BANKING MANAGEMENT SYSTEM

# A MINI PROJECT REPORT

**Submitted by**

# PRITHIVIRAAJ C 220701508

**HARISH A 220701105**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM

CHENNAI-602105

2023 - 24

BONAFIDE CERTIFICATE

Certified that this project report “**BANKING MANAGEMENT SYSTEM**” is the bonafide work of **“PRITHIVIRAAJ C (230701508), HARISH A (230701105)”**

who carried out the project work under my supervision.

# Submitted for the Practical Examination held on

**SIGNATURE SIGNATURE**

**Mrs.Divya.M Mr.Raghu.G**

**Assistant Professor**, **Assistant Professor , Computer Science and Engineering, Computer Science and Engineering,**

**Rajalakshmi Engineerin**g **College Rajalakshmi Engineering College, Thandalam, Chennai - 602 105 Thandalam, Chennai - 602 105**

# INTERNAL EXAMINER EXTERNAL EXAMIER

**ABSTRACT**:

The Bank Management System (BMS) is a sophisticated software application designed to streamline and automate various functions of banking institutions. It serves as an integrated platform that manages day-to-day banking activities, from handling customer accounts and processing transactions to managing loans, generating reports, and ensuring security. The need for such a system has evolved over time due to the increased complexity of banking operations, the rise of digital banking, and the demand for faster, more efficient services.

In traditional banking, most processes were manual, requiring paper-based record-keeping, human intervention, and long processing times. This not only led to inefficiencies but also increased the likelihood of errors, fraud, and delays in customer service. As banking services became more complex, especially with the growing number of transactions, accounts, and financial products, the reliance on paper-based systems was no longer sustainable. In response, the Bank Management System emerged as a comprehensive solution to automate these processes, reducing errors and enhancing the overall customer experience.

**TABLE OF CONTENTS**

**Chapter 1**

# INTRODUCTION

* 1. INTRODUCTION 6
  2. OBJECTIVES 7
  3. MODULES 7

# Chapter 2

1. **SURVEY OF TECHNOLOGIES**
   1. SOFTWARE DESCRIPTION 9
   2. LANGUAGES 9
   3. JAVA ------------------------------------------------------------------10
   4. MYSQL----------------------------------------------------------------10

# Chapter 3

1. **REQUIREMENTS AND ANALYSIS**
   1. [REQUIREMENT SPECIFICATION 11](#_TOC_250002)
      1. [FUNCTIONAL REQUIREMENTS 11](#_TOC_250001)
      2. NON FUNCTIONAL REQUIREMENTS 12
   2. HARDWARE AND SOFTWARE REQUIREMENTS 13
   3. [ARCHITECTURE DIAGRAM 14](#_TOC_250000)
   4. ER DIAGRAM 15
   5. NORMALIZATION 16

# Chapter 4

1. **PROGRAM CODE**
   1. PROGRAM CODE 18

# Chapter 5

1. **RESULTS AND DISCUSSION**
   1. RESULTS AND DISCUSSION 27

# Chapter 6

1. **CONCLUSION** 
   1. CONCLUSION 31

# Chapter 7

1. **REFERENCES**
   1. REFERENCES 32

# Chapter 1 INTRODUCTION

* 1. **INTRODUCTION**

In today’s digital age, building a robust and efficient **Banking Management System (BMS)** requires technologies that ensure scalability, security, and ease of development. Java, known for its platform independence and extensive libraries, combined with MySQL, a powerful and open-source relational database management system, is a popular choice for developing such systems.

**Why Use Java for a Banking Management System?**

Java is a widely used programming language in the financial industry due to its features like:

1. **Platform Independence**: Java applications can run on any platform with the Java Virtual Machine (JVM), making it versatile for various systems.
2. **Security**: Java provides built-in security features such as encryption, secure socket layers (SSL), and authentication mechanisms essential for banking systems.
3. **Object-Oriented Design**: Facilitates modular and scalable application development, allowing efficient management of complex banking operations.
4. **Extensive Libraries**: Frameworks like Spring and Hibernate simplify backend development, while tools like JavaFX and Swing enable interactive user interfaces.
5. **Performance**: Java supports multi-threading and efficient memory management, making it ideal for handling large-scale banking transactions.

**Why Use MySQL for Database Management?**

MySQL is an excellent choice for managing the backend data of a BMS due to:

1. **Relational Database Model**: Perfect for managing structured data like customer details, transactions, and account records.
2. **Scalability**: Handles small to large datasets, making it suitable for growing banking operations.
3. **Performance**: Optimized for read-heavy operations like account balance queries or transaction history retrieval.
4. **Data Integrity**: Supports constraints, triggers, and ACID compliance, ensuring accurate and consistent data storage.
5. **Integration with Java**: JDBC (Java Database Connectivity) allows seamless interaction between Java applications and MySQL databases.

**How Java and MySQL Work Together in a BMS**

1. **Backend Development**:
   * Java serves as the primary language for implementing the business logic of the system.
   * MySQL acts as the backend database where all data—such as customer profiles, account details, and transaction logs—is stored.
2. **Data Flow**:
   * Java interacts with MySQL using **JDBC API** for tasks like querying data, updating records, and maintaining database connections.
   * SQL queries are executed to fetch or modify data as per user actions in the application.
3. **User Interface**:
   * Java technologies like JavaFX or Swing can create user-friendly graphical interfaces for bank employees or customers.
   * Alternatively, Java Servlets and JSP can be used for web-based interfaces.
4. **Security**:
   * Secure database connections using prepared statements to prevent SQL injection attacks.
   * Encryption techniques can be applied to sensitive data like passwords and transactions.

**Benefits of Using Java and MySQL for a BMS**

1. **Cost-Effectiveness**: Both Java and MySQL are open-source, reducing development costs.
2. **Cross-Platform Compatibility**: Java applications with MySQL databases can be deployed on various operating systems like Windows, Linux, or macOS.
3. **Community Support**: A large developer community ensures ample resources, tutorials, and support for troubleshooting.

By leveraging Java and MySQL, developers can create a secure, scalable, and efficient Banking Management System capable of handling complex operations, meeting regulatory requirements, and delivering exceptional user experiences.

Top of Form

Bottom of Form

# OBJECTIVES

* To automate routine banking tasks such as account management, transactions, and loan processing to reduce manual intervention, minimize errors, and enhance operational efficiency.
* To enhance customer experience by providing fast, reliable, and accessible banking services.
* To ensure all transactions (deposits, withdrawals, transfers) are processed in real time with immediate updates to customer accounts.
* To safeguard sensitive customer information and financial data from unauthorized access and fraud.
* To provide a centralized platform that can manage all customer accounts, financial products (savings, checking, loans), and transactions.

# MODULES

**1. User Authentication and Authorization Module**

* **Purpose**: Ensures secure access to the system by authenticating users and assigning appropriate roles.
* **Features**:
  + User login and logout functionality.
  + Role-based access control (e.g., admin, manager, customer).
  + Password encryption and reset options.
  + Multi-factor authentication (optional).
* **Implementation**:
  + Use Java's Spring Security or custom logic for authentication.
  + Store user credentials securely in MySQL using hashing (e.g., bcrypt).

**2. Customer Management Module**

* **Purpose**: Manage customer profiles and personal details.
* **Features**:
  + Add, update, or delete customer information.
  + Customer KYC (Know Your Customer) verification.
  + Search and view customer details.
* **Implementation**:
  + Use Java Servlets or JavaFX for the user interface.
  + Maintain a **Customer** table in MySQL with fields like CustomerID, Name, Address, Phone, and Email.

**3. Account Management Module**

* **Purpose**: Handle account-related operations.
* **Features**:
  + Open and close bank accounts (savings, current, fixed deposits).
  + View account details and balance.
  + Manage account types and interest rates.
* **Implementation**:
  + Use MySQL to store account data in an **Accounts** table with fields like AccountNumber, CustomerID, AccountType, and Balance.

**4. Transaction Management Module**

* **Purpose**: Facilitate and record financial transactions.
* **Features**:
  + Deposit and withdrawal of funds.
  + Fund transfers between accounts.
  + View transaction history.
* **Implementation**:
  + Use Java to implement transaction logic.
  + Store transactions in a **Transactions** table with fields like TransactionID, AccountNumber, Type (credit/debit), Amount, and Timestamp.

**5. Loan Management Module**

* **Purpose**: Manage loan processing and repayments.
* **Features**:
  + Loan application and approval workflows.
  + Calculate and display repayment schedules.
  + Track loan status and balance.
* **Implementation**:
  + Use MySQL to store loan data in a **Loans** table with fields like LoanID, CustomerID, LoanAmount, InterestRate, RepaymentTerm, and OutstandingBalance.

**6. Reporting and Analytics Module**

* **Purpose**: Provide insights into banking operations.
* **Features**:
  + Generate account statements.
  + View customer and transaction reports.
  + Analytics for customer behavior and financial trends.
* **Implementation**:
  + Use SQL queries to aggregate and fetch data.
  + Display data in graphical form using Java libraries like **JFreeChart**.

**7. Security and Compliance Module**

* **Purpose**: Ensure secure and compliant banking operations.
* **Features**:
  + Audit logs for system activities.
  + Fraud detection mechanisms.
  + Compliance checks (e.g., AML, GDPR).
* **Implementation**:
  + Store logs and compliance data in a **Logs** or **Audit** table.
  + Implement fraud detection algorithms in Java.

**8. Admin Management Module**

* **Purpose**: Oversee and manage the overall system.
* **Features**:
  + Add, update, or remove bank employees and their roles.
  + Monitor system performance and usage.
  + Perform backups and database maintenance.
* **Implementation**:
  + Use a dedicated **Admin** table in MySQL.
  + Implement admin functionalities using Java's GUI frameworks or web-based interfaces.

**9. Online Banking Module (Optional for Advanced Systems)**

* **Purpose**: Provide customers with digital access to their accounts.
* **Features**:
  + Online login and account access.
  + View balances and transaction history.
  + Perform online fund transfers and bill payments.
* **Implementation**:
  + Use Java Servlets, JSP, or frameworks like **Spring Boot**.
  + Integrate with MySQL for backend data storage.

