**A PROJECT REPORT**

**ON**

**ATM STIMULATOR SYSTEM**

**Prepared by**

**KARAN KATARIYA**

**Drs. Kiran and Pallavi Patel Global University (KPGU)**

**Under the Supervision of:**

**Piyush Sharma**



**ABSTRACT**

The **ATM Simulator** is a software application designed to manage an individual's bank account efficiently. This project aims to simulate the core functionalities of a banking account system and provide a simple yet effective solution for basic banking operations. The primary goal is to assist users in performing essential financial transactions in a secure and user-friendly environment.

The project was developed to address the limitations of manual banking systems, offering an automated solution that simplifies banking tasks such as balance inquiries, cash withdrawals, deposits, and account management. By providing these services digitally, the system improves transaction speed, accuracy, and user convenience.

This simulator goes beyond conventional banking applications by integrating additional functionalities that enhance the user's experience. It serves as a step toward modernizing banking services, making them more accessible and efficient.

The project is built using the **Java programming language**, employing relevant technologies to ensure system reliability and performance. One of the key challenges in software development, especially in IT and product development, is creating and managing clear requirements. Proper requirement management is essential for meeting customer expectations, ensuring compliance, maintaining timelines, and staying within budget.

|  |  |
| --- | --- |
| **List of Figures** Figure 1: activity Diagram……………………………………………………………...  Figure 2:Class Diagram…..……………………………………………………………  **Table of Contents**  **Chapter 1 Overview of project** ............................................... ..............................................  1.1overview ..............................................................................................................................  1.2 Scope of Work.....................................................................................................................  1.3 Organization of Project ......................................................................................................  1.4 Capacity of Plant ..............................................................................................................  **Chapter 2 Overview and Layout of the Project**…..............................................................  2.1 detail of work…… …..........................................................................................................  2.2technical symphony……...…………………………………………………………….…  2.2 Stages of Production..........................................................................................................  **Chapter 3 Introduction of Project**.......................................................................................  3.1Project Summary................................................................................................................  3.2 Purpose .............................................................................................................................  3.3 Objective ...........................................................................................................................  3.4 Scope ……………………………………………………………………………….…  3.5 Technology………………………………………………………………………. …...  **Chapter 4 System Analysis**…………………….………………….……………………....  4.1 Study of current system …….……………………………………….………………….  4.2 Problems and Weakness………………………………………………….……………… | |
| 4.3 Requirements of a New System……………………………………….…………………  4.4 System Feasibility……………………………………….…………….…………………  4.5 Features of New System………………………………………………………… ……..  4.6 Tools/ Materials………………………………………………………………...………...  **Chapter 5 System Design**……………………………………………….………………….  5.1 System Design and Methodology………………………………………………………...  5.2 Structure Design………………………………………………………….……………....  5.3Activity Diagram…………………………………………………………………………  5.4 Class Diagram…………………………………………………………………………….  **Chapter 6 Implementation**……………………………………………………………...….  6.1 module specification……………….………………………………………………….….  6.2 Testing Results / Experiment…………………………………...........................................  **Chapter 7 Conclusion and Discussion**……………………………………...........................  7.1 Overall Analysis of Project Viabilities………………………………….…………….......  7.2 Summary………………………………………………………………….........................  7.3 Limitations and Future Enhancement………………………………………………...….  7.4 Conclusion….……………………………………………………………….………...….  **Reference**………………………………………………………………………………...…. | |

## **1. Overview of the Project**

### **1.1 Overview**

The Simple banking application is a user-friendly financial management tool designed to provide customers with a seamless banking experience. Developed with a focus on simplicity and ease of use, the application offers a range of features to help users manage their finances effectively.

* **Account Management**

Users can easily view their account balances, transaction history, and account details.

Transfer funds between accounts within the same bank or to external accounts.

Set up recurring transfers for bills and savings.

* **Mobile Deposits**

deposit status and receive instant notifications.

* **Bill Payment**

Pay bills directly from the app, scheduling one-time or recurring payments.

Receive reminders for upcoming bills to avoid late payments.

* **Budgeting Tools**

Categorize expenses to track spending habits.

Set budget limits for different categories and receive alerts when approaching limits.

* **Customer Support**

Access customer support via chat within the app for quick assistance.

Report lost or stolen cards and request replacements.

* **Security**

Two-factor authentication for login and transactions.

Ability to lock or unlock cards instantly in case of suspicious activity.

Touch ID/Face ID support for secure and convenient access.

### **1.2 Scope of Work**

The main scope of work is to designing and developing this Internet banking System Java primarily based Engineering project is to provide secure and efficient net banking facilities to the banking customers over the internet. Apache Server Pages, MYSQL database used to develop this bank application where all banking customers can login through the secured web page by their account login id and password. Users will have all options and features in that application like get money from western union, money transfer to others, and send cash or money to inter banking as well as other banking customers by simply adding them as payees.

**Capabilities:-**

* Management: Users can create, view, and manage their accounts.
* Balance Inquiry: Users can check their account balances in real-time.
* Transaction History: Users can view a history of their transactions, including deposits, withdrawals, and transfers.
* Fund Transfers: Users can transfer funds between their own accounts or to other accounts within the same bank.

**Limitations :-**

* Functionality: Simple ATM Stimulator may lack advanced features such as investment management, financial planning tools, or integration with third-party services.
* Security Concerns: Basic banking apps may have weaker security measures compared to more robust platforms, making them more vulnerable to cyber threats.
* Limited Accessibility: Some users may find it challenging to access the application if it's not available on multiple platforms (e.g., mobile, web) or if it lacks compatibility with older devices.
* Scalability Limited.
  1. **Organization Chart:**

The ATM Stimulator project involved creating a simulated ATM (Automated Teller Machine) system for training and educational purposes. This simulator mimicked the functions of a real ATM, allowing users to practice transactions like withdrawing, depositing, and checking balances in a safe environment. It aimed to provide a realistic experience without the risk of using actual money or compromising security

* 1. **Capacity of Plant :**

The Atm simulator boasts a robust capacity that caters to a wide array of financial needs. From basic functions like checking balances and transferring funds to more advanced features such as bill payments and account management, it offers a comprehensive suite of services. This capacity extends to its user-friendly interface, making navigation intuitive for users of all levels. Moreover, the application's security measures ensure that transactions are safe and protected.

