ATM SIMULATION SYSTEM

MINI PROJECT REPORT

Submitted by

YASHWANTH [RA2311003011081]
JAHNAVI.M [RA2311003011089]
VEERAVARDHAN REDDY[RA2311003011099]
Y.P.V.S.SAI KUMAR[RA2311003011103]

Under the Guidance of

DR.GEETHA.R

21CSC203P – ADVANCED PROGRAMMING PRACTICES

DEPARTMENT OF COMPUTING TECHNOLOGIES

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2024

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the 21CSC203P Advance Programming Practice course project report titled **"ATM SIMULATION SYSTEM"** is the bonafide work done by **YASHWANTH[RA2311003011081],JAHNAVI.M[RA2311003011089],VEERA VARDHAN REDDY[RA2311003011099], Y.P.V.S.SAI KUMAR[RA2311003011103] of II Year/III Sem B.Tech(CSE)** who carried out the mini project under my supervision.

SIGNATURE

DR.GEETHA.R
Assistant Professor

Department of Computing Technologies,
School of Computing,
SRM Institute of Science and Technology
Kattankulathur

Department of Computing Technologies Kattankulathur 603 203 *SRM IST*

SIGNATURE

DR.NIRANJANA.G
Head of the Department
Professor and Head

Department of Computing Technologies,
School of Computing,
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

The program simulates an ATM with the ability to exchange currencies based on daily market rates. It handles user registration, displays balances in different currencies, and allows transactions like deposits, withdrawals, and currency exchange. The exchange rates are fetched dynamically (in real-life scenarios, they could be from an API), and the user can perform operations like depositing or withdrawing money in various currencies, with the system converting between them based on the current market rate.

The Currency Exchange Using ATM Simulation program is a conceptual implementation of an Automated Teller Machine (ATM) system that allows users to manage their accounts, perform transactions like deposits, withdrawals, and exchange currencies. It simulates how an ATM would allow users to interact with their funds, convert between different currencies based on live market rates, and log every operation for future reference.

The ATM simulates the basic functions of an ATM, including deposits, withdrawals, and currency exchanges. Users can deposit money into their accounts in their home currency, withdraw funds in the same currency, and exchange their balance into different currencies at real-time rates.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honourable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavours. We would like to express my warmth of gratitude to our **Registrar Dr. S.PONNUSAMY,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.GOPAL,** for bringing out novelty in all executions. We would like to express my heartfelt thanks to our Chairperson, School of Computing **Dr. REVATHI VENKATARAMAN,** for imparting confidence to complete my course project.

We extend my gratitude to our Associate Chairperson **Dr. PUSHPALATHA. M**, **Professor** and our **HOD Dr. NIRANJANA. G, Professor and Head, Department of Computing Technologies** for their Support.

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu.G, Professor, Department of Data Science and Business Systems** and **Course Coordinators Dr. MANJULA. R, Assistant Professor and Dr. N.A.S VINOTH, Assistant Professor** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **DR.GEETHA.R Assistant Professor, Department of Computing Technologies** for her assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

TABLE OF CONTENTS

INTRODUCTION

In today's globalized world, financial transactions have become an essential part of daily life. Banks and financial institutions offer various services to help individuals manage their money, make payments, and exchange currencies. One of the most common tools for facilitating these transactions is the **Automated Teller Machine (ATM)**. ATMs allow users to withdraw money, check their balances, transfer funds, and in some advanced systems, perform currency exchanges. A more sophisticated version of an ATM is one that can simulate the exchange of currencies, which is the focal point of the **Currency Exchange Using ATM Simulation**.

The **ATM simulation** program with currency exchange functionality provides a platform where users can not only interact with their accounts to perform standard banking tasks like depositing and withdrawing money but also exchange currencies based on live market rates. This simulation concept brings together the core principles of banking transactions and foreign exchange (forex), demonstrating how technology can bridge these aspects to create a comprehensive financial system.

The emergence of ATMs revolutionized the banking industry by providing customers with 24/7 access to their accounts. ATMs are self-service machines that allow users to perform banking tasks such as withdrawing cash, checking account balances, transferring funds between accounts, and more, without the need to visit a bank branch. Initially, ATMs served as a way to withdraw cash from a user's bank account, but as technology evolved, so did the capabilities of ATMs.

Modern ATMs are now equipped with the ability to perform a variety of functions such as depositing money, paying bills, transferring funds, and even buying stamps or tickets. One of the most innovative features being introduced in some advanced ATMs today is the currency exchange, which allows customers to convert their local currency into foreign currency directly from the ATM.

LITERATURE SURVEY

A literature survey for the **Currency Exchange Using ATM Simulation** explores existing studies, technologies, and systems related to automated teller machines, currency exchange, financial transactions, and simulation models. This survey will highlight relevant advancements in these areas, examining how the convergence of banking technology and foreign exchange services within ATM systems has been discussed in academic and industry literature. It will focus on key areas such as ATM technology, currency exchange mechanisms, and simulation tools in financial systems.