**10. Notification and Alerts Module**

* **Purpose**: Notify customers about account activities.
* **Features**:
  + SMS or email alerts for transactions.
  + Reminders for loan repayments or account renewals.
  + Notifications for suspicious activities.
* **Implementation**:
  + Use Java Mail API for email notifications.
  + Maintain a **Notifications** table in MySQL to track sent alerts.

**Database Design Overview**

* **Tables**:
  + Customers
  + Accounts
  + Transactions
  + Loans
  + Admins
  + Logs/Audit
  + Notifications

**Development Workflow**

1. Design the database schema in MySQL.
2. Develop backend logic using Java to handle each module.
3. Integrate Java with MySQL using JDBC or frameworks like Hibernate.
4. Create a user interface using JavaFX, Swing, or a web-based frontend.

These modules work together to create a comprehensive Banking Management System, ensuring efficient operations, enhanced user experience, and robust data management.

Top of Form

Bottom of Form

# Chapter 2 SURVEY OF TECHNOLOGIES

* 1. **SOFTWARE DESCRIPTION**

Banking Management System (BMS) is a software solution designed to automate and manage various banking operations, including customer account management, transaction processing, loan management, bill payments, and reporting. It provides secure, efficient services to customers through web or mobile platforms, allowing them to manage their accounts, perform transactions, and apply for loans. For bank administrators, the system offers features such as user management, real-time reporting, and compliance monitoring.

* 1. **LANGUAGES**

A Banking Management System uses Java for implementing backend logic, handling transactions, and business operations. MySQL is used to store and manage data, such as customer accounts, transactions, and loan details.

**2.2.2 SQL**

Role: SQL (Structured Query Language) is used for managing and manipulating the relational database that stores all data.

Usage:All information related to users and transactions in the system are stored and implemented as a relational schema using SQL.

Advantages:

• Database Connectivity: It offers robust support for database operations,

particularly with MySQL, facilitating efficient data management.

• Efficient Data Management: SQL allows ⁫for efficient querying, updating,

and management of large datasets.

Data Retrieval: SQL is used to query the database and retrieve information such as account details, transaction history, customer information, and loan statuses.

# Chapter 3 REQUIREMENTS AND ANALYSIS

# REQUIREMENT SPECIFICATION

# Functional Requirements

**User Management**

1. **User ManagementCustomer Registration and Account Creation:**

Customers should be able to register on the platform and create accounts (savings, current, etc.).

**Customer Login and Authentication:**Customers should be able to log in securely using their credentials (username, password).

Multi-factor authentication (MFA) should be supported for enhanced security.

**User Profile Management:**Customers should be able to update their personal information (contact details, address, etc.).

**Employee Login and Access:**Bank employees should have a login portal with different access levels (e.g., customer service, admin).

2. **Account ManagementView Account Information:**

Customers should be able to view their account details (balance, account type, account number, etc.).

**Create and Close Accounts:**Customers should be able to create or close different types of accounts (e.g., savings, checking).

**Account Details Update:**Customers should be able to update their account details (e.g., change address, phone number).

**Account Balance Inquiry:**Customers should be able to check their account balance.

**3. Transaction ManagementDeposit Funds:**

Customers should be able to deposit money into their account.

**Withdraw Funds:**Customers should be able to withdraw funds from their account via ATM, bank counter, or online transfer.

**Fund Transfers:**Customers should be able to transfer funds between their accounts or to external accounts (internal and external transfers).

**Transaction History:**Customers should be able to view a detailed transaction history (date, amount, type).

**Transaction Notifications:**Real-time notifications (SMS/email) should be sent for every transaction.

**4. Admin ManagementEmployee Role Management:**

Admins should be able to add, update, and delete bank employee accounts, assign roles, and set access permissions.

**Branch Management:**Admins should be able to manage different branches of the bank (e.g., opening/closing branches, assigning employees).

**System Monitoring:**Admins should be able to monitor system activities, transactions, and user activities for security and auditing purposes.

**Audit Trails:**The system should keep logs of all actions (user activity, system changes) for compliance and auditing.

**5. Security ManagementData Encryption:**

All sensitive data (account numbers, personal information) should be encrypted to protect user privacy.

**Authentication and Authorization:**Role-based access control (RBAC) should be implemented to ensure users (customers, bank employees) have appropriate access to system resources.

**Fraud Detection:**The system should have fraud detection algorithms to identify suspicious activities (e.g., multiple failed login attempts, unusual transactions).

**Transaction Security:**Secure protocols (e.g., SSL/TLS) should be used to protect transaction data during transfers.

**6. Reporting and AnalyticsAccount Statement Generation:**

Customers should be able to generate account statements over a specific period.

**Transaction Reports:**Bank employees and admins should be able to generate transaction reports for audit and analysis purposes.

**Loan Reports:**The system should provide detailed loan reports (e.g., loan status, outstanding balance).

**Financial Summary:**Admins and managers should have access to financial summaries, including balance sheets, profit/loss reports, and transaction trends.

**3.1.2 Non-Functional Requirements**

**1.Security:**

**Data Encryption:** All sensitive data, such as personal information and transaction details, must be encrypted using AES-256 encryption to ensure data privacy.

**Authentication:** The system must use multi-factor authentication (MFA) to ensure secure login and access control.

**Authorization:** Role-based access control (RBAC) must be implemented to ensure that users can only access the data and functions they are authorized for.

**Secure Transactions:** All transactions must be processed securely using SSL/TLS encryption to protect against eavesdropping and man-in-the-middle attacks.

**2. Performance Response Time:**

The system must respond to user requests (e.g., viewing account balance, transferring funds) within 2 seconds for regular operations.

**Throughput:** The system should be able to handle a high volume of transactions, supporting at least 1000 transactions per second without degradation in performance.

**Scalability:** The system should be able to scale to accommodate an increasing number of users and transactions, with minimal changes to architecture.

**3. AvailabilityUptime:** The system must be available 99.9% of the time to ensure continuous banking services, with minimal downtime for maintenance.

**Fault Tolerance:** The system must be able to continue operating, even in the event of hardware or software failures, by using redundant systems and failover mechanisms.

**4. UsabilityUser Interface:** The system should have an intuitive and easy-to-navigate interface that is user-friendly for customers and bank employees, with accessibility features for people with disabilities (e.g., WCAG 2.1 compliance).

**Multi-language Support:** The system should provide multi-language support for customers in different regions, such as English, Spanish, French, etc.Mobile Compatibility: The system must be fully functional and responsive on mobile devices, supporting both Android and iOS platforms.

# SOFTWARE REQUIREMENTS

# Software Requirements:

1. Operating System:

Windows, Linux, or macOS for development and deployment depending on the environment (server-side or client-side).

Server Operating System (e.g., Ubuntu, CentOS, Windows Server) for deployment of the backend system.

2. Database Management System (DBMS)MySQL:

A relational database management system (RDBMS) for storing customer information, account details, transactions, and other financial data.Alternatively, other RDBMS like PostgreSQL or Oracle can also be used depending on the needs.

3. Programming LanguagesJava:

The primary language for developing the backend logic of the Banking Management System.Frameworks like Spring Boot for creating REST APIs, handling transactions, and simplifying security and database integration.

SQL: For interacting with the database, writing queries for data retrieval, insertion, and updates (e.g., MySQL or PostgreSQL SQL queries).

4. Web FrameworksSpring Boot:

To quickly build and deploy Java-based web applications. It simplifies the integration with databases, security configurations, and deployment.Hibernate/JPA: Object-Relational Mapping (ORM) tools for database interaction, making it easier to persist and retrieve data without writing raw SQL.

5. Front-End TechnologiesHTML5, CSS3, JavaScript:

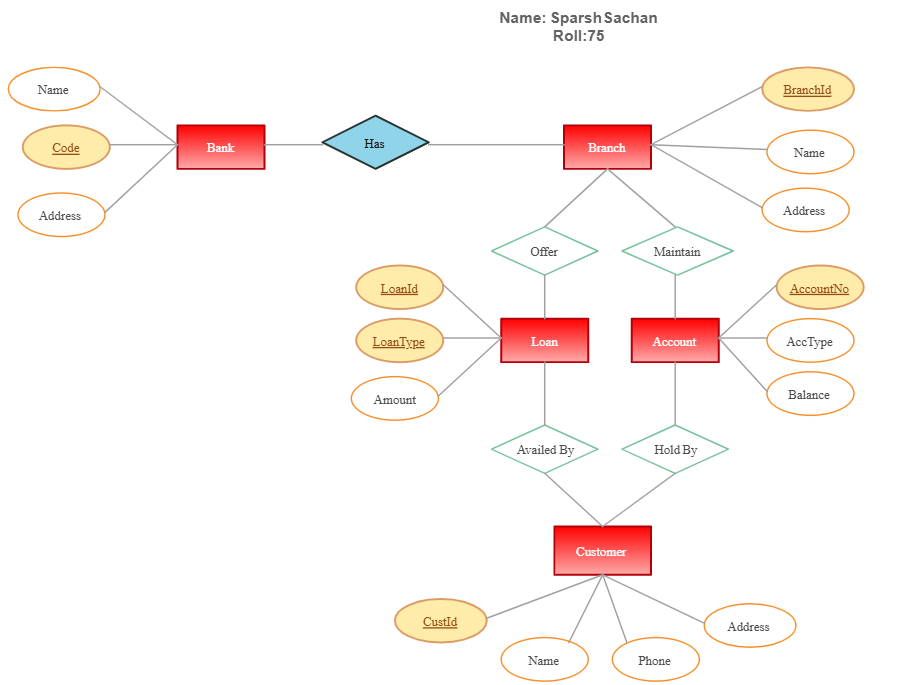
Core web technologies for creating the user interface (UI) for the Banking Management System, including web forms, account dashboards, and transaction views.