## **2. Overview and Layout of the Project**

### **2.1 Details of Work**

A simple banking application typically involves several key features and functionalities that allow users to manage their finances. Here's an outline of the basic components and their functionalities:

**User Authentication**:

Login/Logout: Users should be able to securely log in to their accounts using credentials like username/email and password.

**Registration:** New users should have the ability to create an account by providing necessary information and verifying their identity.

**Account Management**:

View Balances: Users can check the balances of their accounts (savings, checking, etc.). Transaction History: View a list of recent transactions, including deposits, withdrawals, transfers, and payments.

Account Statements: Ability to generate and download account statements for a specified period.

**Account Details**:

View and edit account details such as name, contact information, etc..

**Fund Transfers:**

Scheduled Transfers: Set up recurring transfers for bills or savings.

**Deposits and Withdrawals**:

Deposit Funds: Make deposits to accounts via mobile check deposit, ACH transfer, or in-person deposit.

Withdraw Funds: Withdraw money from accounts through ATM withdrawals or in-branch withdrawals.

**Password Reset:**

Allow users to securely reset their passwords if forgotten. Session Management: Automatically log out users after a period of inactivity.

**Technologies Used:**

Frontend: java Swing

Backend: java programming language

Database: MySQL.

**2.2 Technical Symphony:**

Creating a technical symphony in simple banking applications using Java and MySQL involves designing a robust system with clear functionality. In Java, classes like Account, Transaction, and Bank can be created to model the entities. The Account class manages customer account details, with attributes such as account number, balance, and owner information. Transaction class handles deposit, withdrawal, and transfer functionalities, ensuring data integrity. The Bank class acts as a controller, orchestrating interactions between accounts and transactions. MySQL is utilized for data storage, with tables for accounts and transactions, maintaining a normalized database structure. Java's JDBC API facilitates connectivity, allowing the application to execute SQL queries for CRUD operations. Error handling, security measures like encryption, and multi-threading for concurrent operations enhance the system's reliability and performance. This symphony of Java and MySQL harmonizes data management and user interactions, creating a seamless banking experience.

### **2.3 Stages of Production**

The production stage of a Atm simulator Application typically involves several key steps:

**Requirement Gathering**: Understand the requirements of the application, including features, user roles, security requirements, and integration needs.

**Design**: Create a design for the application, including the user interface (UI) design, database schema, and system architecture. This phase may also involve prototyping and wireframing. **Development**: Write the code for the application based on the design. This involves front-end development for the user interface, back-end development for server-side logic, and database development for data storage and retrieval.

**Testing**: Test the application to ensure that it functions correctly and meets the requirements. This includes unit testing, integration testing, and user acceptance testing.

**Deployment:** Deploy the application to a production environment where it can be accessed by users. This may involve setting up servers, configuring databases, and implementing security **Monitoring and Maintenance:** Continuously monitor the application for issues such as bugs, performance issues, and security vulnerabilities. Regular maintenance may be required to apply updates, patches, and new features

## **3. Introduction to Project**

### **3.1 Project Summary**

The “Simple ATM Application “project is a model Internet Banking Site. This site enables the customers to perform the basic banking transactions by sitting at their office or at homes through PC or laptop. The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. The customers can access the banks website for viewing their Account details and perform the transactions on account as per their requirements. With Internet Banking, the brick and mortar structure of the traditional banking gets converted into a click and portal model, thereby giving a concept of virtual banking a real shape. Thus, today’s banking is no longer confined to branches. E- banking facilitates banking transactions by customers round the clock globally.

The primary aim of this “Simple ATM Application “is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for core sector like banking. This necessitates the design to be expandable and modifiable and so a modular approach is used in developing the application software.

Anybody who is an Account holder in this bank can become a member of Banking Application. He has to fill a form with his personal details and Account Number. Bank is the place where customers feel the sense of safety for their property. In the bank, customers deposit and withdraw their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the positive need for management of the bank, which can handle all this with comfort and ease. Smooth and efficient management affects the satisfaction of the customers and staff members, indirectly. And of course, it encourages management committee in taking some needed decision for future enhancement of the bank.

### **3.2 Purpose**

The Traditional way of maintaining details of user in a bank was to enter the details and record them. Every time the user needs to perform some transactions he has to go to bank and perform necessary actions, which may not be so feasible all the time. It may be a hard-hitting task for users and the bankers too. The project gives real life understanding of ATM Stimulator and activities performed by various roles in the supply chain. Here, we provide automation for banking system through Internet.

Online Banking System project captures activities performed by different roles in real life banking which provides enhanced techniques for maintaining the required information up-to-date, which results in efficiency. The project gives real life understanding of Online Banking System and activities performed by various roles in the supply chain.

Main Goal :

1. Motto: Our motto is to develop a software program for managing entire bank process related to Administration accounts customer accountsand to keep each every track about their property and their various transaction processes efficiently. Hereby, our main objective is the  customer’s satisfaction considering today’s faster in the world.
2. Customer Satisfaction: Client can do his operations comfortably without any risk or losing of his privacy. Our software will perform and fulfil all the tasks that any customer would desire.
3. Saving Customer Time :Client doesn't need to go to the bank to do small operation.

Protecting the Customer: It helps the customer to be satisfied and comfortable in his choices, this protection contains customer’s account, money and his privacy.

1. Transferring Money: Help client transferring money to/or another bank or country.

### **3.3 Objective**

The objective of a simple banking application is to provide basic banking functionalities to users in a convenient and user-friendly manner. This typically includes features such as account balance checking, fund transfers between accounts, bill payments, transaction history viewing, and possibly basic account management tasks like updating personal information. The primary goal is to allow users to perform essential banking tasks efficiently and securely from their mobile devices or computers.

