalized suggestions, and implementing a feedback system for continuous improvement and public engagement.

REFERENCES

- https://docs.oracle.com/javase/7/docs/api/
- https://www.javatpoint.com/java-swing
- https://stackoverflow.com/
- https://github.com/