ONLINE BANKING MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

Submitted by

SACHIN.N 230701274

RUKESH KUMARAN . M 230701271

In partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project “ONLINE BANKING MANAGEMENT SYSTEM” is the

bonafide work of “SACHIN.N,RUKESH KUMARAN.M”who carried out the project work under

my supervision.

SIGNATURE

Mr. G SARAVANA GOKUL

ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,

Rajalakshmi Engineering College

Chennai-602105

SIGNATURE

MRS.JANANEE

ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,

Rajalakshmi Engineering College

Chennai-602105

This mini project report is submitted for the viva voce examination to be held on _____

TABLE OF CONTENTS

Online Banking System - Project Documentation

Introduction

The Online Banking System is a web-based application that enables users to perform banking transactions online. It allows customers to manage their bank accounts, view balances, deposit funds, and make withdrawals securely. The system is built using **Java**, with **Swing** for the frontend and **MySQL** as the backend database.

Features

1. User Authentication:

- Login system for users to securely access their accounts.
- Password protection with encryption for secure login.

2. Account Management:

- View account balance in real-time.
- Perform transactions such as deposits and withdrawals.
- Transaction history tracking.

3. Transaction Management:

- Users can deposit and withdraw funds.
- Transaction logs are stored for every deposit and withdrawal made.

4. Database Integration:

- **MySQL** database used for storing customer information, account details, and transaction history.

- **JDBC** used for database connection and CRUD operations.

Technologies Used:

- **Java**: Core programming language for the application.
- **Swing**: GUI framework for building the user interface.
- **MySQL**: Backend database for storing user data and transaction records.
- **JDBC**: Java Database Connectivity for database interaction.

System Architecture:

The system follows a **Client-Server architecture**:

- **Frontend**: Java Swing GUI provides a user-friendly interface for interacting with the application.
- **Backend**: MySQL database stores user information and transaction records.
- **Communication**: Java JDBC is used to establish communication between the frontend and backend.

Database Schema:

The following tables are used in the database:
Customers

Stores customer information.

Column	Type	Description
id	INT	Unique identifier (Primary Key)
name	VARCHAR(100)	Customer's name
account_number	VARCHAR(20)	Unique account number (Unique Key)
password	VARCHAR(50)	Customer's login password
balance	DOUBLE	Customer's account balance

transactions

Stores transaction history.

Column	Type	Description
id	INT	Unique identifier (Primary Key)
account_number	VARCHAR(20)	The associated account number
transaction_type	VARCHAR(20)	Type of transaction (Deposit/Withdraw)
amount	DOUBLE	Amount transacted
date	TIMESTAMP	Transaction timestamp

Setup Instructions:

Prerequisites

- **MySQL** database installed and running.
- **Java** Development Kit (JDK) installed.
- **JDBC** driver for MySQL added to the project.

Database Setup:

1. Create a database named bank_management.
2. Create the customers and transactions tables as described in the database schema.
3. Insert initial customer data into the customers table.

Running the Application:

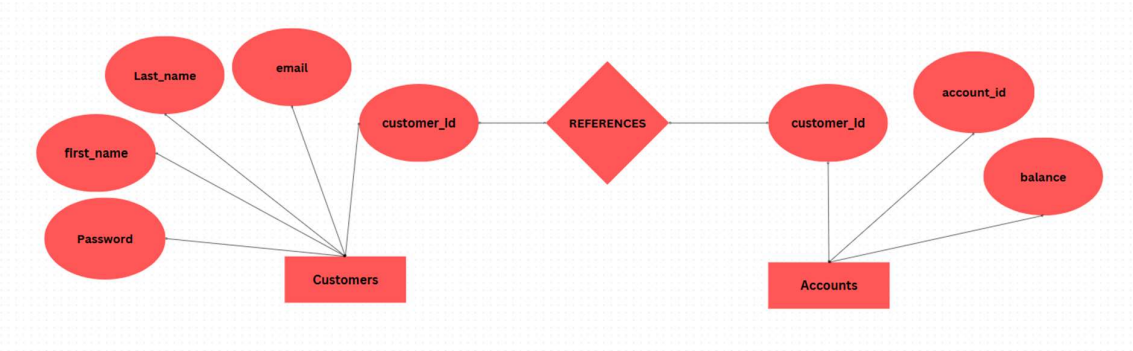
1. Download and install the **MySQL JDBC driver** and add it to your classpath.
2. Compile the Java files and run the BankManagementSystem.java class.
3. The application will launch the login screen, where you can log in using the default credentials or your own test account.