React.js or Angular: Modern JavaScript frameworks for creating dynamic and interactive user interfaces.

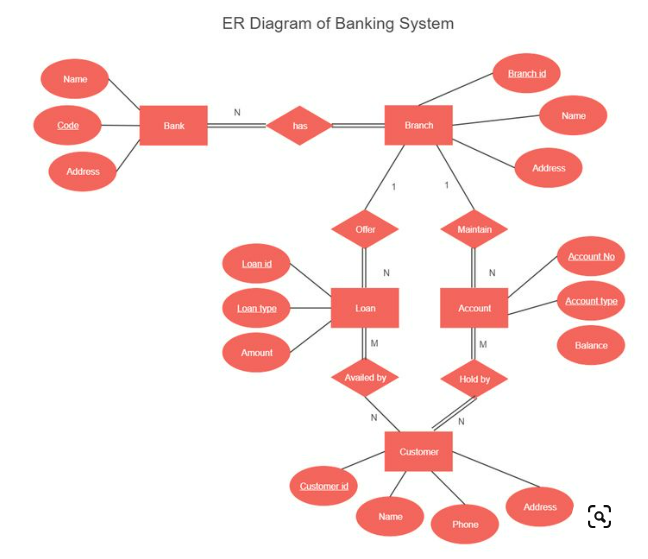
Bootstrap or Material UI: For responsive and mobile-friendly front-end design

# ARCHITECTURE DIAGRAM

**Presentation Layer**, **Application Layer**, and **Database Layer**. The **Presentation Layer** includes user interfaces like web and mobile apps, communicating with the **Application Layer** via HTTP/S. The **Application Layer** implements core functionalities such as authentication, account management, transactions, and reporting using Java. The **Database Layer**, powered by MySQL, stores structured data in tables like Customers, Accounts, Transactions, Loans, and Audit Logs, ensuring secure and efficient data management.



# ER DIAGRAM

****

* 1. **NORMALISATION**

Normalization is a database design technique used to organize and structure data efficiently in relational databases. The goal is to reduce redundancy (duplicate data) and ensure data integrity by organizing data into tables and relationships that minimize anomalies (such as insertion, update, and deletion anomalies).

**Raw Database**

| **CustomerID** | **FirstName** | **LastName** | **DateOfBirth** | **Address** | **PhoneNumber** | **Email** | **CreatedAt** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | John | Doe | 1985-07-14 | 123 Elm St, NY | 1234567890 | john.doe@example.com | 2023-01-01 10:00:00 |
| 2 | Jane | Smith | 1990-03-22 | 456 Oak St, CA | 2345678901 | jane.smith@example.com | 2023-02-15 14:00:00 |
| 3 | Robert | Brown | 1982-11-10 | 789 Pine St, TX | 3456789012 | robert.brown@example.com | 2023-03-20 09:30:00 |

**First Normal Form (1NF)**

To achieve 1NF, we need to ensure that each cell contains only a single value and each record is unique. We'll separate the blood\_types and phone number column into individual rows

| **CustomerID** | **FirstName** | **LastName** | **DateOfBirth** | **Address** | **PhoneNumber** | **Email** | **CreatedAt** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | John | Doe | 1985-07-14 | 123 Elm St, NY | 1234567890 | john.doe@example.com | 2023-01-01 10:00:00 |
| 1 | John | Doe | 1985-07-14 | 123 Elm St, NY | 1234567890 | john.doe122@example.com | 2023-01-01 10:00:00 |
| 2 | Jane | Smith | 1990-03-22 | 456 Oak St, CA | 2345678901 | jane.smith@example.com | 2023-02-15 14:00:00 |
| 3 | Robert | Brown | 1982-11-10 | 789 Pine St, TX | 3456789012 | robert.brown@example.com | 2023-03-20 09:30:00 |

# Chapter 4 PROGRAM CODE

Login Page

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.ResultSet;

import javax.swing.\*;

import java.util.\*;

class BalanceEnquiry extends JFrame implements ActionListener {

JTextField t1, t2;

JButton b1, b2, b3;

JLabel l1, l2, l3;

String pin;

BalanceEnquiry(String pin) {

this.pin = pin;

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/atm.jpg"));

Image i2 = i1.getImage().getScaledInstance(1000, 1180, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l3 = new JLabel(i3);

l3.setBounds(0, 0, 960, 1080);

add(l3);

l1 = new JLabel();

l1.setForeground(Color.WHITE);

l1.setFont(new Font("System", Font.BOLD, 16));

b1 = new JButton("BACK");

setLayout(null);

l1.setBounds(190, 350, 400, 35);

l3.add(l1);

b1.setBounds(390, 633, 150, 35);

l3.add(b1);

int balance = 0;

try{

Conn c1 = new Conn();

ResultSet rs = c1.s.executeQuery("select \* from bank where pin = '"+pin+"'");

while (rs.next()) {

if (rs.getString("mode").equals("Deposit")) {

balance += Integer.parseInt(rs.getString("amount"));

} else {

balance -= Integer.parseInt(rs.getString("amount"));

}

}

}catch(Exception e){}

l1.setText("Your Current Account Balance is Rs "+balance);

b1.addActionListener(this);

setSize(960, 1080);

setUndecorated(true);

setLocation(500, 0);

setVisible(true);

}

public void actionPerformed(ActionEvent ae) {

setVisible(false);

new Transactions(pin).setVisible(true);

}

public static void main(String[] args) {

new BalanceEnquiry("").setVisible(true);

}

}

package ASimulatorSystem;

import java.sql.\*;

public class Conn{

Connection c;

Statement s;

public Conn(){

try{

Class.forName("com.mysql.cj.jdbc.Driver");

c =DriverManager.getConnection("jdbc:mysql:///bankmanagementsystem","root","root");

s =c.createStatement();

}catch(Exception e){

System.out.println(e);

}

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

public class Deposit extends JFrame implements ActionListener{

JTextField t1,t2;

JButton b1,b2,b3;

JLabel l1,l2,l3;

String pin;

Deposit(String pin){

this.pin = pin;

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/atm.jpg"));

Image i2 = i1.getImage().getScaledInstance(1000, 1180, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l3 = new JLabel(i3);

l3.setBounds(0, 0, 960, 1080);

add(l3);

l1 = new JLabel("ENTER AMOUNT YOU WANT TO DEPOSIT");

l1.setForeground(Color.WHITE);

l1.setFont(new Font("System", Font.BOLD, 16));

t1 = new JTextField();

t1.setFont(new Font("Raleway", Font.BOLD, 22));

b1 = new JButton("DEPOSIT");

b2 = new JButton("BACK");

setLayout(null);

l1.setBounds(190,350,400,35);

l3.add(l1);

t1.setBounds(190,420,320,25);

l3.add(t1);

b1.setBounds(390,588,150,35);

l3.add(b1);

b2.setBounds(390,633,150,35);

l3.add(b2);

b1.addActionListener(this);

b2.addActionListener(this);

setSize(960,1080);

setUndecorated(true);

setLocation(500,0);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

try{

String amount = t1.getText();

Date date = new Date();

if(ae.getSource()==b1){

if(t1.getText().equals("")){

JOptionPane.showMessageDialog(null, "Please enter the Amount to you want to Deposit");

}else{

Conn c1 = new Conn();

c1.s.executeUpdate("insert into bank values('"+pin+"', '"+date+"', 'Deposit', '"+amount+"')");

JOptionPane.showMessageDialog(null, "Rs. "+amount+" Deposited Successfully");

setVisible(false);

new Transactions(pin).setVisible(true);

}

}else if(ae.getSource()==b2){

setVisible(false);

new Transactions(pin).setVisible(true);

}

}catch(Exception e){

e.printStackTrace();

}

}

public static void main(String[] args){

new Deposit("").setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

import java.util.Date;

public class FastCash extends JFrame implements ActionListener {

JLabel l1, l2;

JButton b1, b2, b3, b4, b5, b6, b7, b8;

JTextField t1;

String pin;

FastCash(String pin) {

this.pin = pin;

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/atm.jpg"));

Image i2 = i1.getImage().getScaledInstance(1000, 1180, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l3 = new JLabel(i3);

l3.setBounds(0, 0, 960, 1080);

add(l3);

l1 = new JLabel("SELECT WITHDRAWL AMOUNT");

l1.setForeground(Color.WHITE);

l1.setFont(new Font("System", Font.BOLD, 16));

b1 = new JButton("Rs 100");

b2 = new JButton("Rs 500");

b3 = new JButton("Rs 1000");

b4 = new JButton("Rs 2000");

b5 = new JButton("Rs 5000");

b6 = new JButton("Rs 10000");

b7 = new JButton("BACK");

setLayout(null);

l1.setBounds(235, 400, 700, 35);

l3.add(l1);

b1.setBounds(170, 499, 150, 35);

l3.add(b1);

b2.setBounds(390, 499, 150, 35);

l3.add(b2);

b3.setBounds(170, 543, 150, 35);

l3.add(b3);

b4.setBounds(390, 543, 150, 35);

l3.add(b4);

b5.setBounds(170, 588, 150, 35);

l3.add(b5);

b6.setBounds(390, 588, 150, 35);

l3.add(b6);

b7.setBounds(390, 633, 150, 35);

l3.add(b7);

b1.addActionListener(this);

b2.addActionListener(this);

b3.addActionListener(this);

b4.addActionListener(this);

b5.addActionListener(this);

b6.addActionListener(this);

b7.addActionListener(this);

setSize(960, 1080);

setLocation(500, 0);

setUndecorated(true);

setVisible(true);

}

public void actionPerformed(ActionEvent ae) {

try {

String amount = ((JButton)ae.getSource()).getText().substring(3); //k

Conn c = new Conn();

ResultSet rs = c.s.executeQuery("select \* from bank where pin = '"+pin+"'");

int balance = 0;

while (rs.next()) {

if (rs.getString("mode").equals("Deposit")) {

balance += Integer.parseInt(rs.getString("amount"));