REQUIREMENT ANALYSIS

The requirement analysis for a **Currency Exchange Using ATM Simulation** program aims to identify the key components, functionalities, and system requirements needed to implement an ATM system that allows users to perform standard banking tasks as well as currency exchange transactions. This analysis breaks down user needs, functional requirements, non-functional requirements, and technical specifications essential to develop a robust simulation system.

1. User Requirements

Understanding user needs is crucial in designing an ATM simulation that offers a realistic and user-friendly experience. Key user requirements for this system include:

- **User Registration and Login**: Users need a secure way to register for the system and authenticate their identity before performing any transactions.
- **Account Balance Viewing**: Users should be able to view their account balance in their registered home currency.
- **Deposits and Withdrawals**: Users must be able to deposit money into their account and withdraw funds within the available balance.
- **Currency Exchange**: Users need the ability to convert their account balance from one currency to another based on current exchange rates.
- **Transaction History**: Users should have access to a detailed history of their transactions, including deposits, withdrawals, and currency exchanges.
- **User Interface**: Users require an intuitive interface, either command-line-based or graphical, to interact with the ATM system easily.

2. Functional Requirements

Functional requirements define the core functions that the simulation system must perform:

1. **User Account Management**
    - **Registration**: Users can register by providing their name, an initial balance,

and their home currency.
    - **Login/Authentication**: The system should verify user credentials to allow secure access to the ATM services.
    - **Balance Check**: The system should display the user's current balance in their selected currency.
2. **Currency Exchange Functionality**
    - **Real-Time Exchange Rates**: The system should display current exchange rates for multiple currencies (e.g., USD, EUR, INR).
    - **Currency Conversion**: The system should convert a specified amount from one currency to another using the latest exchange rate.
    - **Balance Update**: After conversion, the system should update the user's balance in the new currency and log the transaction.
3. **Financial Transactions**
    - **Deposit**: Users can deposit funds into their account, and the system should update the balance accordingly.
    - **Withdrawal**: Users can withdraw funds within the limits of their current balance.
    - **Transaction Logging**: Each transaction (deposit, withdrawal, currency exchange) should be logged in a database with details including transaction type, amount, date, and currency involved.
4. **Transaction History**
    - **Viewing History**: Users should be able to view a detailed transaction history, including deposits, withdrawals, and exchanges, organized by date.
5. **Database Management**
    - **User Data Storage**: A database should securely store user account information, including balance and transaction history.
    - **Exchange Rate Storage**: The system should store exchange rates, either through hardcoded values or by updating from an API (in real-world applications).
    - **Transaction Logs**: The database should maintain transaction records for future reference and accountability.

3. Non-Functional Requirements

These requirements define the system's qualities and constraints, focusing on performance, security, and usability:

1. **Performance**:
    - The system should provide quick response times for all transactions (e.g., less than 2 seconds per transaction).
    - The exchange rate calculation should be instantaneous to maintain the accuracy of conversions.
2. **Security**:
    - The system must ensure data security and protect sensitive user information.
    - Secure authentication methods (e.g., password encryption) should be

implemented for user login.
    - All transactions should be logged securely to prevent unauthorized access.
3. **Usability**:
    - The user interface should be intuitive and easy to navigate, allowing users to perform transactions without extensive training.
    - Error messages and instructions should be clear, guiding users through each transaction smoothly.
4. **Reliability**:
    - The simulation should handle typical use cases, such as deposits, withdrawals, and currency exchanges, without failures.
    - The database should maintain data integrity, ensuring that all transactions are accurately recorded and accessible.
5. **Scalability**:
    - The database and backend should be designed to support multiple users and large transaction logs, allowing the system to scale for future use cases.
    - The system should be able to integrate real-time APIs if required in the future for live exchange rates.

4. Technical Requirements

Technical requirements specify the tools, technologies, and frameworks needed to build the simulation system effectively:

1. **Programming Language**:
    - **Java** will be the primary programming language, as it is compatible with NetBeans and well-suited for backend development.
2. **Integrated Development Environment (IDE)**:
    - **NetBeans** will be used as the primary IDE for developing, testing, and debugging the application, as the user is familiar with this environment.
3. **Database**:
    - A relational database management system, such as **MySQL**, will be used to store user data, transaction history, and exchange rates.
    - The database will maintain tables for user details, balances, transaction logs, and exchange rates.
4. **User Interface (UI)**:
    - A command-line interface (CLI) can be used for a basic version, while a GUI-based interface can be added for better usability.
    - In a more advanced setup, Java Swing or JavaFX could be used for creating a user-friendly GUI.
5. **Exchange Rate API (Optional)**:
    - To simulate real-time currency exchange, the system could integrate with a free exchange rate API like **Open Exchange Rates** or **ExchangeRate-API** (this would require internet access and API key management).
    - For a simpler simulation, exchange rates can be hardcoded and periodically updated manually.