Objectives:

The primary objectives of the Online Bank Management System are:

- To provide a secure platform for managing bank accounts online.
- To enable users to perform banking transactions such as account management, money transfer, transaction history, and balance inquiry.
- To implement user-friendly interfaces for customers to interact with.
- To ensure high security and data protection, with encryption and multi-factor authentication.
- To enable banks to manage and monitor customer accounts and transactions effectively.
- The **primary objectives of an Online Banking Management System** are centered on providing customers with a secure, efficient, and convenient platform to manage their finances. The system must ensure robust security, offer a wide range of banking services (like transfers, bill payments, and loans), and adhere to financial regulations while providing a smooth and user-friendly experience.

ER DIAGRAM:



MySQL Database:

```
CREATE DATABASE banking_system;

USE banking_system;

CREATE TABLE Customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password VARCHAR(255)
);

CREATE TABLE Accounts (
    account_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    balance DECIMAL(15, 2) DEFAULT 0.00,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
```

```
);
```

DatabaseConnection.java:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {

    public static Connection getConnection() throws SQLException {
        try {
            // Load MySQL JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");
            // Database connection details
            String url = "jdbc:mysql://localhost:3306/banking_system";  // Update with
your database details
            String username = "root";  // Update with your MySQL username
            String password = "";  // Update with your MySQL password

            return DriverManager.getConnection(url, username, password);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            throw new SQLException("Database connection failed.");
        }
    }
}
```

UserAuth.java:

```
import java.sql.*;