} else {

balance -= Integer.parseInt(rs.getString("amount"));

}

} String num = "17";

if (ae.getSource() != b7 && balance < Integer.parseInt(amount)) {

JOptionPane.showMessageDialog(null, "Insuffient Balance");

return;

}

if (ae.getSource() == b7) {

this.setVisible(false);

new Transactions(pin).setVisible(true);

}else{

Date date = new Date();

c.s.executeUpdate("insert into bank values('"+pin+"', '"+date+"', 'Withdrawl', '"+amount+"')");

JOptionPane.showMessageDialog(null, "Rs. "+amount+" Debited Successfully");

setVisible(false);

new Transactions(pin).setVisible(true);

}

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

new FastCash("").setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class Login extends JFrame implements ActionListener{

JLabel l1,l2,l3;

JTextField tf1;

JPasswordField pf2;

JButton b1,b2,b3;

Login(){

setTitle("AUTOMATED TELLER MACHINE");

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/logo.jpg"));

Image i2 = i1.getImage().getScaledInstance(100, 100, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l11 = new JLabel(i3);

l11.setBounds(70, 10, 100, 100);

add(l11);

l1 = new JLabel("WELCOME TO ATM");

l1.setFont(new Font("Osward", Font.BOLD, 38));

l1.setBounds(200,40,450,40);

add(l1);

l2 = new JLabel("Card No:");

l2.setFont(new Font("Raleway", Font.BOLD, 28));

l2.setBounds(125,150,375,30);

add(l2);

tf1 = new JTextField(15);

tf1.setBounds(300,150,230,30);

tf1.setFont(new Font("Arial", Font.BOLD, 14));

add(tf1);

l3 = new JLabel("PIN:");

l3.setFont(new Font("Raleway", Font.BOLD, 28));

l3.setBounds(125,220,375,30);

add(l3);

pf2 = new JPasswordField(15);

pf2.setFont(new Font("Arial", Font.BOLD, 14));

pf2.setBounds(300,220,230,30);

add(pf2);

b1 = new JButton("SIGN IN");

b1.setBackground(Color.BLACK);

b1.setForeground(Color.WHITE);

b2 = new JButton("CLEAR");

b2.setBackground(Color.BLACK);

b2.setForeground(Color.WHITE);

b3 = new JButton("SIGN UP");

b3.setBackground(Color.BLACK);

b3.setForeground(Color.WHITE);

setLayout(null);

b1.setFont(new Font("Arial", Font.BOLD, 14));

b1.setBounds(300,300,100,30);

add(b1);

b2.setFont(new Font("Arial", Font.BOLD, 14));

b2.setBounds(430,300,100,30);

add(b2);

b3.setFont(new Font("Arial", Font.BOLD, 14));

b3.setBounds(300,350,230,30);

add(b3);

b1.addActionListener(this);

b2.addActionListener(this);

b3.addActionListener(this);

getContentPane().setBackground(Color.WHITE);

setSize(800,480);

setLocation(550,200);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

try{

if(ae.getSource()==b1){

Conn c1 = new Conn();

String cardno = tf1.getText();

String pin = pf2.getText();

String q = "select \* from login where cardno = '"+cardno+"' and pin = '"+pin+"'";

ResultSet rs = c1.s.executeQuery(q);

if(rs.next()){

setVisible(false);

new Transactions(pin).setVisible(true);

}else{

JOptionPane.showMessageDialog(null, "Incorrect Card Number or PIN");

}

}else if(ae.getSource()==b2){

tf1.setText("");

pf2.setText("");

}else if(ae.getSource()==b3){

setVisible(false);

new Signup().setVisible(true);

}

}catch(Exception e){

e.printStackTrace();

}

}

public static void main(String[] args){

new Login().setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class MiniStatement extends JFrame implements ActionListener{

JButton b1, b2;

JLabel l1;

MiniStatement(String pin){

super("Mini Statement");

getContentPane().setBackground(Color.WHITE);

setSize(400,600);

setLocation(20,20);

l1 = new JLabel();

add(l1);

JLabel l2 = new JLabel("Indian Bank");

l2.setBounds(150, 20, 100, 20);

add(l2);

JLabel l3 = new JLabel();

l3.setBounds(20, 80, 300, 20);

add(l3);

JLabel l4 = new JLabel();

l4.setBounds(20, 400, 300, 20);

add(l4);

try{

Conn c = new Conn();

ResultSet rs = c.s.executeQuery("select \* from login where pin = '"+pin+"'");

while(rs.next()){

l3.setText("Card Number: " + rs.getString("cardno").substring(0, 4) + "XXXXXXXX" + rs.getString("cardno").substring(12));

}

}catch(Exception e){}

try{

int balance = 0;

Conn c1 = new Conn();

ResultSet rs = c1.s.executeQuery("SELECT \* FROM bank where pin = '"+pin+"'");

while(rs.next()){

l1.setText(l1.getText() + "<html>"+rs.getString("date")+ "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;" + rs.getString("mode") + "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;" + rs.getString("amount") + "<br><br><html>");

if(rs.getString("mode").equals("Deposit")){

balance += Integer.parseInt(rs.getString("amount"));

}else{

balance -= Integer.parseInt(rs.getString("amount"));

}

}

l4.setText("Your total Balance is Rs "+balance);

}catch(Exception e){

e.printStackTrace();

}

setLayout(null);

b1 = new JButton("Exit");

add(b1);

b1.addActionListener(this);

l1.setBounds(20, 140, 400, 200);

b1.setBounds(20, 500, 100, 25);

}

public void actionPerformed(ActionEvent ae){

this.setVisible(false);

}

public static void main(String[] args){

new MiniStatement("").setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class Pin extends JFrame implements ActionListener{

JPasswordField t1,t2;

JButton b1,b2;

JLabel l1,l2,l3;

String pin;

Pin(String pin){

this.pin = pin;

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/atm.jpg"));

Image i2 = i1.getImage().getScaledInstance(1000, 1180, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l4 = new JLabel(i3);

l4.setBounds(0, 0, 960, 1080);

add(l4);

l1 = new JLabel("CHANGE YOUR PIN");

l1.setFont(new Font("System", Font.BOLD, 16));

l1.setForeground(Color.WHITE);

l2 = new JLabel("New PIN:");

l2.setFont(new Font("System", Font.BOLD, 16));

l2.setForeground(Color.WHITE);

l3 = new JLabel("Re-Enter New PIN:");

l3.setFont(new Font("System", Font.BOLD, 16));

l3.setForeground(Color.WHITE);

t1 = new JPasswordField();

t1.setFont(new Font("Raleway", Font.BOLD, 25));

t2 = new JPasswordField();

t2.setFont(new Font("Raleway", Font.BOLD, 25));

b1 = new JButton("CHANGE");

b2 = new JButton("BACK");

b1.addActionListener(this);

b2.addActionListener(this);

setLayout(null);

l1.setBounds(280,330,800,35);

l4.add(l1);

l2.setBounds(180,390,150,35);

l4.add(l2);

l3.setBounds(180,440,200,35);

l4.add(l3);

t1.setBounds(350,390,180,25);

l4.add(t1);

t2.setBounds(350,440,180,25);

l4.add(t2);

b1.setBounds(390,588,150,35);

l4.add(b1);

b2.setBounds(390,633,150,35);

l4.add(b2);

setSize(960,1080);

setLocation(500,0);

setUndecorated(true);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

try{

String npin = t1.getText();

String rpin = t2.getText();

if(!npin.equals(rpin)){

JOptionPane.showMessageDialog(null, "Entered PIN does not match");

return;

}

if(ae.getSource()==b1){

if (t1.getText().equals("")){

JOptionPane.showMessageDialog(null, "Enter New PIN");

}

if (t2.getText().equals("")){

JOptionPane.showMessageDialog(null, "Re-Enter new PIN");

}

Conn c1 = new Conn();

String q1 = "update bank set pin = '"+rpin+"' where pin = '"+pin+"' ";

String q2 = "update login set pin = '"+rpin+"' where pin = '"+pin+"' ";

String q3 = "update signup3 set pin = '"+rpin+"' where pin = '"+pin+"' ";

c1.s.executeUpdate(q1);

c1.s.executeUpdate(q2);

c1.s.executeUpdate(q3);

JOptionPane.showMessageDialog(null, "PIN changed successfully");

setVisible(false);

new Transactions(rpin).setVisible(true);

}else if(ae.getSource()==b2){

new Transactions(pin).setVisible(true);

setVisible(false);

}

}catch(Exception e){

e.printStackTrace();

}

}

public static void main(String[] args){

new Pin("").setVisible(true);

}

}

package ASimulatorSystem;

import javax.imageio.ImageIO;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.File;

public final class Practice {

boolean negative;

public Practice() {

this(false);

}

public Practice(final boolean negative) {

this.negative = negative;

}

public String convert(final BufferedImage image) {

StringBuilder sb = new StringBuilder((image.getWidth() + 1) \* image.getHeight());

for (int y = 0; y < image.getHeight(); y++) {

if (sb.length() != 0) sb.append("\n");

for (int x = 0; x < image.getWidth(); x++) {

Color pixelColor = new Color(image.getRGB(x, y));

double gValue = (double) pixelColor.getRed() \* 0.2989 + (double) pixelColor.getBlue() \* 0.5870 + (double) pixelColor.getGreen() \* 0.1140;

final char s = negative ? returnStrNeg(gValue) : returnStrPos(gValue);

sb.append(s);

}

}

return sb.toString();

}

/\*\*

\* Create a new string and assign to it a string based on the grayscale value.

\* If the grayscale value is very high, the pixel is very bright and assign characters

\* such as . and , that do not appear very dark. If the grayscale value is very lowm the pixel is very dark,

\* assign characters such as # and @ which appear very dark.