6. **Error Handling**:
    - The system should include error-handling mechanisms for issues like invalid login attempts, insufficient balance for withdrawals, and incorrect currency conversions.

5. System Architecture

The system architecture will follow a **three-tier architecture**:

1. **Presentation Layer (UI)**: This layer will handle user interactions and interface design, taking input for deposits, withdrawals, currency exchanges, and showing transaction history.
2. **Application Layer (Business Logic)**: This layer will contain the core functionality, handling user registration, login, transactions, and currency conversion calculations.
3. **Data Layer (Database)**: This layer will interact with the database to retrieve, update, and log user details, balances, and transactions.

6. User Scenarios

The following scenarios describe how users will interact with the system:

1. **User Registration**: A new user registers an account with their name, initial deposit, and home currency. The system saves the information to the database and confirms successful registration.
2. **Login and Balance Check**: The user logs in to view their current balance in their home currency. The system authenticates the user and retrieves balance information.
3. **Deposit and Withdrawal**: The user selects to deposit or withdraw an amount. The system verifies sufficient balance for withdrawals and updates the balance accordingly.
4. **Currency Exchange**: The user selects an amount to convert to a different currency. The system calculates the converted amount using the current exchange rate, updates the balance in the new currency, and logs the transaction.
5. **Transaction History**: The user accesses their transaction history to view details of past deposits, withdrawals, and exchanges.

7. Future Enhancements

Potential future enhancements for the currency exchange ATM simulation could include:

- **Real-Time Exchange Rate Updates**: Integrate an API for live forex rates, allowing the system to use real-time rates for currency conversions.
- **Enhanced Security Features**: Implement two-factor authentication, biometric verification, and advanced encryption.
- **Mobile or Web-Based Interface**: Extend the system to work on web or mobile platforms, allowing users to access their accounts remotely.
- **Predictive Analysis for Exchange Rates**: Implement basic machine learning to suggest optimal times for currency exchange based on historical trends.

LOGIN PAGE CODE

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.*;


class BackgroundPanel extends JPanel {
    private Image backgroundImage;

    public BackgroundPanel(String fileName) {
        try {
            backgroundImage = new ImageIcon(fileName).getImage();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (backgroundImage != null) {
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), null);
        }
    }
}

public class LoginForm {
    private static final String DB_URL = "jdbc:postgresql://localhost:5432/app_proj";
    private static final String DB_USER = "postgres";
    private static final String DB_PASSWORD = "6575";
    private static final String AUTH_QUERY = "SELECT * FROM public.user WHERE email = ?
AND password = ?";
    public static void main(String[] args) {
        JFrame frame = new JFrame("Login Form");
        frame.setSize(1000, 700);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);
        BackgroundPanel panel = new BackgroundPanel("back1.png");
```

```
panel.setBounds(0, 0, 1000, 700);
panel.setLayout(null);
frame.add(panel);
ImageIcon logoIcon = new ImageIcon("n1.png");
Image image = logoIcon.getImage();
Image scaledImage = image.getScaledInstance(200, 200, Image.SCALE_SMOOTH);
logoIcon = new ImageIcon(scaledImage);
JLabel logoLabel = new JLabel(logoIcon);
logoLabel.setBounds(100, 10, 200, 200);
panel.add(logoLabel);

JLabel welcomeLabel = new JLabel("Welcome to IntelliAC");
welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);
welcomeLabel.setBounds(100, 220, 200, 30);
welcomeLabel.setFont(new Font("Times New Roman", Font.BOLD, 18));
welcomeLabel.setForeground(Color.RED);
panel.add(welcomeLabel);
JLabel loginLabel = new JLabel("Login");
loginLabel.setHorizontalAlignment(SwingConstants.CENTER);
loginLabel.setBounds(150, 260, 100, 25);
loginLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));
loginLabel.setForeground(Color.WHITE);
panel.add(loginLabel);
JTextField emailField = new JTextField("email@example.com");
emailField.setBounds(100, 300, 200, 30);
emailField.setToolTipText("Enter your Email");
emailField.setFont(new Font("Times New Roman", Font.PLAIN, 14));
emailField.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, 2));
panel.add(emailField);
JPasswordField passwordField = new JPasswordField("password");
passwordField.setBounds(100, 340, 200, 30);
passwordField.setToolTipText("Enter your Password");
passwordField.setFont(new Font("Times New Roman", Font.PLAIN, 14));
passwordField.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, 2));
panel.add(passwordField);
JButton loginButton = new JButton("Login");
loginButton.setBounds(100, 380, 200, 30);
loginButton.setFont(new Font("Times New Roman", Font.PLAIN, 14));
loginButton.setBackground(Color.PINK);
loginButton.setForeground(Color.BLACK);
loginButton.setBorder(BorderFactory.createLineBorder(Color.CYAN, 2));
panel.add(loginButton);
loginButton.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseEntered(java.awt.event.MouseEvent evt) {
      loginButton.setBackground(Color.CYAN);
```