### **3.4 Scope**

The main scope of work is to designing and developing this Internet atm simulator System Java primarily based Engineering project is to provide secure and efficient net banking facilities to the banking customers over the internet. Apache Server Pages, MYSQL database used to develop this bank application where all banking customers can login through the secured web page by their account login id and password. Users will have all options and features in that application like get money from western union, money transfer to others, and send cash or money to inter banking as well as other banking customers by simply adding them as payees.

**Capabilities:-**

* Management: Users can create, view, and manage their accounts.
* Balance Inquiry: Users can check their account balances in real-time.
* Transaction History: Users can view a history of their transactions, including deposits, withdrawals, and transfers.
* Fund Transfers: Users can transfer funds between their own accounts or to other accounts within the same bank.

**Limitations :-**

* Functionality: Simple ATM Stimulator may lack advanced features such as investment management, financial planning tools, or integration with third-party services.
* Security Concerns: Basic banking apps may have weaker security measures compared to more robust platforms, making them more vulnerable to cyber threats.
* Limited Accessibility: Some users may find it challenging to access the application if it's not available on multiple platforms (e.g., mobile, web) or if it lacks compatibility with older devices.
* Scalability Limited.

### **3.5 Technology**

* Laptop/Pc
* Memory : 8 GB RAM
* Processor : 11th Gen Intel Core i5
* Operating System : Windows 11

## **4. System Analysis**

**4.1 Study of Current System :**

An in-depth examination of prevailing ATM stimulator systems reveals significant inefficiencies and challenges plaguing conventional approaches. These include:

* Account Management: Users can create accounts, view balances, and manage transactions like deposits and withdrawals.
* Transfers: This feature allows users to transfer money between their own accounts or to other users within the same bank.
* Bill Payments: Users can pay bills directly from their accounts, such as utility bills, credit card bills, and more.
* Mobile Deposits: Users can deposit checks by taking pictures of them with their mobile devices, eliminating the need to visit a branch..
* Security: Implementations of security features like two-factor authentication, encryption, and biometric logins for safe and secure transactions.
* Account Statements: Users can view and download their account statements for a specific period.

**4.2Problem and Weaknesses of Current System :**

**Limiited Features**: Simple banking apps often lack advanced features like investment tracking or financial analysis tools.

**Security Risks:** They may have vulnerabilities, making them targets for cyber-attacks and data breaches.

**Poor User Experience**: Basic apps can be clunky, with unintuitive interfaces that frustrate users.

**Lack of Personalization**: These apps may not offer

### **4.3 Requirements of a New System**

Creating a simple banking application involves several key requirements to ensure it functions smoothly and securely. Here are some common requirements for a new system in a simple banking application:

**Security**:

Implement robust security measures to protect user data and transactions.

Use encryption for sensitive data transmission.

Regularly update security protocols to prevent fraud and cyber-attacks.

**Customer Support**:

Provide channels for customer support such as live chat, email support, and phone support.

Include a help section or FAQ within the application for common queries.

**Accessibility and Usability**:

Ensure the application is user-friendly and accessible to all users, including those with disabilities.

Design a responsive interface for both web and mobile platforms.

**Compliance:**

Adhere to banking regulations and compliance standards (e.g., KYC - Know Your Customer, AML - Anti-Money Laundering).

Maintain audit logs for compliance and regulatory purposes.

**Integration:**

Integrate with external systems such as payment gateways, credit bureaus, and third-party financial services if needed.

### **4.4 System Feasibility**

This feasibility study aims to assess the viability and practicality of developing a Simple Banking Application. The application is designed to provide basic banking functionalities to users, including account management, fund transfers, transaction history, and account balance checks. This study will delve into the technical, operational, and economic feasibility of such a system.

1. **Technical Feasibility**

Hardware Requirements: The Simple Banking Application will require minimal hardware resources, such as servers to host the application and databases to store user information. Software Requirements: The application will be developed using modern programming languages and frameworks suitable for web and mobile platforms. Compatibility with popular browsers and mobile operating systems will be ensured.

Security Measures: Robust security measures will be implemented to protect user data and transactions. This includes encryption protocols, secure authentication methods, and regular security updates.

Scalability: The system will be designed to scale as the user base grows, ensuring that it can handle increased loads without significant performance degradation.

1. **Operational Feasibility**

User-Friendly Interface: The application will have an intuitive and user-friendly interface, making it easy for users to navigate and perform banking tasks without extensive training. Training: Minimal training will be required for bank staff to assist customers with the application. Online guides and tutorials will also be provided for users.

Customer Support: A dedicated customer support system will be established to address any issues or queries users may have while using the application.

1. **Economic Feasibility**

Development Costs: The initial development costs will include software development, server setup, security implementation, and testing. A detailed budget will be prepared to estimate these costs accurately.

Operating Costs: Ongoing costs will include maintenance, server hosting fees, customer support, and marketing. These costs will be offset by transaction fees and potential revenue streams from premium services.

Return on Investment (ROI): The ROI will be analyzed based on projected user adoption rates, transaction volumes, and potential upsell opportunities for additional banking products.

### **4.5 Features of New System**

1. User-Friendly Interface
2. Secure Authentication:
3. Notification
4. Customer Support
5. Security

.

### **4.6 TOOLS / MATERIALS**

**Integrated Development Environment (IDE)**:

Recommended: IntelliJ IDEA, Eclipse, NetBeans, or any Java IDE you are comfortable with.

**Java Development Kit (JDK):**

Ensure JDK is installed on your system (Java 8 or newer recommended).

**MySQL Database:**

Install MySQL Server for storing account information, transactions, etc.

**MySQL Workbench** (Optional):

GUI tool for managing and interacting with MySQL databases.

Java Libraries:

**JDBC (Java Database Connectivity):**

Included in JDK.

Allows Java programs to interact with databases like MySQL.

JavaFX (for GUI, optional but recommended for a graphical interface):

Included in Java 8 and later versions.

Used for building the graphical user interface.

**MySQL Database Design:**

Tables: accounts, transactions, etc.

Design columns based on your application needs (e.g., account number, balance, transaction amount, timestamps).

**Database Setup:**

Create a MySQL database. Design tables (accounts, transactions).