\*

\* @param g grayscale

\* @return char

\*/

private char returnStrPos(double g)//takes the grayscale value as parameter

{

final char str;

if (g >= 230.0) {

str = ' ';

} else if (g >= 200.0) {

str = '.';

} else if (g >= 180.0) {

str = '\*';

} else if (g >= 160.0) {

str = ':';

} else if (g >= 130.0) {

str = 'o';

} else if (g >= 100.0) {

str = '&';

} else if (g >= 70.0) {

str = '8';

} else if (g >= 50.0) {

str = '#';

} else {

str = '@';

}

return str; // return the character

}

/\*\*

\* Same method as above, except it reverses the darkness of the pixel. A dark pixel is given a light character and vice versa.

\*

\* @param g grayscale

\* @return char

\*/

private char returnStrNeg(double g) {

final char str;

if (g >= 230.0) {

str = '@';

} else if (g >= 200.0) {

str = '#';

} else if (g >= 180.0) {

str = '8';

} else if (g >= 160.0) {

str = '&';

} else if (g >= 130.0) {

str = 'o';

} else if (g >= 100.0) {

str = ':';

} else if (g >= 70.0) {

str = '\*';

} else if (g >= 50.0) {

str = '.';

} else {

str = ' ';

}

return str;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setFileFilter(new FileNameExtensionFilter("Images", "jpg", "gif", "png"));

while (fileChooser.showOpenDialog(null) == JFileChooser.APPROVE\_OPTION) {

try {

File f = fileChooser.getSelectedFile();

final BufferedImage image = ImageIO.read(f);

if (image == null) throw new IllegalArgumentException(f + " is not a valid image.");

final String ascii = new Practice().convert(image);

final JTextArea textArea = new JTextArea(ascii, image.getHeight(), image.getWidth());

textArea.setFont(new Font("Monospaced", Font.BOLD, 5));

textArea.setEditable(false);

final JDialog dialog = new JOptionPane(new JScrollPane(textArea), JOptionPane.PLAIN\_MESSAGE).createDialog(Practice.class.getName());

dialog.setResizable(true);

dialog.setVisible(true);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, e.toString(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

System.exit(0);

}

});

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

import com.toedter.calendar.JDateChooser;

import java.util.\*;

public class Signup extends JFrame implements ActionListener{

JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15;

JTextField t1,t2,t3,t4,t5,t6,t7;

JRadioButton r1,r2,r3,r4,r5;

JButton b;

JDateChooser dateChooser;

Random ran = new Random();

long first4 = (ran.nextLong() % 9000L) + 1000L;

String first = "" + Math.abs(first4);

Signup(){

setTitle("NEW ACCOUNT APPLICATION FORM");

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/logo.jpg"));

Image i2 = i1.getImage().getScaledInstance(100, 100, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l11 = new JLabel(i3);

l11.setBounds(20, 0, 100, 100);

add(l11);

l1 = new JLabel("APPLICATION FORM NO. "+first);

l1.setFont(new Font("Raleway", Font.BOLD, 38));

l2 = new JLabel("Page 1: Personal Details");

l2.setFont(new Font("Raleway", Font.BOLD, 22));

l3 = new JLabel("Name:");

l3.setFont(new Font("Raleway", Font.BOLD, 20));

l4 = new JLabel("Father's Name:");

l4.setFont(new Font("Raleway", Font.BOLD, 20));

l5 = new JLabel("Date of Birth:");

l5.setFont(new Font("Raleway", Font.BOLD, 20));

l6 = new JLabel("Gender:");

l6.setFont(new Font("Raleway", Font.BOLD, 20));

l7 = new JLabel("Email Address:");

l7.setFont(new Font("Raleway", Font.BOLD, 20));

l8 = new JLabel("Marital Status:");

l8.setFont(new Font("Raleway", Font.BOLD, 20));

l9 = new JLabel("Address:");

l9.setFont(new Font("Raleway", Font.BOLD, 20));

l10 = new JLabel("City:");

l10.setFont(new Font("Raleway", Font.BOLD, 20));

l11 = new JLabel("Pin Code:");

l11.setFont(new Font("Raleway", Font.BOLD, 20));

l12 = new JLabel("State:");

l12.setFont(new Font("Raleway", Font.BOLD, 20));

l13 = new JLabel("Date");

l13.setFont(new Font("Raleway", Font.BOLD, 14));

l14 = new JLabel("Month");

l14.setFont(new Font("Raleway", Font.BOLD, 14));

l15 = new JLabel("Year");

l15.setFont(new Font("Raleway", Font.BOLD, 14));

t1 = new JTextField();

t1.setFont(new Font("Raleway", Font.BOLD, 14));

t2 = new JTextField();

t2.setFont(new Font("Raleway", Font.BOLD, 14));

t3 = new JTextField();

t3.setFont(new Font("Raleway", Font.BOLD, 14));

t4 = new JTextField();

t4.setFont(new Font("Raleway", Font.BOLD, 14));

t5 = new JTextField();

t5.setFont(new Font("Raleway", Font.BOLD, 14));

t6 = new JTextField();

t6.setFont(new Font("Raleway", Font.BOLD, 14));

t7 = new JTextField();

t7.setFont(new Font("Raleway", Font.BOLD, 14));

b = new JButton("Next");

b.setFont(new Font("Raleway", Font.BOLD, 14));

b.setBackground(Color.BLACK);

b.setForeground(Color.WHITE);

r1 = new JRadioButton("Male");

r1.setFont(new Font("Raleway", Font.BOLD, 14));

r1.setBackground(Color.WHITE);

r2 = new JRadioButton("Female");

r2.setFont(new Font("Raleway", Font.BOLD, 14));

r2.setBackground(Color.WHITE);

ButtonGroup groupgender = new ButtonGroup();

groupgender.add(r1);

groupgender.add(r2);

r3 = new JRadioButton("Married");

r3.setFont(new Font("Raleway", Font.BOLD, 14));

r3.setBackground(Color.WHITE);

r4 = new JRadioButton("Unmarried");

r4.setFont(new Font("Raleway", Font.BOLD, 14));

r4.setBackground(Color.WHITE);

r5 = new JRadioButton("Other");

r5.setFont(new Font("Raleway", Font.BOLD, 14));

r5.setBackground(Color.WHITE);

ButtonGroup groupstatus = new ButtonGroup();

groupstatus.add(r3);

groupstatus.add(r4);

groupstatus.add(r5);

dateChooser = new JDateChooser();

//dateChooser.setBorder(new LineBorder(new Color(0, 0, 0), 1, true));

dateChooser.setForeground(new Color(105, 105, 105));

dateChooser.setBounds(137, 337, 200, 29);

add(dateChooser);

setLayout(null);

l1.setBounds(140,20,600,40);

add(l1);

l2.setBounds(290,80,600,30);

add(l2);

l3.setBounds(100,140,100,30);

add(l3);

t1.setBounds(300,140,400,30);

add(t1);

l4.setBounds(100,190,200,30);

add(l4);

t2.setBounds(300,190,400,30);

add(t2);

l5.setBounds(100,240,200,30);

add(l5);

dateChooser.setBounds(300, 240, 400, 30);

l6.setBounds(100,290,200,30);

add(l6);

r1.setBounds(300,290,60,30);

add(r1);

r2.setBounds(450,290,90,30);

add(r2);

l7.setBounds(100,340,200,30);

add(l7);

t3.setBounds(300,340,400,30);

add(t3);

l8.setBounds(100,390,200,30);

add(l8);

r3.setBounds(300,390,100,30);

add(r3);

r4.setBounds(450,390,100,30);

add(r4);

r5.setBounds(635,390,100,30);

add(r5);

l9.setBounds(100,440,200,30);

add(l9);

t4.setBounds(300,440,400,30);

add(t4);

l10.setBounds(100,490,200,30);

add(l10);

t5.setBounds(300,490,400,30);

add(t5);

l11.setBounds(100,540,200,30);

add(l11);

t6.setBounds(300,540,400,30);

add(t6);

l12.setBounds(100,590,200,30);

add(l12);

t7.setBounds(300,590,400,30);

add(t7);

b.setBounds(620,660,80,30);

add(b);

b.addActionListener(this);

getContentPane().setBackground(Color.WHITE);

setSize(850,800);

setLocation(500,120);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

String formno = first;

String name = t1.getText();

String fname = t2.getText();

String dob = ((JTextField) dateChooser.getDateEditor().getUiComponent()).getText();

String gender = null;

if(r1.isSelected()){

gender = "Male";

}else if(r2.isSelected()){

gender = "Female";

}

String email = t3.getText();

String marital = null;

if(r3.isSelected()){

marital = "Married";

}else if(r4.isSelected()){

marital = "Unmarried";

}else if(r5.isSelected()){

marital = "Other";

}

String address = t4.getText();

String city = t5.getText();

String pincode = t6.getText();

String state = t7.getText();

try{

if(t6.getText().equals("")){

JOptionPane.showMessageDialog(null, "Fill all the required fields");

}else{

Conn c1 = new Conn();

String q1 = "insert into signup values('"+formno+"','"+name+"','"+fname+"','"+dob+"','"+gender+"','"+email+"','"+marital+"','"+address+"','"+city+"','"+pincode+"','"+state+"')";

c1.s.executeUpdate(q1);

new Signup2(first).setVisible(true);

setVisible(false);

}

}catch(Exception e){

e.printStackTrace();

}

}

public static void main(String[] args){

new Signup().setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class Signup2 extends JFrame implements ActionListener{

JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13;

JButton b;

JRadioButton r1,r2,r3,r4;

JTextField t1,t2,t3;

JComboBox c1,c2,c3,c4,c5;

String formno;

Signup2(String formno){

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/logo.jpg"));