```
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            loginButton.setBackground(Color.PINK);
        }
    });
    loginButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String email = emailField.getText();
            String password = new String(passwordField.getPassword());
            try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
                 PreparedStatement preparedStatement =
connection.prepareStatement(AUTH_QUERY)) {
                preparedStatement.setString(1, email);
                preparedStatement.setString(2, password); // Correct this line
                try (ResultSet resultSet = preparedStatement.executeQuery()) {
                    if (resultSet.next()) {
                        System.out.println("Login successful");
                        // Add logic here to proceed after successful login
                    } else {
                        System.out.println("Invalid email or password");
                    }
                }
            } catch (SQLException ex) {
                System.out.println("Error occurred during database connection: " + ex.getMessage());
            }
        }
    });

    frame.setVisible(true);
  }
```

BALANCE ENQUIRY

```
package bank.management.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;

public class BalanceEnquriy extends JFrame implements ActionListener {
```

```
    String pin;
    JLabel label2;
    JButton b1;
    BalanceEnquriy(String pin){
        this.pin =pin;

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/atm2.png"));
        Image i2 = i1.getImage().getScaledInstance(1550,830,Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel l3 = new JLabel(i3);
        l3.setBounds(0,0,1550,830);
        add(l3);

        JLabel label1 = new JLabel("Your Current Balance is Rs ");
        label1.setForeground(Color.WHITE);
        label1.setFont(new Font("System", Font.BOLD, 16));
        label1.setBounds(430,180,700,35);
        l3.add(label1);

        label2 = new JLabel();
        label2.setForeground(Color.WHITE);
        label2.setFont(new Font("System", Font.BOLD, 16));
        label2.setBounds(430,220,400,35);
        l3.add(label2);

        b1 = new JButton("Back");
        b1.setBounds(700,406,150,35);
        b1.setBackground(new Color(65,125,128));
        b1.setForeground(Color.WHITE);
        b1.addActionListener(this);
        l3.add(b1);

        int balance =0;
        try{
            Connn c = new Connn();
            ResultSet resultSet = c.statement.executeQuery("Select * from bank where pin =
'"+pin+"'");
            while (resultSet.next()){
                if (resultSet.getString("type").equals("Deposit")){
                    balance += Integer.parseInt(resultSet.getString("amount"));
                }else {
                    balance -= Integer.parseInt(resultSet.getString("amount"));
                }
            }
        }catch (Exception e){
```

```
                e.printStackTrace();
            }

            label2.setText(""+balance);

            setLayout(null);
            setSize(1550,1080);
            setLocation(0,0);
            setVisible(true);
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new main_Class(pin);
        }

        public static void main(String[] args) {
            new BalanceEnquriy("");
        }
    }
```

DEPOSIT

```
package bank.management.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Date;

public class Deposit extends JFrame implements ActionListener {
  String pin;
  TextField textField;

  JButton b1, b2;
   Deposit(String pin){
      this.pin = pin;

      ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/atm2.png"));
      Image i2 = i1.getImage().getScaledInstance(1550,830,Image.SCALE_DEFAULT);
      ImageIcon i3 = new ImageIcon(i2);
      JLabel l3 = new JLabel(i3);
      l3.setBounds(0,0,1550,830);
      add(l3);
```

```
    JLabel label1 = new JLabel("ENETR AMOUNT YOU WANT TO DEPOSIT");
    label1.setForeground(Color.WHITE);
    label1.setFont(new Font("System", Font.BOLD, 16));
    label1.setBounds(460,180,400,35);
    l3.add(label1);

    textField = new TextField();
    textField.setBackground(new Color(65,125,128));
    textField.setForeground(Color.WHITE);
    textField.setBounds(460,230,320,25);
    textField.setFont(new Font("Raleway", Font.BOLD,22));
    l3.add(textField);

    b1 = new JButton("DEPOSIT");
    b1.setBounds(700,362,150,35);
    b1.setBackground(new Color(65,125,128));
    b1.setForeground(Color.WHITE);
    b1.addActionListener(this);
    l3.add(b1);

    b2 = new JButton("BACK");
    b2.setBounds(700,406,150,35);
    b2.setBackground(new Color(65,125,128));
    b2.setForeground(Color.WHITE);
    b2.addActionListener(this);
    l3.add(b2);




    setLayout(null);
    setSize(1550,1080);
    setLocation(0,0);
    setVisible(true);

}

@Override
public void actionPerformed(ActionEvent e) {
    try {
        String amount = textField.getText();
        Date date = new Date();
        if (e.getSource()==b1){
            if (textField.getText().equals("")){
```