## **5. System Design**

### **5.1 System Design and Methodology**

A simple banking application involves several components, including the user interface, backend servers, databases, and security measures.

1. User Interface (UI)

Account Dashboard: Allows users to view balances, transaction history, and perform basic banking tasks.

Transfer: Interface for transferring money between accounts.

Payments: Facilitates bill payments and other transactions.

Settings: Where users can manage their account details, security settings, etc.

1. Backend Servers

Authentication Server: Handles user login, registration, and authentication.

Account Management Server: Manages user accounts, balances, transactions, etc.

Transaction Server: Processes and records transactions.

Notification Server: Sends alerts and notifications to users.

1. Databases

User Database: Stores user information (name, email, password hashes, etc.).

Account Database: Holds account details (account numbers, balances, transaction history).

Transaction Log: Records all transactions for auditing and history.

Methodology

1. Requirements Gathering

Identify what the application needs to do (features, security, compliance).

Understand user needs (user stories, feedback).

1. Design

High-Level Design: Define system architecture, components, and how they interact.

Detailed Design: Specify data structures, algorithms, and interfaces.

1. Development

Frontend Development: Implement the user interface.

Backend Development: Create server-side logic, APIs, and database integration.

Testing: Conduct unit testing, integration testing, and user acceptance testing.

1. Security Measures

Encryption: Encrypt sensitive data in transit (HTTPS) and at rest (database). Authentication:

Secure login with strong password requirements, multi-factor authentication.

Authorization: Role-based access control (users, admins, tellers).

Auditing: Log all transactions and access for auditing purposes.

Firewalls & Monitoring: Protect servers with firewalls, monitor for suspicious activity.

Backend: User login details checked against User Database -> Account balance checked/updated in Account Database -> Transaction recorded in Transaction Log.

By embracing Agile Scrum methodology, the Smart Waste Management System can navigate the complexities of waste management challenges with agility and resilience.

### **5.2 Structure Design**

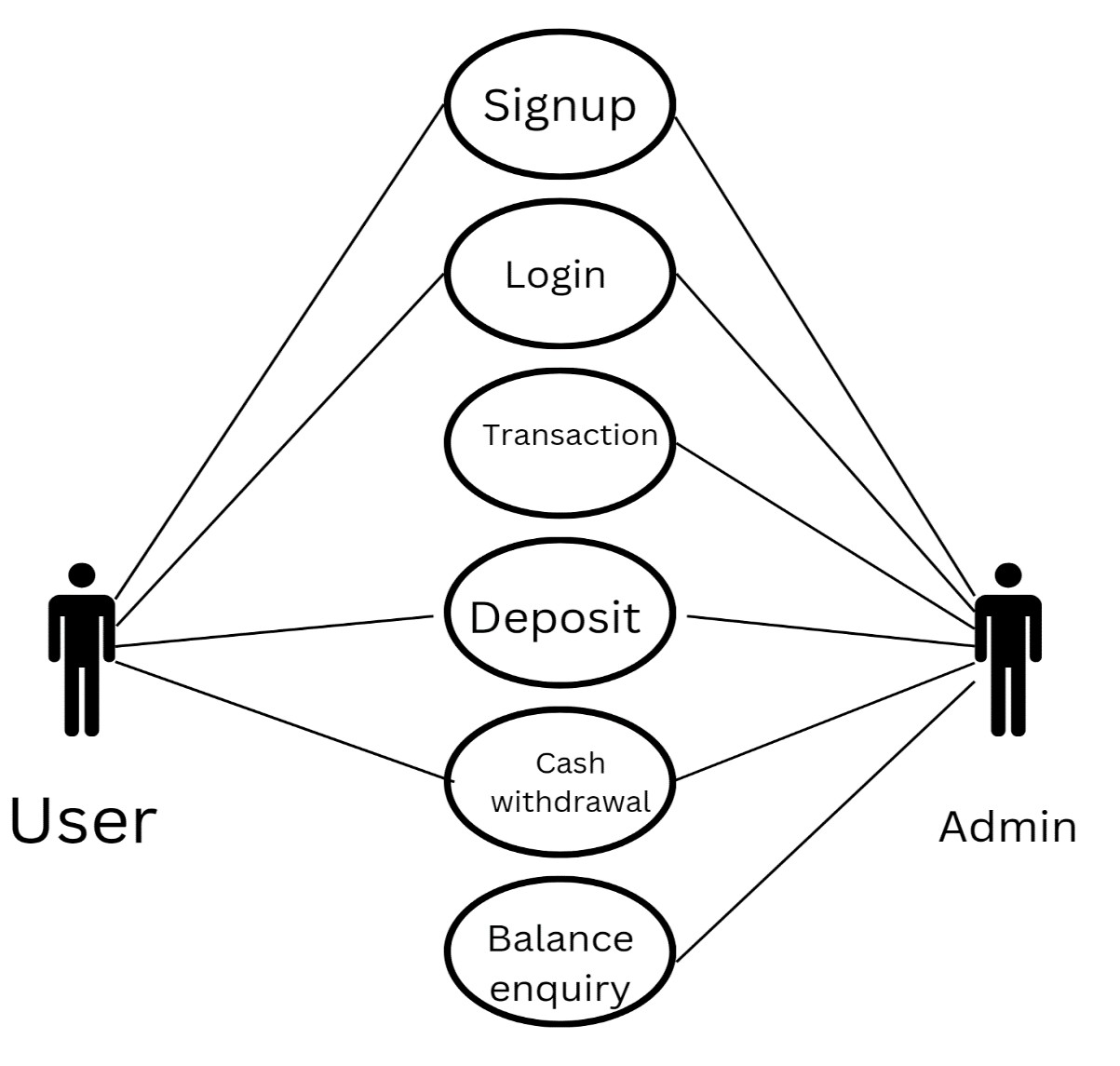
The structure design for a atm simulator project typically involves several key components. At the core is the user interface (UI), which includes screens for login, account overview, transactions, and settings. The UI communicates with a backend server through APIs, which handle the business logic and data management.