Image i2 = i1.getImage().getScaledInstance(100, 100, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l14 = new JLabel(i3);

l14.setBounds(150, 0, 100, 100);

add(l14);

this.formno = formno;

setTitle("NEW ACCOUNT APPLICATION FORM - PAGE 2");

l1 = new JLabel("Page 2: Additonal Details");

l1.setFont(new Font("Raleway", Font.BOLD, 22));

l2 = new JLabel("Religion:");

l2.setFont(new Font("Raleway", Font.BOLD, 18));

l3 = new JLabel("Category:");

l3.setFont(new Font("Raleway", Font.BOLD, 18));

l4 = new JLabel("Income:");

l4.setFont(new Font("Raleway", Font.BOLD, 18));

l5 = new JLabel("Educational");

l5.setFont(new Font("Raleway", Font.BOLD, 18));

l11 = new JLabel("Qualification:");

l11.setFont(new Font("Raleway", Font.BOLD, 18));

l6 = new JLabel("Occupation:");

l6.setFont(new Font("Raleway", Font.BOLD, 18));

l7 = new JLabel("PAN Number:");

l7.setFont(new Font("Raleway", Font.BOLD, 18));

l8 = new JLabel("Aadhar Number:");

l8.setFont(new Font("Raleway", Font.BOLD, 18));

l9 = new JLabel("Senior Citizen:");

l9.setFont(new Font("Raleway", Font.BOLD, 18));

l10 = new JLabel("Existing Account:");

l10.setFont(new Font("Raleway", Font.BOLD, 18));

l12 = new JLabel("Form No:");

l12.setFont(new Font("Raleway", Font.BOLD, 13));

l13 = new JLabel(formno);

l13.setFont(new Font("Raleway", Font.BOLD, 13));

b = new JButton("Next");

b.setFont(new Font("Raleway", Font.BOLD, 14));

b.setBackground(Color.BLACK);

b.setForeground(Color.WHITE);

t1 = new JTextField();

t1.setFont(new Font("Raleway", Font.BOLD, 14));

t2 = new JTextField();

t2.setFont(new Font("Raleway", Font.BOLD, 14));

r1 = new JRadioButton("Yes");

r1.setFont(new Font("Raleway", Font.BOLD, 14));

r1.setBackground(Color.WHITE);

r2 = new JRadioButton("No");

r2.setFont(new Font("Raleway", Font.BOLD, 14));

r2.setBackground(Color.WHITE);

r3 = new JRadioButton("Yes");

r3.setFont(new Font("Raleway", Font.BOLD, 14));

r3.setBackground(Color.WHITE);

r4 = new JRadioButton("No");

r4.setFont(new Font("Raleway", Font.BOLD, 14));

r4.setBackground(Color.WHITE);

String religion[] = {"Hindu","Muslim","Sikh","Christian","Other"};

c1 = new JComboBox(religion);

c1.setBackground(Color.WHITE);

c1.setFont(new Font("Raleway", Font.BOLD, 14));

String category[] = {"General","OBC","SC","ST","Other"};

c2 = new JComboBox(category);

c2.setBackground(Color.WHITE);

c2.setFont(new Font("Raleway", Font.BOLD, 14));

String income[] = {"Null","<1,50,000","<2,50,000","<5,00,000","Upto 10,00,000","Above 10,00,000"};

c3 = new JComboBox(income);

c3.setBackground(Color.WHITE);

c3.setFont(new Font("Raleway", Font.BOLD, 14));

String education[] = {"Non-Graduate","Graduate","Post-Graduate","Doctrate","Others"};

c4 = new JComboBox(education);

c4.setBackground(Color.WHITE);

c4.setFont(new Font("Raleway", Font.BOLD, 14));

String occupation[] = {"Salaried","Self-Employmed","Business","Student","Retired","Others"};

c5 = new JComboBox(occupation);

c5.setBackground(Color.WHITE);

c5.setFont(new Font("Raleway", Font.BOLD, 14));

setLayout(null);

l12.setBounds(700,10,60,30);

add(l12);

l13.setBounds(760,10,60,30);

add(l13);

l1.setBounds(280,30,600,40);

add(l1);

l2.setBounds(100,120,100,30);

add(l2);

c1.setBounds(350,120,320,30);

add(c1);

l3.setBounds(100,170,100,30);

add(l3);

c2.setBounds(350,170,320,30);

add(c2);

l4.setBounds(100,220,100,30);

add(l4);

c3.setBounds(350,220,320,30);

add(c3);

l5.setBounds(100,270,150,30);

add(l5);

c4.setBounds(350,270,320,30);

add(c4);

l11.setBounds(100,290,150,30);

add(l11);

l6.setBounds(100,340,150,30);

add(l6);

c5.setBounds(350,340,320,30);

add(c5);

l7.setBounds(100,390,150,30);

add(l7);

t1.setBounds(350,390,320,30);

add(t1);

l8.setBounds(100,440,180,30);

add(l8);

t2.setBounds(350,440,320,30);

add(t2);

l9.setBounds(100,490,150,30);

add(l9);

r1.setBounds(350,490,100,30);

add(r1);

r2.setBounds(460,490,100,30);

add(r2);

l10.setBounds(100,540,180,30);

add(l10);

r3.setBounds(350,540,100,30);

add(r3);

r4.setBounds(460,540,100,30);

add(r4);

b.setBounds(570,640,100,30);

add(b);

b.addActionListener(this);

getContentPane().setBackground(Color.WHITE);

setSize(850,750);

setLocation(500,120);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

String religion = (String)c1.getSelectedItem();

String category = (String)c2.getSelectedItem();

String income = (String)c3.getSelectedItem();

String education = (String)c4.getSelectedItem();

String occupation = (String)c5.getSelectedItem();

String pan = t1.getText();

String aadhar = t2.getText();

String scitizen = "";

if(r1.isSelected()){

scitizen = "Yes";

}

else if(r2.isSelected()){

scitizen = "No";

}

String eaccount = "";

if(r3.isSelected()){

eaccount = "Yes";

}else if(r4.isSelected()){

eaccount = "No";

}

try{

if(t2.getText().equals("")){

JOptionPane.showMessageDialog(null, "Fill all the required fields");

}else{

Conn c1 = new Conn();

String q1 = "insert into signup2 values('"+formno+"','"+religion+"','"+category+"','"+income+"','"+education+"','"+occupation+"','"+pan+"','"+aadhar+"','"+scitizen+"','"+eaccount+"')";

c1.s.executeUpdate(q1);

new Signup3(formno).setVisible(true);

setVisible(false);

}

}catch(Exception ex){

ex.printStackTrace();

}

}

public static void main(String[] args){

new Signup2("").setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

import java.util.\*;

public class Signup3 extends JFrame implements ActionListener{

JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12;

JRadioButton r1,r2,r3,r4;

JButton b1,b2;

JCheckBox c1,c2,c3,c4,c5,c6,c7;

String formno;

Signup3(String formno){

this.formno = formno;

setTitle("NEW ACCOUNT APPLICATION FORM - PAGE 3");

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/logo.jpg"));

Image i2 = i1.getImage().getScaledInstance(100, 100, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l14 = new JLabel(i3);

l14.setBounds(150, 0, 100, 100);

add(l14);

l1 = new JLabel("Page 3: Account Details");

l1.setFont(new Font("Raleway", Font.BOLD, 22));

l2 = new JLabel("Account Type:");

l2.setFont(new Font("Raleway", Font.BOLD, 18));

l3 = new JLabel("Card Number:");

l3.setFont(new Font("Raleway", Font.BOLD, 18));

l4 = new JLabel("XXXX-XXXX-XXXX-4184");

l4.setFont(new Font("Raleway", Font.BOLD, 18));

l5 = new JLabel("(Your 16-digit Card number)");

l5.setFont(new Font("Raleway", Font.BOLD, 12));

l6 = new JLabel("It would appear on ATM Card/Cheque Book and Statements");

l6.setFont(new Font("Raleway", Font.BOLD, 12));

l7 = new JLabel("PIN:");

l7.setFont(new Font("Raleway", Font.BOLD, 18));

l8 = new JLabel("XXXX");

l8.setFont(new Font("Raleway", Font.BOLD, 18));

l9 = new JLabel("(4-digit password)");

l9.setFont(new Font("Raleway", Font.BOLD, 12));

l10 = new JLabel("Services Required:");

l10.setFont(new Font("Raleway", Font.BOLD, 18));

l11 = new JLabel("Form No:");

l11.setFont(new Font("Raleway", Font.BOLD, 14));

l12 = new JLabel(formno);

l12.setFont(new Font("Raleway", Font.BOLD, 14));

b1 = new JButton("Submit");

b1.setFont(new Font("Raleway", Font.BOLD, 14));

b1.setBackground(Color.BLACK);

b1.setForeground(Color.WHITE);

b2 = new JButton("Cancel");

b2.setFont(new Font("Raleway", Font.BOLD, 14));

b2.setBackground(Color.BLACK);

b2.setForeground(Color.WHITE);

c1 = new JCheckBox("ATM CARD");

c1.setBackground(Color.WHITE);

c1.setFont(new Font("Raleway", Font.BOLD, 16));

c2 = new JCheckBox("Internet Banking");

c2.setBackground(Color.WHITE);

c2.setFont(new Font("Raleway", Font.BOLD, 16));

c3 = new JCheckBox("Mobile Banking");

c3.setBackground(Color.WHITE);

c3.setFont(new Font("Raleway", Font.BOLD, 16));

c4 = new JCheckBox("EMAIL Alerts");

c4.setBackground(Color.WHITE);

c4.setFont(new Font("Raleway", Font.BOLD, 16));

c5 = new JCheckBox("Cheque Book");

c5.setBackground(Color.WHITE);

c5.setFont(new Font("Raleway", Font.BOLD, 16));

c6 = new JCheckBox("E-Statement");

c6.setBackground(Color.WHITE);

c6.setFont(new Font("Raleway", Font.BOLD, 16));

c7 = new JCheckBox("I hereby declares that the above entered details correct to th best of my knowledge.",true);

c7.setBackground(Color.WHITE);

c7.setFont(new Font("Raleway", Font.BOLD, 12));

r1 = new JRadioButton("Saving Account");

r1.setFont(new Font("Raleway", Font.BOLD, 16));

r1.setBackground(Color.WHITE);

r2 = new JRadioButton("Fixed Deposit Account");

r2.setFont(new Font("Raleway", Font.BOLD, 16));

r2.setBackground(Color.WHITE);

r3 = new JRadioButton("Current Account");

r3.setFont(new Font("Raleway", Font.BOLD, 16));

r3.setBackground(Color.WHITE);

r4 = new JRadioButton("Recurring Deposit Account");

r4.setFont(new Font("Raleway", Font.BOLD, 16));

r4.setBackground(Color.WHITE);