```
                JOptionPane.showMessageDialog(null,"Please enter the Amount you want to Deposit");
            }else {
                Connn c = new Connn();
                c.statement.executeUpdate("insert into bank values('"+pin+"', '"+date+"','Deposit',
'"+amount+"')");
                JOptionPane.showMessageDialog(null,"Rs. "+amount+" Deposited Successfully");
                setVisible(false);
                new main_Class(pin);
            }
        }else if (e.getSource()==b2){
            setVisible(false);
            new main_Class(pin);
        }
      }catch (Exception E){
        E.printStackTrace();
      }

   }

   public static void main(String[] args) {
      new Deposit("");
   }
}
```

WITHDRAWL

```
package bank.management.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;
import java.util.Date;

public class Withdrawl extends JFrame implements ActionListener {

   String pin;
   TextField textField;

   JButton b1, b2;
   Withdrawl(String pin){
      this.pin=pin;
      ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/atm2.png"));
      Image i2 = i1.getImage().getScaledInstance(1550,830,Image.SCALE_DEFAULT);
      ImageIcon i3 = new ImageIcon(i2);
      JLabel l3 = new JLabel(i3);
      l3.setBounds(0,0,1550,830);
      add(l3);
```

```
    JLabel label1 = new JLabel("MAXIMUM WITHDRAWAL IS RS.10,000");
    label1.setForeground(Color.WHITE);
    label1.setFont(new Font("System", Font.BOLD, 16));
    label1.setBounds(460,180,700,35);
    l3.add(label1);

    JLabel label2 = new JLabel("PLEASE ENTER YOUR AMOUNT");
    label2.setForeground(Color.WHITE);
    label2.setFont(new Font("System", Font.BOLD, 16));
    label2.setBounds(460,220,400,35);
    l3.add(label2);


    textField = new TextField();
    textField.setBackground(new Color(65,125,128));
    textField.setForeground(Color.WHITE);
    textField.setBounds(460,260,320,25);
    textField.setFont(new Font("Raleway", Font.BOLD,22));
    l3.add(textField);

    b1 = new JButton("WITHDRAW");
    b1.setBounds(700,362,150,35);
    b1.setBackground(new Color(65,125,128));
    b1.setForeground(Color.WHITE);
    b1.addActionListener(this);
    l3.add(b1);

    b2 = new JButton("BACK");
    b2.setBounds(700,406,150,35);
    b2.setBackground(new Color(65,125,128));
    b2.setForeground(Color.WHITE);
    b2.addActionListener(this);
    l3.add(b2);

    setLayout(null);
    setSize(1550,1080);
    setLocation(0,0);
    setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==b1) {
        try {
            String amount = textField.getText();
            Date date = new Date();
            if (textField.getText().equals("")) {
                JOptionPane.showMessageDialog(null, "Please enter the Amount you want to withdraw");
            } else {
                Connn c = new Connn();
                ResultSet resultSet = c.statement.executeQuery("select * from bank where pin = '" + pin + "'");
                int balance = 0;
                while (resultSet.next()) {
                    if (resultSet.getString("type").equals("Deposit")) {
                        balance += Integer.parseInt(resultSet.getString("amount"));
                    } else {
                        balance -= Integer.parseInt(resultSet.getString("amount"));
```

```
                }
            }
            if (balance < Integer.parseInt(amount)) {
                JOptionPane.showMessageDialog(null, "Insuffient Balance");
                return;
            }

            c.statement.executeUpdate("insert into bank values('" + pin + "', '" + date + "', 'Withdrawl', '" + amount +
"' )");
            JOptionPane.showMessageDialog(null, "Rs. " + amount + " Debited Successfully");
            setVisible(false);
            new main_Class(pin);

          }
        } catch (Exception E) {

        }
      } else if (e.getSource()==b2) {
        setVisible(false);