The backend server is responsible for user authentication, account management, transaction processing, and interfacing with the database. A robust authentication system ensures secure access to accounts, with features like two-factor authentication for added security.

The database stores user information, account details, transaction history, and other relevant data. It should be designed for efficiency and scalability, utilizing technologies like SQL or NoSQL depending on the requirements.

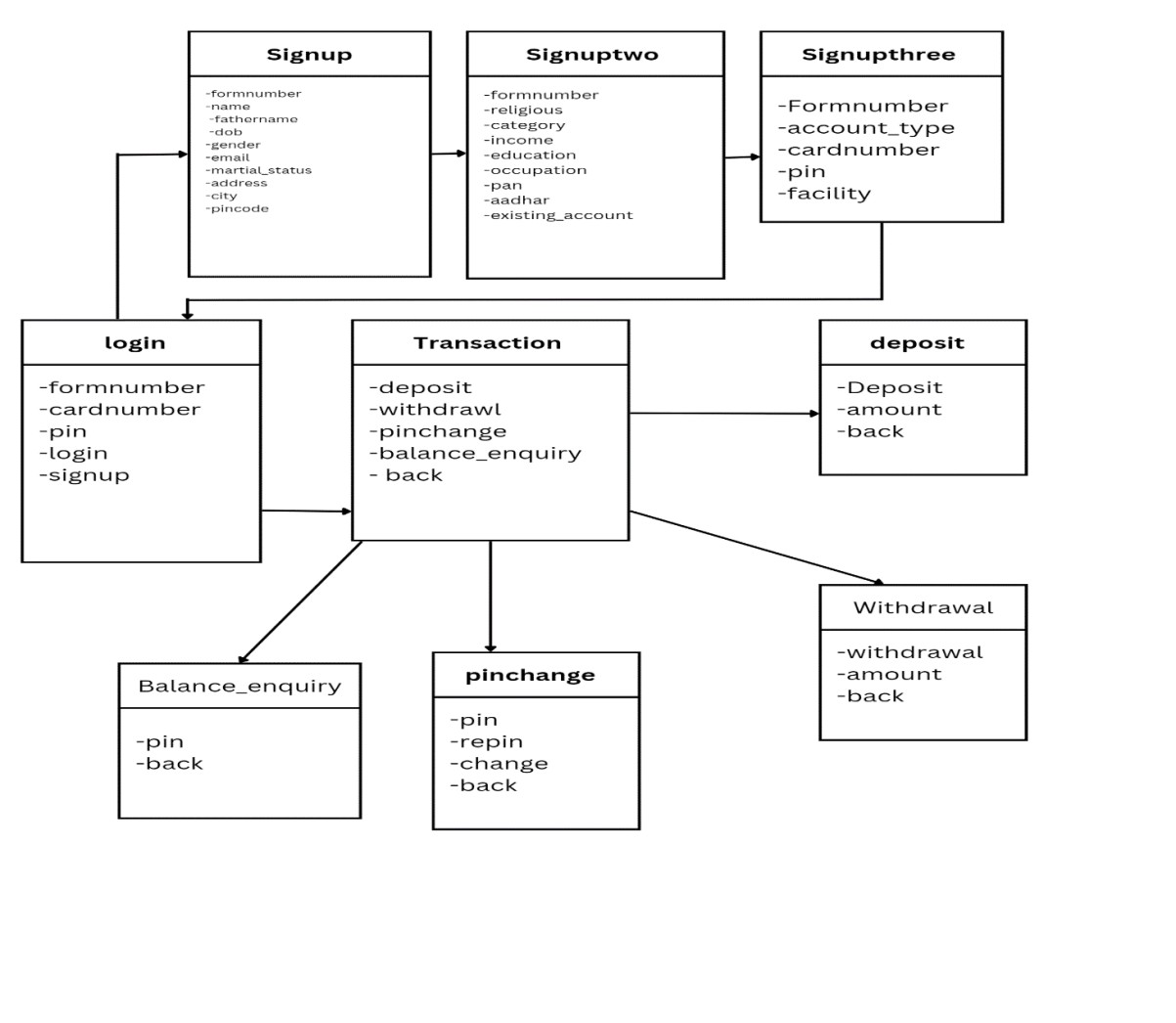
Additionally, error handling and logging are crucial for maintaining the application's stability and troubleshooting. Properly implemented, they provide valuable insights into system behavior and user interactions.

### **5.4 Activity Diagram**



*Figure 4: Sequency Diagram*

**CLASS DIAGRAM:**



**6.2 Program / Modules Specification(s) :**

Login page :-

package bankmangment;

import java.awt.Color;

import java.awt.Font;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.ResultSet;

import javax.swing.JPasswordField;

import javax.swing.\*;

    public  class login extends JFrame implements ActionListener {

        JButton login,signup,clear;

        JTextField pinTextField, cardTextField;

        JPasswordField pinTeextField;

        login()

        {

         setTitle("autometed teleer machine");

setLayout(null);

ImageIcon I1 =new ImageIcon(ClassLoader.getSystemResource("images.png")) ;

        Image I2 = I1.getImage().getScaledInstance(100,100, Image.SCALE\_DEFAULT);

           ImageIcon I3 =new ImageIcon(I2);

           JLabel label = new JLabel(I3);

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

           add(label);

           JLabel text = new JLabel("Welcome to ATM");

           text.setFont(new Font("osward", Font.BOLD, 38));

           text.setBounds(200, 40, 400, 40);

           add(text);

           JLabel cardno = new JLabel("Card no:");

           cardno.setFont(new Font("railway", Font.BOLD, 28));

           cardno.setBounds(120, 150, 150, 30);

           add(cardno);

           cardTextField = new JTextField();

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

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

           add(cardTextField);

           JLabel pinno = new JLabel("PIN:");

           pinno.setFont(new Font("railway", Font.BOLD, 28));

           pinno.setBounds(120, 220, 400, 40);

           add(pinno);

            pinTextField = new JPasswordField();

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

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

           add(pinTextField);

           login =new JButton("SIGN IN");