ButtonGroup groupgender = new ButtonGroup();

groupgender.add(r1);

groupgender.add(r2);

groupgender.add(r3);

groupgender.add(r4);

setLayout(null);

l11.setBounds(700,10,70,30);

add(l11);

l12.setBounds(770,10,40,30);

add(l12);

l1.setBounds(280,40,400,40);

add(l1);

l2.setBounds(100,140,200,30);

add(l2);

r1.setBounds(100,180,150,30);

add(r1);

r2.setBounds(350,180,300,30);

add(r2);

r3.setBounds(100,220,250,30);

add(r3);

r4.setBounds(350,220,250,30);

add(r4);

l3.setBounds(100,300,200,30);

add(l3);

l4.setBounds(330,300,250,30);

add(l4);

l5.setBounds(100,330,200,20);

add(l5);

l6.setBounds(330,330,500,20);

add(l6);

l7.setBounds(100,370,200,30);

add(l7);

l8.setBounds(330,370,200,30);

add(l8);

l9.setBounds(100,400,200,20);

add(l9);

l10.setBounds(100,450,200,30);

add(l10);

c1.setBounds(100,500,200,30);

add(c1);

c2.setBounds(350,500,200,30);

add(c2);

c3.setBounds(100,550,200,30);

add(c3);

c4.setBounds(350,550,200,30);

add(c4);

c5.setBounds(100,600,200,30);

add(c5);

c6.setBounds(350,600,200,30);

add(c6);

c7.setBounds(100,680,600,20);

add(c7);

b1.setBounds(250,720,100,30);

add(b1);

b2.setBounds(420,720,100,30);

add(b2);

getContentPane().setBackground(Color.WHITE);

setSize(850,850);

setLocation(500,120);

setVisible(true);

b1.addActionListener(this);

b2.addActionListener(this);

}

public void actionPerformed(ActionEvent ae){

String atype = null;

if(r1.isSelected()){

atype = "Saving Account";

}

else if(r2.isSelected()){

atype = "Fixed Deposit Account";

}

else if(r3.isSelected()){

atype = "Current Account";

}else if(r4.isSelected()){

atype = "Recurring Deposit Account";

}

Random ran = new Random();

long first7 = (ran.nextLong() % 90000000L) + 5040936000000000L;

String cardno = "" + Math.abs(first7);

long first3 = (ran.nextLong() % 9000L) + 1000L;

String pin = "" + Math.abs(first3);

String facility = "";

if(c1.isSelected()){

facility = facility + " ATM Card";

}

if(c2.isSelected()){

facility = facility + " Internet Banking";

}

if(c3.isSelected()){

facility = facility + " Mobile Banking";

}

if(c4.isSelected()){

facility = facility + " EMAIL Alerts";

}

if(c5.isSelected()){

facility = facility + " Cheque Book";

}

if(c6.isSelected()){

facility = facility + " E-Statement";

}

try{

if(ae.getSource()==b1){

if(atype.equals("")){

JOptionPane.showMessageDialog(null, "Fill all the required fields");

}else{

Conn c1 = new Conn();

String q1 = "insert into signup3 values('"+formno+"','"+atype+"','"+cardno+"','"+pin+"','"+facility+"')";

String q2 = "insert into login values('"+formno+"','"+cardno+"','"+pin+"')";

c1.s.executeUpdate(q1);

c1.s.executeUpdate(q2);

JOptionPane.showMessageDialog(null, "Card Number: " + cardno + "\n Pin:"+ pin);

new Deposit(pin).setVisible(true);

setVisible(false);

}

}else if(ae.getSource()==b2){

System.exit(0);

}

}catch(Exception ex){

ex.printStackTrace();

}

}

public static void main(String[] args){

new Signup3("").setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class Transactions extends JFrame implements ActionListener{

JLabel l1;

JButton b1,b2,b3,b4,b5,b6,b7;

String pin;

Transactions(String pin){

this.pin = pin;

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/atm.jpg"));

Image i2 = i1.getImage().getScaledInstance(1000, 1180, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l2 = new JLabel(i3);

l2.setBounds(0, 0, 960, 1080);

add(l2);

l1 = new JLabel("Please Select Your Transaction");

l1.setForeground(Color.WHITE);

l1.setFont(new Font("System", Font.BOLD, 16));

b1 = new JButton("DEPOSIT");

b2 = new JButton("CASH WITHDRAWL");

b3 = new JButton("FAST CASH");

b4 = new JButton("MINI STATEMENT");

b5 = new JButton("PIN CHANGE");

b6 = new JButton("BALANCE ENQUIRY");

b7 = new JButton("EXIT");

setLayout(null);

l1.setBounds(235,400,700,35);

l2.add(l1);

b1.setBounds(170,499,150,35);

l2.add(b1);

b2.setBounds(390,499,150,35);

l2.add(b2);

b3.setBounds(170,543,150,35);

l2.add(b3);

b4.setBounds(390,543,150,35);

l2.add(b4);

b5.setBounds(170,588,150,35);

l2.add(b5);

b6.setBounds(390,588,150,35);

l2.add(b6);

b7.setBounds(390,633,150,35);

l2.add(b7);

b1.addActionListener(this);

b2.addActionListener(this);

b3.addActionListener(this);

b4.addActionListener(this);

b5.addActionListener(this);

b6.addActionListener(this);

b7.addActionListener(this);

setSize(960,1080);

setLocation(500,0);

setUndecorated(true);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

if(ae.getSource()==b1){

setVisible(false);

new Deposit(pin).setVisible(true);

}else if(ae.getSource()==b2){

setVisible(false);

new Withdrawl(pin).setVisible(true);

}else if(ae.getSource()==b3){

setVisible(false);

new FastCash(pin).setVisible(true);

}else if(ae.getSource()==b4){

new MiniStatement(pin).setVisible(true);

}else if(ae.getSource()==b5){

setVisible(false);

new Pin(pin).setVisible(true);

}else if(ae.getSource()==b6){

this.setVisible(false);

new BalanceEnquiry(pin).setVisible(true);

}else if(ae.getSource()==b7){

System.exit(0);

}

}

public static void main(String[] args){

new Transactions("").setVisible(true);

}

}

package ASimulatorSystem;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.Date;

import java.sql.\*;

public class Withdrawl extends JFrame implements ActionListener{

JTextField t1,t2;

JButton b1,b2,b3;

JLabel l1,l2,l3,l4;

String pin;

Withdrawl(String pin){

this.pin = pin;

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/atm.jpg"));

Image i2 = i1.getImage().getScaledInstance(1000, 1180, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel l3 = new JLabel(i3);

l3.setBounds(0, 0, 960, 1080);

add(l3);

l1 = new JLabel("MAXIMUM WITHDRAWAL IS RS.10,000");

l1.setForeground(Color.WHITE);

l1.setFont(new Font("System", Font.BOLD, 16));

l2 = new JLabel("PLEASE ENTER YOUR AMOUNT");

l2.setForeground(Color.WHITE);

l2.setFont(new Font("System", Font.BOLD, 16));

t1 = new JTextField();

t1.setFont(new Font("Raleway", Font.BOLD, 25));

b1 = new JButton("WITHDRAW");

b2 = new JButton("BACK");

setLayout(null);

l1.setBounds(190,350,400,20);

l3.add(l1);

l2.setBounds(190,400,400,20);

l3.add(l2);

t1.setBounds(190,450,330,30);

l3.add(t1);

b1.setBounds(390,588,150,35);

l3.add(b1);

b2.setBounds(390,633,150,35);

l3.add(b2);

b1.addActionListener(this);

b2.addActionListener(this);

setSize(960,1080);

setLocation(500,0);

setUndecorated(true);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

try{

String amount = t1.getText();

Date date = new Date();

if(ae.getSource()==b1){

if(t1.getText().equals("")){

JOptionPane.showMessageDialog(null, "Please enter the Amount to you want to Withdraw");

}else{

Conn c1 = new Conn();

ResultSet rs = c1.s.executeQuery("select \* from bank where pin = '"+pin+"'");

int balance = 0;

while(rs.next()){

if(rs.getString("mode").equals("Deposit")){

balance += Integer.parseInt(rs.getString("amount"));

}else{

balance -= Integer.parseInt(rs.getString("amount"));

}

}

if(balance < Integer.parseInt(amount)){

JOptionPane.showMessageDialog(null, "Insuffient Balance");

return;

}

c1.s.executeUpdate("insert into bank values('"+pin+"', '"+date+"', 'Withdrawl', '"+amount+"')");

JOptionPane.showMessageDialog(null, "Rs. "+amount+" Debited Successfully");

setVisible(false);

new Transactions(pin).setVisible(true);

}

}else if(ae.getSource()==b2){

setVisible(false);

new Transactions(pin).setVisible(true);

}

}catch(Exception e){

e.printStackTrace();

System.out.println("error: "+e);