public class UserAuth {

    public static boolean authenticate(String email, String password) {
        String query = "SELECT * FROM users WHERE email = ? AND password = ?";
        try (Connection con = DatabaseConnection.getConnection();
             PreparedStatement ps = con.prepareStatement(query)) {
            ps.setString(1, email);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                return true;  // User exists and credentials are valid
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;  // Authentication failed
    }
}
```

Banking Dashboard:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.ArrayList;

public class BankingDashboard extends JFrame {

    private JPanel contentPanel;
    private JLabel lblBalance;
    private JTextArea transactionHistoryArea;
    private JButton btnTransfer;
    private String currentUserEmail;

    public BankingDashboard(String email) {
        currentUserEmail = email;
        setTitle("Banking Dashboard");
        setLayout(new BorderLayout());

        contentPanel = new JPanel();
        contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.Y_AXIS));

        lblBalance = new JLabel("Account Balance: $0.00");
        lblBalance.setFont(new Font("Arial", Font.BOLD, 16));
        contentPanel.add(lblBalance);

        transactionHistoryArea = new JTextArea(10, 30);
        transactionHistoryArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(transactionHistoryArea);
        contentPanel.add(scrollPane);

        btnTransfer = new JButton("Transfer Money");
        btnTransfer.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                showTransferDialog();
            }
        });
        contentPanel.add(btnTransfer);

        add(contentPanel, BorderLayout.CENTER);
```

```
        // Load account balance and transaction history
        loadAccountDetails();
    }

    private void loadAccountDetails() {
        try (Connection con = DatabaseConnection.getConnection()) {
            // Fetch account balance
            String balanceQuery = "SELECT balance FROM users WHERE email = ?";
            try (PreparedStatement ps = con.prepareStatement(balanceQuery)) {
                ps.setString(1, currentUserEmail);
                ResultSet rs = ps.executeQuery();
                if (rs.next()) {
                    double balance = rs.getDouble("balance");
                    lblBalance.setText("Account Balance: $" + balance);
                }
            }

            // Fetch transaction history
            String transactionQuery = "SELECT * FROM transactions WHERE user_id =
(SELECT id FROM users WHERE email = ?)";
            try (PreparedStatement ps = con.prepareStatement(transactionQuery)) {
                ps.setString(1, currentUserEmail);
                ResultSet rs = ps.executeQuery();
                StringBuilder history = new StringBuilder();
                while (rs.next()) {
                    String type = rs.getString("type");
                    double amount = rs.getDouble("amount");
                    String description = rs.getString("description");
                    history.append(type).append(" of $").append(amount).append(":
").append(description).append("\n");
                }
                transactionHistoryArea.setText(history.toString());
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void showTransferDialog() {
        JTextField recipientField = new JTextField(20);
        JTextField amountField = new JTextField(20);
```

```
        JTextField descriptionField = new JTextField(20);

        JPanel transferPanel = new JPanel();
        transferPanel.setLayout(new GridLayout(3, 2));
        transferPanel.add(new JLabel("Recipient Email:"));
        transferPanel.add(recipientField);
        transferPanel.add(new JLabel("Amount:"));
        transferPanel.add(amountField);
        transferPanel.add(new JLabel("Description:"));
        transferPanel.add(descriptionField);

        int option = JOptionPane.showConfirmDialog(this, transferPanel, "Enter
Transfer Details", JOptionPane.OK_CANCEL_OPTION);

        if (option == JOptionPane.OK_OPTION) {
            String recipientEmail = recipientField.getText();
            double amount = Double.parseDouble(amountField.getText());
            String description = descriptionField.getText();
            processTransfer(recipientEmail, amount, description);
        }
    }

    private void processTransfer(String recipientEmail, double amount, String
description) {
        try (Connection con = DatabaseConnection.getConnection()) {
            // Fetch sender ID
            String senderQuery = "SELECT id, balance FROM users WHERE email = ?";
            PreparedStatement ps = con.prepareStatement(senderQuery);
            ps.setString(1, currentUserEmail);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                int senderId = rs.getInt("id");
                double senderBalance = rs.getDouble("balance");

                if (senderBalance >= amount) {
                    // Deduct amount from sender's account
                    String updateSenderQuery = "UPDATE users SET balance = balance -
? WHERE id = ?";
                    ps = con.prepareStatement(updateSenderQuery);
                    ps.setDouble(1, amount);
                    ps.setInt(2, senderId);
                    ps.executeUpdate();
```

```
                // Add transaction to the history table
                String transactionQuery = "INSERT INTO transactions (user_id, amount,
type, description) VALUES (?, ?, 'debit', ?)";
                ps = con.prepareStatement(transactionQuery);
                ps.setInt(1, senderId);
                ps.setDouble(2, amount);
                ps.setString(3, description);
                ps.executeUpdate();
                // Add amount to recipient's account
                String recipientQuery = "SELECT id FROM users WHERE email = ?";
                ps = con.prepareStatement(recipientQuery);
                ps.setString(1, recipientEmail);
                rs = ps.executeQuery();
                if (rs.next()) {
                    int recipientId = rs.getInt("id");
                    String updateRecipientQuery = "UPDATE users SET balance =
balance + ? WHERE id = ?";
                    ps = con.prepareStatement(updateRecipientQuery);
                    ps.setDouble(1, amount);
                    ps.setInt(2, recipientId);
                    ps.executeUpdate();

                    // Add recipient transaction history
                    String recipientTransactionQuery = "INSERT INTO transactions
(user_id, amount, type, description) VALUES (?, ?, 'credit', ?)";
                    ps = con.prepareStatement(recipientTransactionQuery);
                    ps.setInt(1, recipientId);
                    ps.setDouble(2, amount);
                    ps.setString(3, description);
                    ps.executeUpdate();

                    JOptionPane.showMessageDialog(this, "Transfer successful!");
                    loadAccountDetails();  // Reload account balance and transaction
history
                }
            } else {
                JOptionPane.showMessageDialog(this, "Insufficient balance.");
            }
        }

    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error processing transfer.");
```