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

           login.setBackground(Color.BLACK);

           login.setForeground(Color.WHITE);

           login.addActionListener(this);

           add(login);

           clear =new JButton("CLEAR");

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

           clear.setBackground(Color.BLACK);

           clear.setForeground(Color.WHITE);

           clear.addActionListener(this);

           add(clear);

           signup =new JButton("SIGN UP");

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

           signup.setBackground(Color.BLACK);

           signup.setForeground(Color.WHITE);

           signup.addActionListener(this);

           add(signup);

           getContentPane().setBackground(Color.GRAY);

            setSize(800,480);

            setVisible(true);

            setLocation(350,200);

        }

        public  static void main(String args[])

        {

            new login();

        }

        @Override

        public void actionPerformed(ActionEvent e) {

            if(e.getSource() == clear)

            {   cardTextField.setText("");

                pinTextField.setText("");

            } else if(e.getSource() == login)

            {

                conn cc = new conn();

                String cardno = cardTextField.getText();

                String pinno = pinTextField.getText();

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

                try {

                  ResultSet rs =  cc.s.executeQuery(query);

                  if (rs.next()) {

                        setVisible(false);

                        new Transaction(pinno).setVisible(true);

                  }

                  else{

                    JOptionPane.showMessageDialog(null,"Incorrect cardNumber or pin");

                  }

                } catch (Exception ex) {

                    System.out.println(ex);

                }

            } else if(e.getSource() == signup){

                    setVisible(false);

                    new signupone().setVisible(true);

            }

        }

}

Transaction page :-

package bankmangment;

import java.awt.Color;

import java.awt.Font;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

public class Transaction extends JFrame  implements ActionListener {

  JButtondeposite,withdrawl,miniStatement,pinchange,fastcash,balanceenquiery,exit;

  String pinno;

Transaction(String pinno){

  this.pinno=pinno;

   setLayout(null);

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

Image i2 = i1.getImage().getScaledInstance(900, 900, Image.SCALE\_DEFAULT);

ImageIcon i3= new ImageIcon(i2);

JLabel image =  new JLabel(i3);

image.setBounds(0, 0, 900, 900);

add(image);

JLabel text = new JLabel("Please select your Transaction");

text.setBounds(210,300,700,35);

text.setFont(new Font("system", Font.BOLD, 16));

text.setForeground(Color.WHITE);

image.add( text );

        deposite = new JButton("Deposite");

        deposite.setBounds(170, 415, 150, 30);

        deposite.addActionListener(this);

        image.add(deposite);

        withdrawl = new JButton("Cash Withdrawl");

        withdrawl.setBounds(355, 415, 150, 30);

        withdrawl.addActionListener(this);

        image.add(withdrawl);

        fastcash = new JButton("Fast Cash");

        fastcash.setBounds(170, 450, 150, 30);

        fastcash.addActionListener(this);

        image.add(fastcash);

        miniStatement = new JButton("Mini Statement");

        miniStatement.setBounds(355, 450, 150, 30);

        miniStatement.addActionListener(this);

        image.add(miniStatement);

        pinchange = new JButton("Pin Change");

        pinchange.setBounds(170, 485, 150, 30);

        pinchange.addActionListener(this);

        image.add(pinchange);

        balanceenquiery= new JButton("Balance Enquiry");

        balanceenquiery.setBounds(355, 485, 150, 30);

        balanceenquiery.addActionListener(this);

        image.add(balanceenquiery);

        exit = new JButton("Exit");

        exit.setBounds(355, 520, 150, 30);

        exit.addActionListener(this);

        image.add(exit);

setSize(900, 900);

        setLocation(300, 0);

        setUndecorated(true);

        setVisible(true);

    }

        public static void main(String[] args) {

            new Transaction("");

          }

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == exit)

            {

                System.exit(0);

            }else if (e.getSource() == deposite) {

                setVisible(false);

                new Deposit(pinno).setVisible(true);

            } else if (e.getSource() == withdrawl) {

                setVisible(false);

                new withdrawl(pinno).setVisible(true);

            }else if (e.getSource()  == fastcash )

            {

                setVisible(false);

                new fastcash(pinno).setVisible(true);

            }  else if (e.getSource() == pinchange) {

                setVisible(false);

                new PINchange(pinno).setVisible(true);

            }  else if (e.getSource() == balanceenquiery) {

                setVisible(false);

                new BalanceEnquiry(pinno).setVisible(true);

            } else if (e.getSource() == miniStatement){

                new MiniStatement(pinno).setVisible(true);

            }

        }

}

Signup page :-

package bankmangment;

import java.awt.Color;

import java.awt.Font;

import java.util.Random;

import javax.swing.\*;

import com.toedter.calendar.JDateChooser;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class signupone extends JFrame implements ActionListener

{ long Random;

  JTextField  nameTextFeild,fnameTextFeild,emailTextFeild,addressTextFeild,cityTextFeild,stateTextFeild,pinTextFeild ;

    JButton next;

    JRadioButton male ,female,other ,married,unmarried;

    JDateChooser dateChooser ;

signupone()

    {

        setLayout(null);

            Random ran =new Random();

            Random = Math.abs((ran.nextLong() % 9000L ) + 1000L );

             JLabel formno = new JLabel("APPLICATION FORM NO." + Random);

                formno.setFont(new Font("raleway", Font.BOLD, 38));

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

                add(formno);

                JLabel personDetail = new JLabel("page 1: person details" );

                personDetail.setFont(new Font("raleway", Font.BOLD, 22));

                personDetail.setBounds(290, 80, 400, 30);

                add(personDetail);

                JLabel name = new JLabel( "Name :");

                name.setFont(new Font ("raleway ",Font.BOLD,20));

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

                add(name);

                nameTextFeild = new JTextField();

                nameTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(nameTextFeild);

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

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

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

                add(fname);

                fnameTextFeild = new JTextField();

                fnameTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(fnameTextFeild);

                JLabel Dob = new JLabel("Date Of Birth:");

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

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

                add(Dob);

                dateChooser = new JDateChooser();

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

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

                add(dateChooser);

                JLabel gender = new JLabel("Gender:");