}

}

public static void main(String[] args){

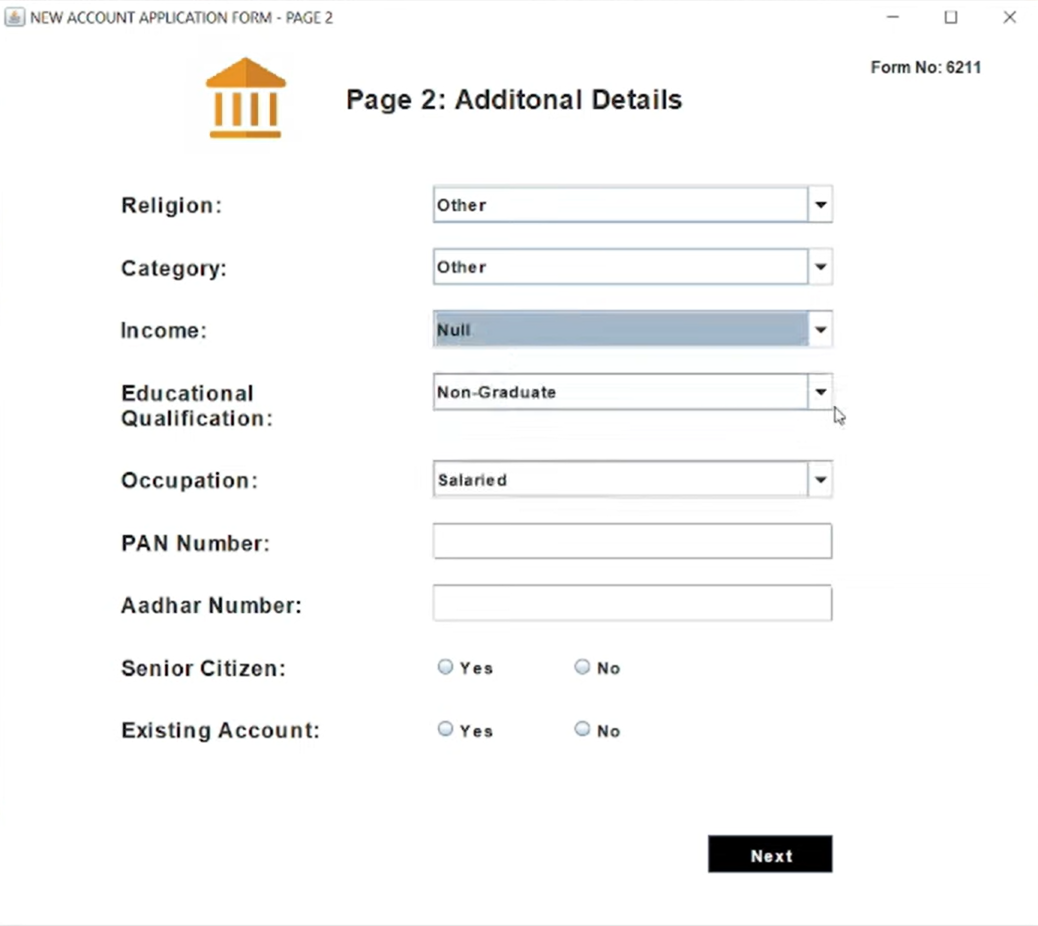
new Withdrawl("").setVisible(true);

}

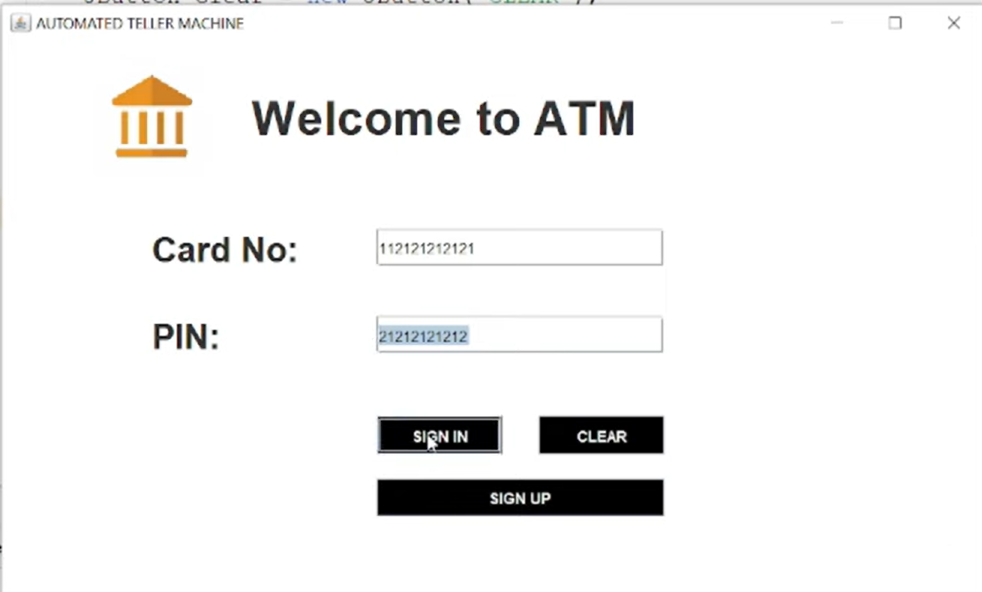
}

# Chapter 5 RESULTS AND DISCUSSION

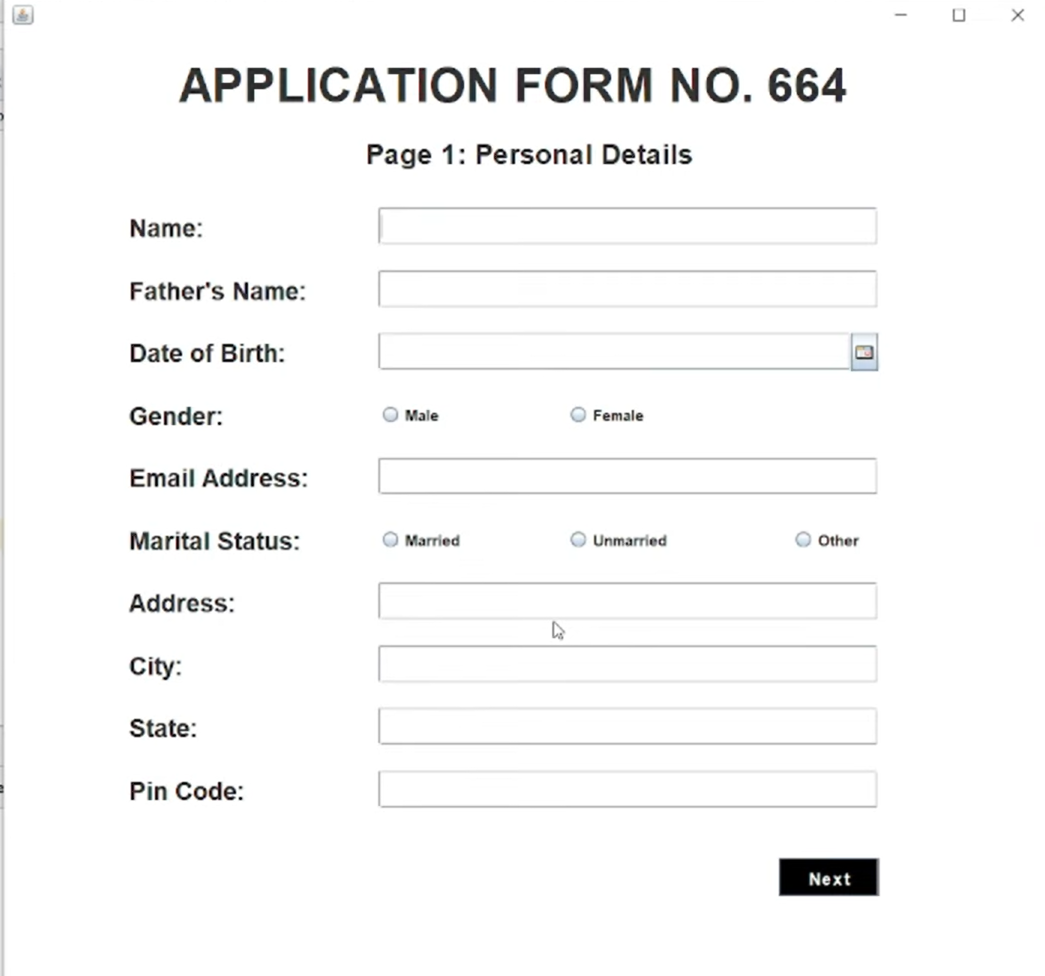
**FILLING DETAILS**

****

**LOGIN PAGE**

****

PERSONAL DETAILS

****

****

****

# Chapter 6 CONCLUSION

**6.1 Conclusion**

The implementation of a **Bank Management System** using Java and MySQL ensures streamlined management of core banking functionalities such as customer management, account handling, transaction processing, and loan management. Key outcomes include:

1. **Efficiency**: Automating banking operations reduces manual workload, minimizes errors, and accelerates processes like account creation, fund transfers, and loan approvals.
2. **Data Integrity and Security**: Leveraging MySQL as the database ensures robust data storage, while implementing encryption and authentication safeguards sensitive information.
3. **Scalability**: A modular architecture with Java-based components allows the system to adapt to growing customer needs, support additional features, and handle increased transaction volumes.
4. **Improved Customer Experience**: Features like real-time transaction tracking, easy account management, and secure online banking enhance user satisfaction and trust.

Overall, the system modernizes banking operations, promotes financial transparency, and builds a strong foundation for digital transformation in the financial sector.

Top of Form

Bottom of Form

# Chapter 7 REFERENCES

**7.1 REFERENCES**

**[1]** [**https://web.ku.ac.th/saranaroo/chap5a.htm**](https://web.ku.ac.th/saranaroo/chap5a.htm)

**[2]** [**http://porcheckin.com/2014/05/blood-**](http://porcheckin.com/2014/05/blood-การบริจาคเลือด/)**การบรจิ าคเลอื ด**[**/**](http://porcheckin.com/2014/05/blood-การบริจาคเลือด/)

1. [**https://www.redcross.or.th/forum/16095**](https://www.redcross.or.th/forum/16095)
2. [**https://www.blood.co.uk/the-donation-process/furtherinformation/tests-**](https://www.blood.co.uk/the-donation-process/further-information/tests-we-carryout/)[**we-carryout/**](https://www.blood.co.uk/the-donation-process/further-information/tests-we-carryout/)
3. Blood Bank Management System M Sai Tarun1, S Ravi kishan2, Shaik Azaad Suraz Basha3, Shaik Raj Ahammad4, U Chandrasekhar5, Neha Bagga6 Department of Computers Science and Engineering Lovely Professional University Jalandhar, India