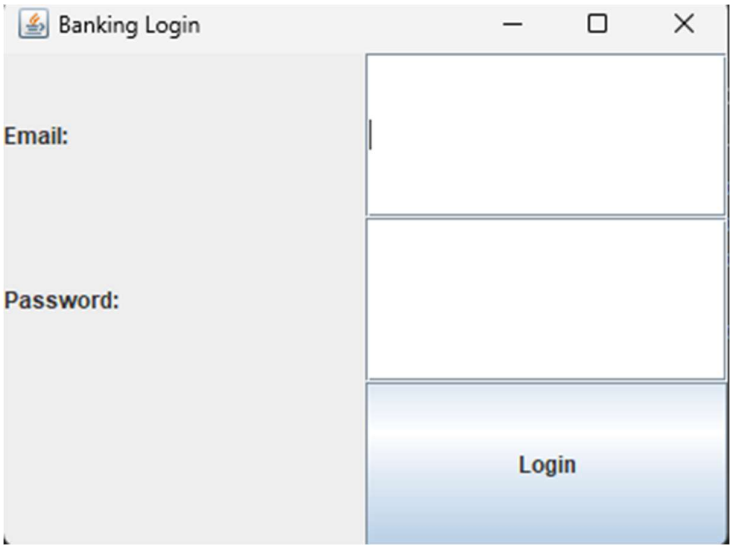
```
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            String email = JOptionPane.showInputDialog("Enter your email:");
            new BankingDashboard(email).setVisible(true);
        });
    }
}
```

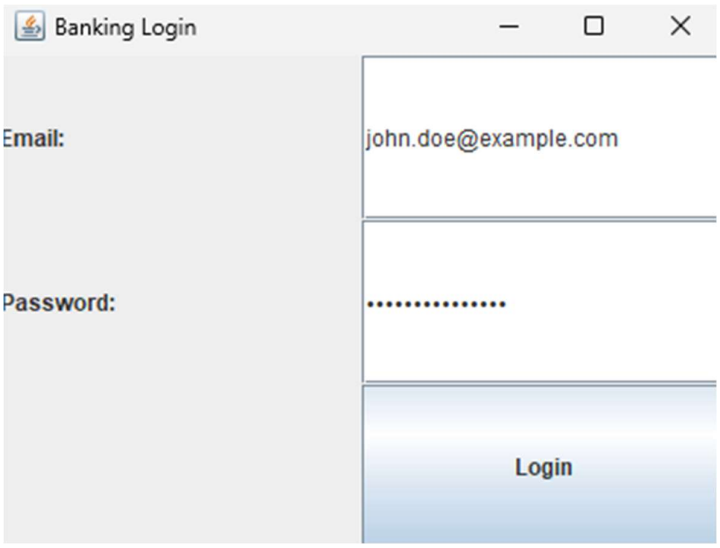
BankingApp:

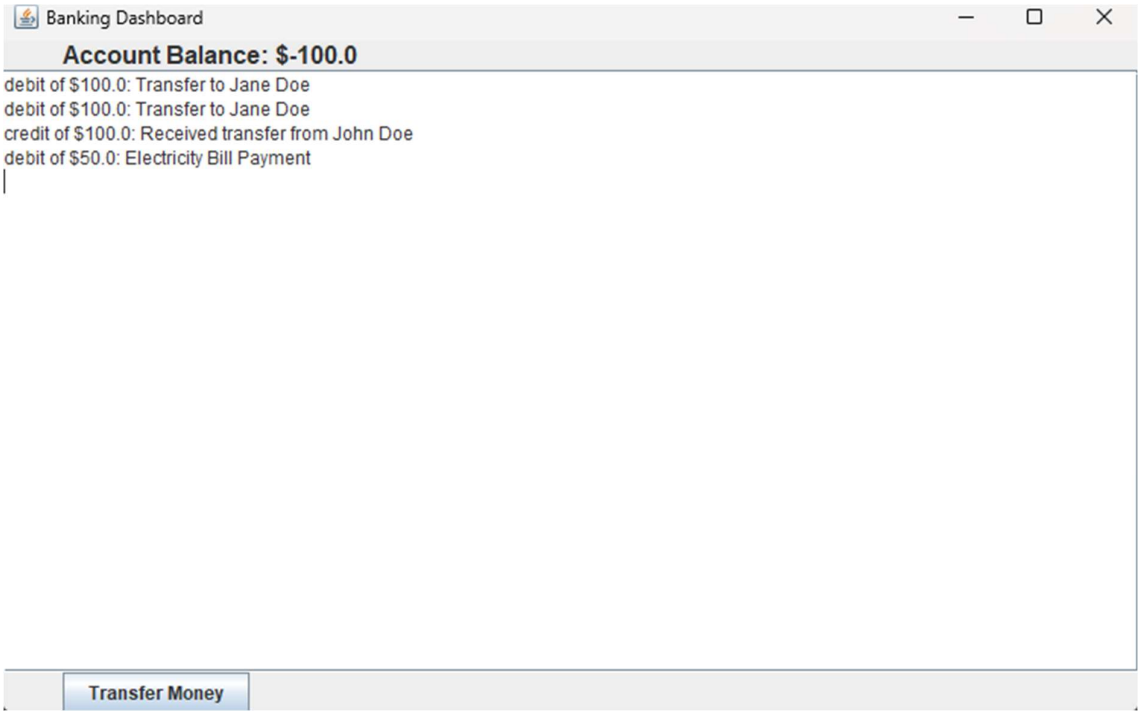
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class BankingApp {