        new main_Class(pin);
      }
    }

    public static void main(String[] args) {
      new Withdrawl("");
    }
}
```

CURRENCY EXCHANGE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;

public class CurrencyExchange extends JFrame implements ActionListener {

    private JComboBox<String> fromCurrency;
    private JComboBox<String> toCurrency;
    private JTextField amountField;
    private JLabel resultLabel;
    private JButton convertButton;

    // Predefined exchange rates
    private HashMap<String, Double> exchangeRates;

    public CurrencyExchange() {
        // Initialize the frame
        setTitle("Currency Exchange");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
```

```
    // Initialize exchange rates (as per example rates)
    initializeExchangeRates();

    // Currency options
    String[] currencies = {"USD", "EUR", "INR", "JPY", "GBP"};

    // UI components
    fromCurrency = new JComboBox<>(currencies);
    toCurrency = new JComboBox<>(currencies);
    amountField = new JTextField(10);
    resultLabel = new JLabel("Converted Amount: ");
    convertButton = new JButton("Convert");

    // Set action listener for the button
    convertButton.addActionListener(this);

    // Layout and add components
    setLayout(new GridLayout(5, 2, 10, 10));

    add(new JLabel("From Currency: "));
    add(fromCurrency);

    add(new JLabel("To Currency: "));
    add(toCurrency);

    add(new JLabel("Amount: "));
    add(amountField);

    add(new JLabel(""));
    add(convertButton);

    add(resultLabel);

    // Set frame visibility
    setVisible(true);
}

private void initializeExchangeRates() {
    exchangeRates = new HashMap<>();
    // Base currency is USD, you can add or update more exchange rates here
    exchangeRates.put("USD", 1.0);
    exchangeRates.put("EUR", 0.91);
    exchangeRates.put("INR", 82.5);
    exchangeRates.put("JPY", 141.5);
    exchangeRates.put("GBP", 0.79);
}

@Override
public void actionPerformed(ActionEvent e) {
    try {
```

```
            // Get user input
            String from = (String) fromCurrency.getSelectedItem();
            String to = (String) toCurrency.getSelectedItem();
            double amount = Double.parseDouble(amountField.getText());

            // Get exchange rates
            double fromRate = exchangeRates.get(from);
            double toRate = exchangeRates.get(to);

            // Calculate the converted amount
            double convertedAmount = (amount / fromRate) * toRate;

            // Display the result
            resultLabel.setText(String.format("Converted Amount: %.2f %s", convertedAmount, to));
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Please enter a valid number", "Invalid Input",
JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void main(String[] args) {
        new CurrencyExchange();
    }
}
```

UPI TRANSFER

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class UPIPayment extends JFrame implements ActionListener {

    private JTextField upiIdField;
    private JTextField amountField;
    private JPasswordField pinField;
    private JButton transferButton;
    private JLabel statusLabel;

    public UPIPayment() {
        // Frame settings
        setTitle("UPI Transfer");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // UPI ID
        JLabel upiIdLabel = new JLabel("UPI ID:");
        upiIdField = new JTextField(20);
```

```
    // Amount
    JLabel amountLabel = new JLabel("Amount:");
    amountField = new JTextField(10);

    // PIN
    JLabel pinLabel = new JLabel("Enter PIN:");
    pinField = new JPasswordField(4);

    // Transfer button
    transferButton = new JButton("Transfer");
    transferButton.addActionListener(this);

    // Status label
    statusLabel = new JLabel(" ");
    statusLabel.setForeground(Color.RED);

    // Layout setup
    setLayout(new GridLayout(5, 2, 10, 10));
    add(upiIdLabel);
    add(upiIdField);
    add(amountLabel);
    add(amountField);
    add(pinLabel);
    add(pinField);
    add(new JLabel("")); // Spacer
    add(transferButton);
    add(statusLabel);

    // Set frame visibility
    setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    String upiId = upiIdField.getText().trim();
    String amountText = amountField.getText().trim();
    String pin = new String(pinField.getPassword());

    // Basic validation
    if (upiId.isEmpty() || amountText.isEmpty() || pin.isEmpty()) {
        statusLabel.setText("All fields are required!");
        return;
    }

    try {
        double amount = Double.parseDouble(amountText);