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

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

                add(gender);

                male = new JRadioButton("Male");

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

                male.setBackground(Color.WHITE);

                add(male);

                 female =new JRadioButton("Female");

                female.setBounds(450,290,120,30);

                female.setBackground(Color.WHITE);

                add(female);

                ButtonGroup gendeGroup =new ButtonGroup();

                gendeGroup.add(male);

                gendeGroup.add(female);

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

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

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

                add(email);

                  emailTextFeild = new JTextField();

                emailTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(emailTextFeild);

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

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

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

                add(marital);

                 married = new JRadioButton("married");

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

                married.setBackground(Color.WHITE);

                add(married);

                 unmarried =new JRadioButton("unmarried");

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

                unmarried.setBackground(Color.WHITE);

                add(unmarried);

                 other =new JRadioButton("other");

                other.setBounds(630,390,100,30);

                other.setBackground(Color.WHITE);

                add(other);

                ButtonGroup maritalGroup=new ButtonGroup();

                maritalGroup.add(married);

                maritalGroup.add(unmarried);

                maritalGroup.add(other);

                JLabel address = new JLabel("Address:");

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

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

                add(address);

                 addressTextFeild = new JTextField();

                addressTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(addressTextFeild);

                JLabel city = new JLabel("City:");

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

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

                add(city);

                cityTextFeild = new JTextField();

                cityTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(cityTextFeild);

                JLabel state = new JLabel("State:");

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

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

                add(state);

                stateTextFeild = new JTextField();

                stateTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(stateTextFeild);

                JLabel pincode = new JLabel("Pincode:");

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

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

                add(pincode);

                pinTextFeild = new JTextField();

                pinTextFeild.setFont(new Font("raleway", Font.BOLD, 14));

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

                add(pinTextFeild);

            next = new JButton("Next");

            next.setBackground(Color.BLACK);

            next.setForeground(Color.WHITE);

            next.setFont(new Font("raleway", Font.BOLD, 14));

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

            next.addActionListener(this);

            add(next);

        getContentPane().setBackground(Color.LIGHT\_GRAY);

setSize(850,800);

        setLocation(350,10);

        setVisible(true);

    }

    public void actionPerformed(ActionEvent ae) {

       String formno = ""+Random;//long

       String name = nameTextFeild.getText();

       String fname = fnameTextFeild.getText();

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

       String gender = null;

       if(male.isSelected())

       {   gender = "male";}

       else if (female.isSelected())

       {   gender = "female";}

       String email = emailTextFeild.getText();

       String marital = null;

       if (married.isSelected())

       {marital = "married";}

else if (unmarried.isSelected())

       { marital = "unmarried";}

else if (other.isSelected())

       {   marital = "other";}

       String address = addressTextFeild.getText();

       String city = cityTextFeild.getText();

       String state = stateTextFeild.getText();

       String pin = pinTextFeild.getText();

 try{  if(name.equals(""))

           { JOptionPane.showMessageDialog(null,"Name is reqired");}

        else  {      conn c =new conn();

                      String query = "insert into signup values('"+formno+"','"+name+"','"+fname+"','"+Dob+"','"+gender+"','"+email+"','"+marital+"','"+address+"','"+city+"','"+pin+"','"+state+"')";

                            c.s.execute(query);

                            System.out.println(query);

                            setVisible(false);

                            new signuptwo(formno).setVisible(true);