    public static void main(String[] args) {
        JFrame loginFrame = new JFrame("Banking Login");
        loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        loginFrame.setSize(400, 300);
        loginFrame.setLayout(new GridLayout(3, 2));

        JLabel lblEmail = new JLabel("Email:");
        JLabel lblPassword = new JLabel("Password:");
        JTextField txtEmail = new JTextField();
        JPasswordField txtPassword = new JPasswordField();
        JButton btnLogin = new JButton("Login");

        loginFrame.add(lblEmail);
        loginFrame.add(txtEmail);
        loginFrame.add(lblPassword);
        loginFrame.add(txtPassword);
        loginFrame.add(new JLabel());
        loginFrame.add(btnLogin);
```

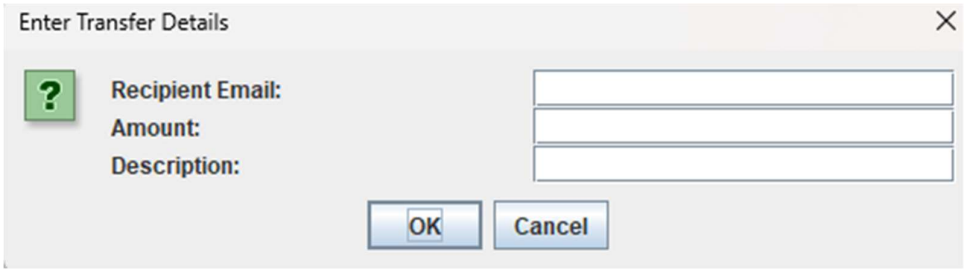
```
        btnLogin.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String email = txtEmail.getText();
                String password = new String(txtPassword.getPassword());

                if (UserAuth.authenticate(email, password)) {
                    loginFrame.setVisible(false);  // Hide login window
                    new BankingDashboard(email).setVisible(true);  // Show dashboard
                } else {
                    JOptionPane.showMessageDialog(loginFrame, "Invalid login
credentials.");
                }
            }
        });