        if (amount <= 0) {
            statusLabel.setText("Amount must be positive!");
            return;
        }
```

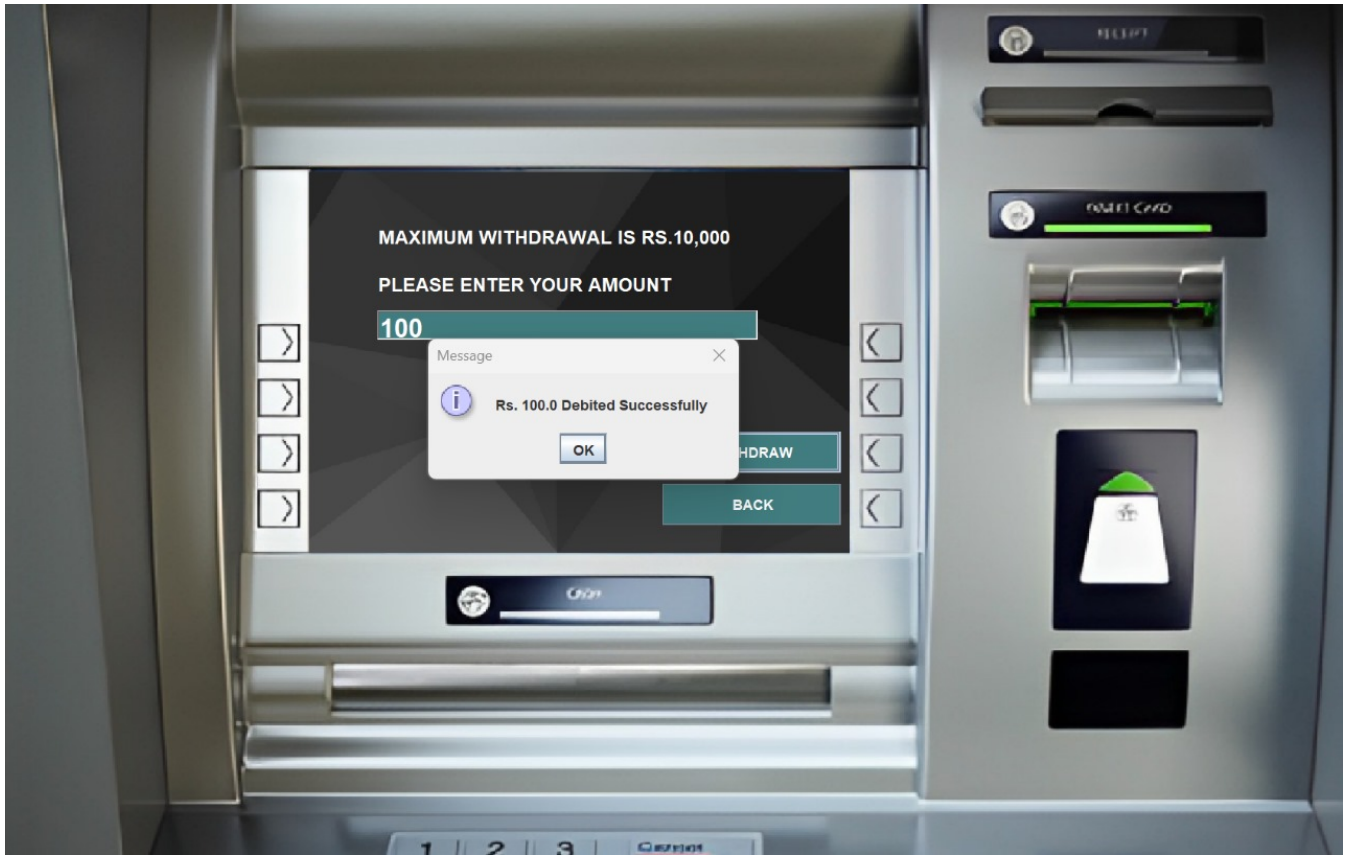
```
            if (pin.length() != 4) {
                statusLabel.setText("PIN must be 4 digits!");
                return;
            }

            // Simulate successful transaction
            statusLabel.setText("Transfer successful!");
            statusLabel.setForeground(new Color(0, 128, 0)); // Green color for success
            JOptionPane.showMessageDialog(this, "₹" + amount + " transferred to " + upiId);

        } catch (NumberFormatException ex) {
            statusLabel.setText("Please enter a valid amount!");
        }
    }