        loginFrame.setVisible(true);
    }
}
```

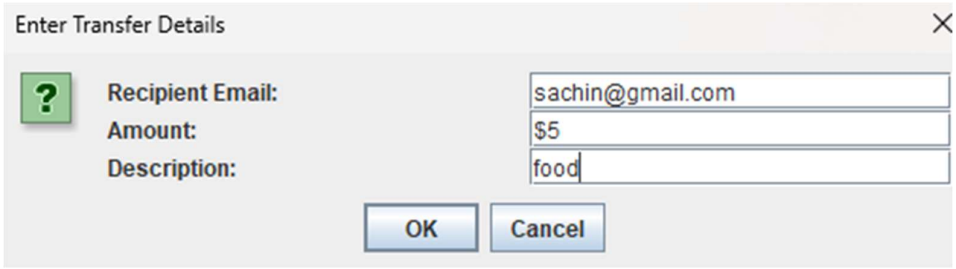
Snapshots:

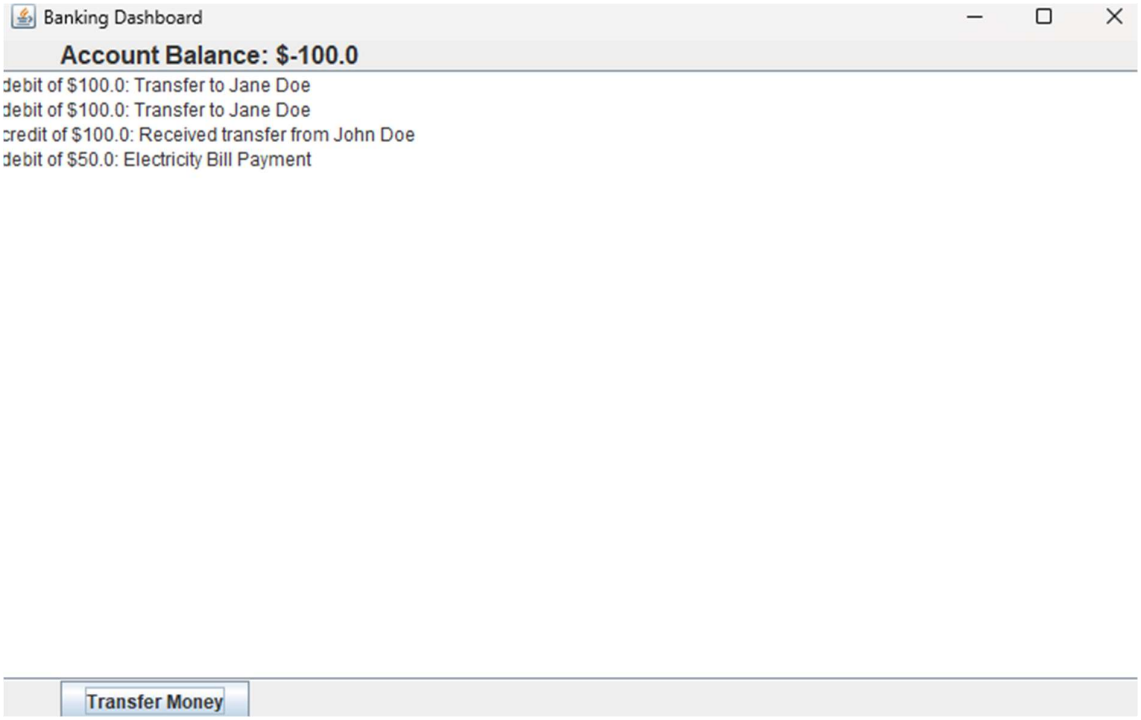

Banking Login
Email:
Password:
Login


Banking Login
Email:
john.doe@example.com
Password:
Login


Banking Dashboard
Account Balance: $-100.0
debit of $100.0: Transfer to Jane Doe
debit of $100.0: Transfer to Jane Doe
credit of $100.0: Received transfer from John Doe
debit of $50.0: Electricity Bill Payment
Transfer Money


Enter Transfer Details
Recipient Email:
Amount:
Description:
OK
Cancel


Enter Transfer Details
Recipient Email:
sachin@gmail.com
Amount:
$5
Description:
food
OK
Cancel


Banking Dashboard
Account Balance: $-100.0
debit of $100.0: Transfer to Jane Doe
debit of $100.0: Transfer to Jane Doe
credit of $100.0: Received transfer from John Doe
debit of $50.0: Electricity Bill Payment
Transfer Money

Conclusion:

The **Online Bank Management System** offers a reliable, secure, and user-friendly platform for customers to manage their banking needs online. The system is designed with high security in mind, incorporating features like multi-factor authentication, data encryption, and a secure transaction process. By leveraging modern technologies, the system aims to provide a seamless experience for both customers and administrators while maintaining scalability and performance.

References:

1. https://www.javatpoint.com/java-tutorial/
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL|Codecademy