    public static void main(String[] args) {
        new UPIPayment();
    }
}
```

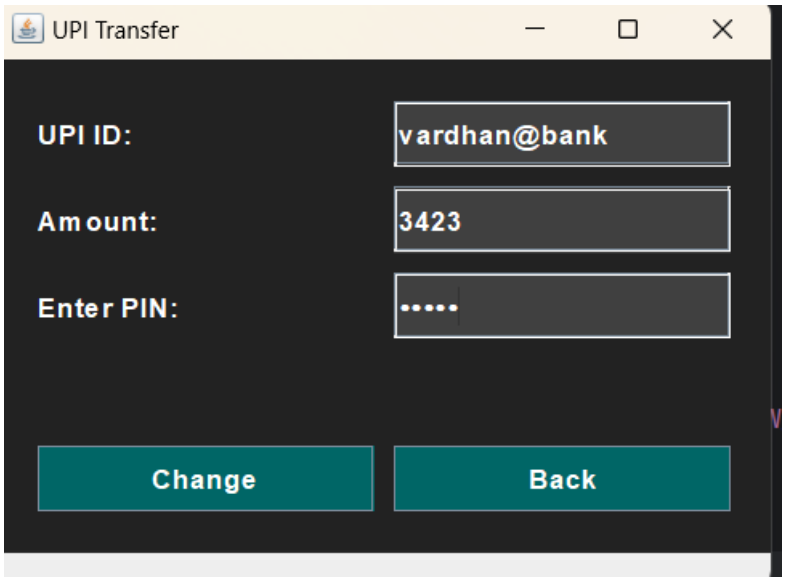
OUTPUT

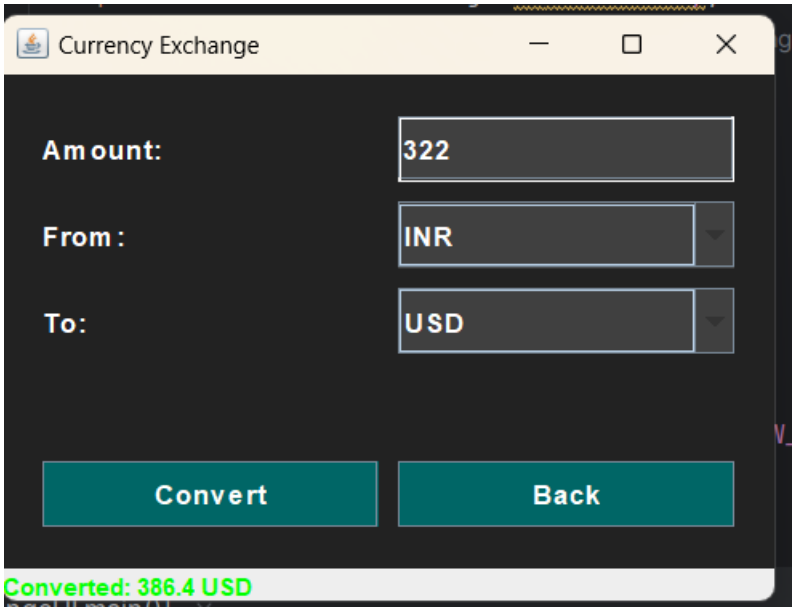

Please Select Your Transaction
DEPOSIT
CASH WITHDRAWL
FAST CASH
MINI STATEMENT
PIN CHANGE
BALANCE ENQUI...
EXIT


ENTER AMOUNT YOU WANT TO DEPOSIT
1000
Message
Rs. 1000 Deposited Successfully
OK
BACK


CHANGE YOUR PIN
New PIN:
Re-Enter New PIN:
CHANGE
BACK


MAXIMUM WITHDRAWAL IS RS.10,000
PLEASE ENTER YOUR AMOUNT
100
Message
Rs. 100.0 Debited Successfully
OK
BACK


UPI Transfer
UPI ID:
vardhan@bank
Amount:
3423
Enter PIN:
Change
Back


Currency Exchange
Amount:
322
From :
INR
To:
USD
Convert
Back
Converted: 386.4 USD

FUTURE SCOPE

The **future scope** of a Currency Exchange ATM Simulation system spans advancements in user experience, security, and integration with real-world financial services. As financial technology evolves, so do the expectations for automated banking systems like ATMs. Here's a look at some of the promising future directions:

1. Real-Time Exchange Rate Integration

Incorporating **live exchange rates** from financial markets into ATM systems could enhance the system's relevance and accuracy. By integrating APIs from forex data providers (such as Open Exchange Rates or ExchangeRate-API), the ATM could offer real-time currency conversions, providing users with up-to-the-minute exchange rates. This feature would be particularly beneficial for ATMs situated in high-traffic areas like airports and tourist destinations, where currency exchanges are frequent.

Future enhancement possibilities include:

- Predictive analytics to notify users of favorable exchange rates.
- Dynamic currency adjustments based on market conditions.
- Configurable alerts for customers when specific rate thresholds are reached.

2. Enhanced Security Features

Security is crucial for financial transactions. Future iterations of currency exchange ATMs could include **advanced security measures** like two-factor authentication (2FA), biometric verification (e.g., fingerprint or facial recognition), and end-to-end encryption. As cyber threats grow more sophisticated, adopting multi-layered security protocols will be essential for safeguarding users' financial information.

Potential advancements in this area might include:

- Integration with **biometric ID systems** to verify identities securely.
- Utilizing blockchain technology for secure, immutable transaction records.
- Real-time fraud detection and alerts for suspicious transaction patterns.

3. Mobile and Web-Based Extensions

Allowing customers to **access currency exchange services remotely** through mobile apps or web platforms could expand the system's usability. By linking ATMs with mobile and online banking platforms, users could view exchange rates, initiate currency exchanges, and manage their accounts remotely. This digital extension would provide flexibility and enable users to plan transactions before arriving at the ATM.

Future possibilities in this area:

- Notifications for users about current exchange rates or preferred withdrawal options.
- Mobile payment options integrated with digital wallets for faster transactions.

- Virtual currency exchanges, allowing users to convert currencies before arriving at a physical ATM.

4. AI-Driven Exchange Rate Predictions and Recommendations

Artificial Intelligence (AI) can transform currency exchange ATM systems by using **predictive models** to forecast exchange rates and make recommendations. AI algorithms could analyze historical data, geopolitical events, and economic trends to help users make informed decisions. By predicting rate fluctuations, the system could notify users when favorable rates are available, enhancing the customer experience.

Potential future applications:

- **Machine learning algorithms** to detect patterns in exchange rates.
- Customized exchange recommendations based on individual user behaviors and past transactions.
- Chatbot assistance for guiding users through currency exchange processes in multiple languages.

CONCLUSION

The future scope for currency exchange ATMs is expansive, driven by the rapid pace of technological advancements in financial services, security, and user expectations. Integrating real-time data, advanced security protocols, AI-driven features, blockchain technology, and enhanced UX can transform ATMs into versatile financial hubs. By focusing on user-centered innovation and emerging technologies, future ATMs with currency exchange functionalities could become indispensable tools for global finance, improving accessibility, convenience, and value for users worldwide.

THANK YOU