                        }}

      catch (Exception e) {

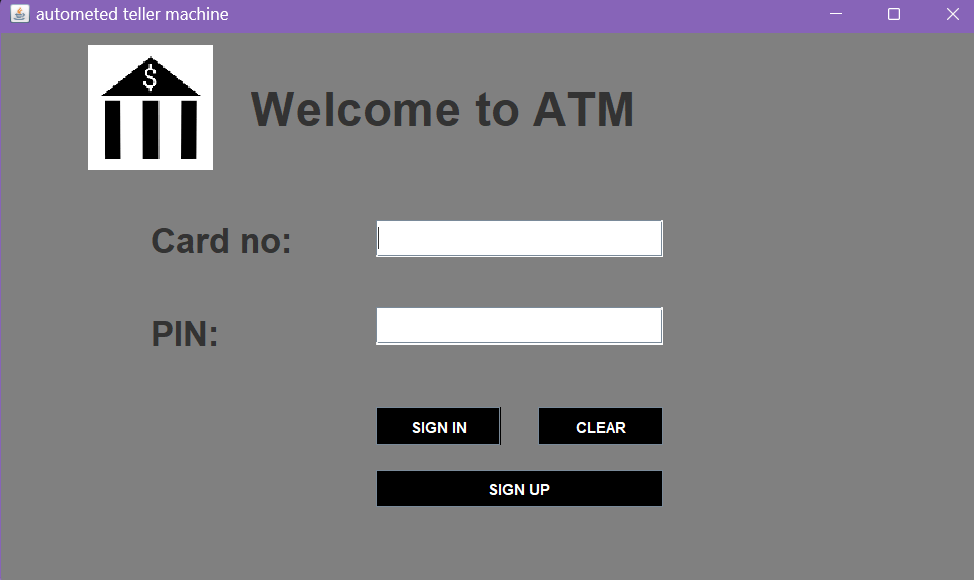
            System.out.println(e);  }}

public static void main(String[] args) {

     new signupone(); }

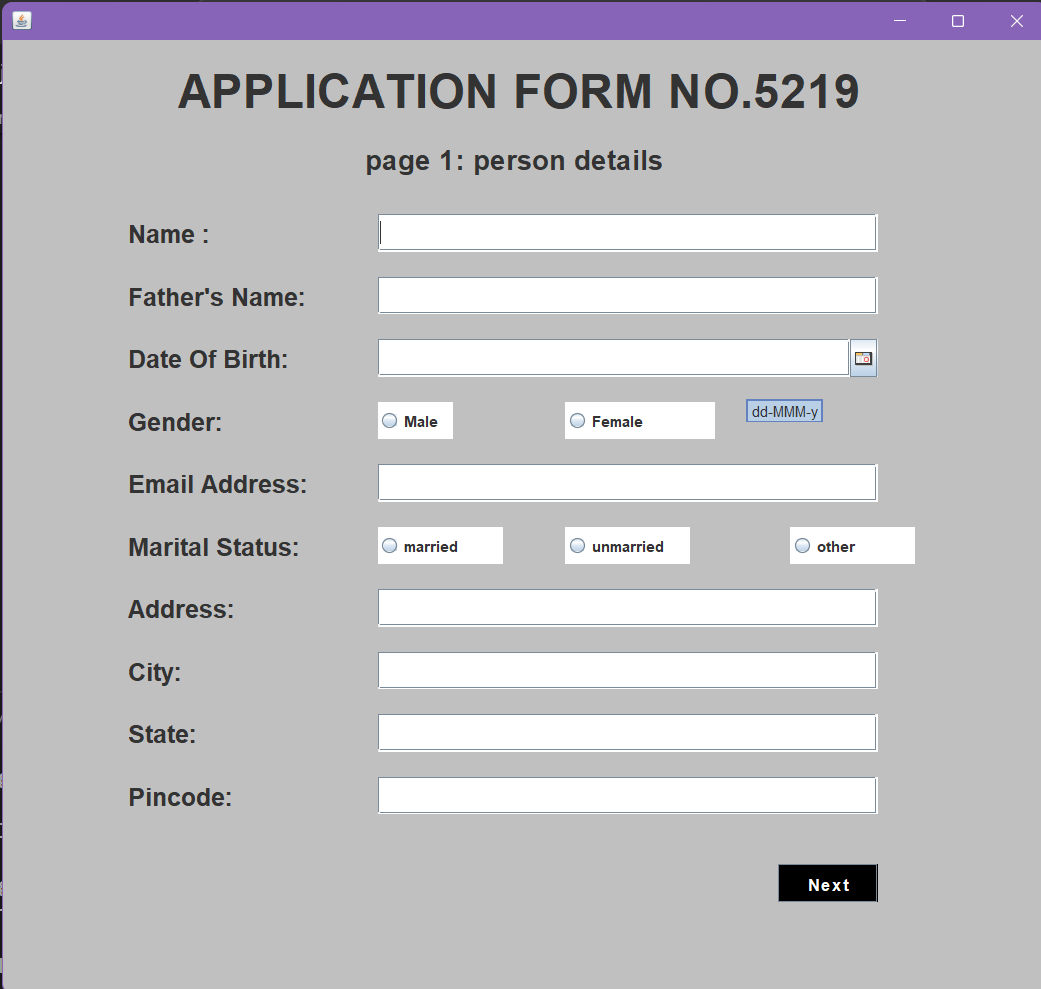
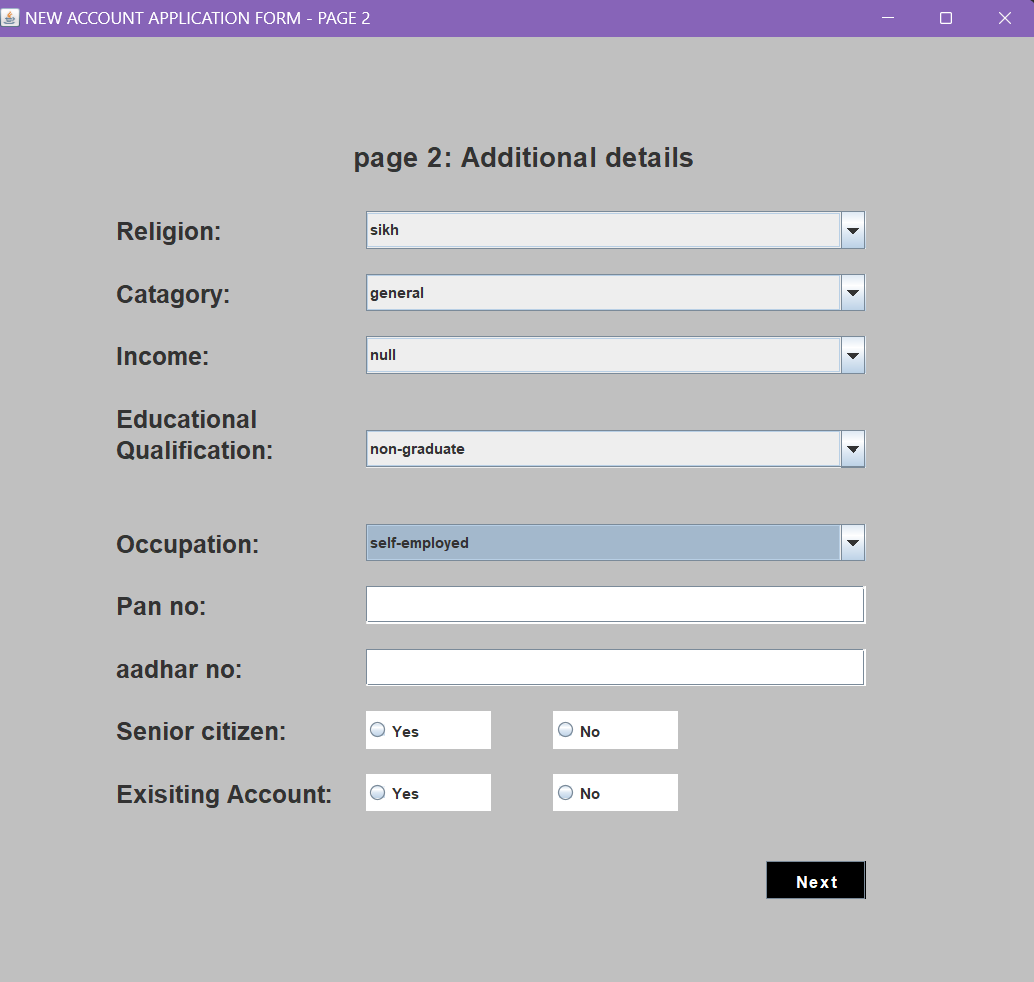
**Testing (Screenshot – Result amd Analysis)**

**Login page :-**

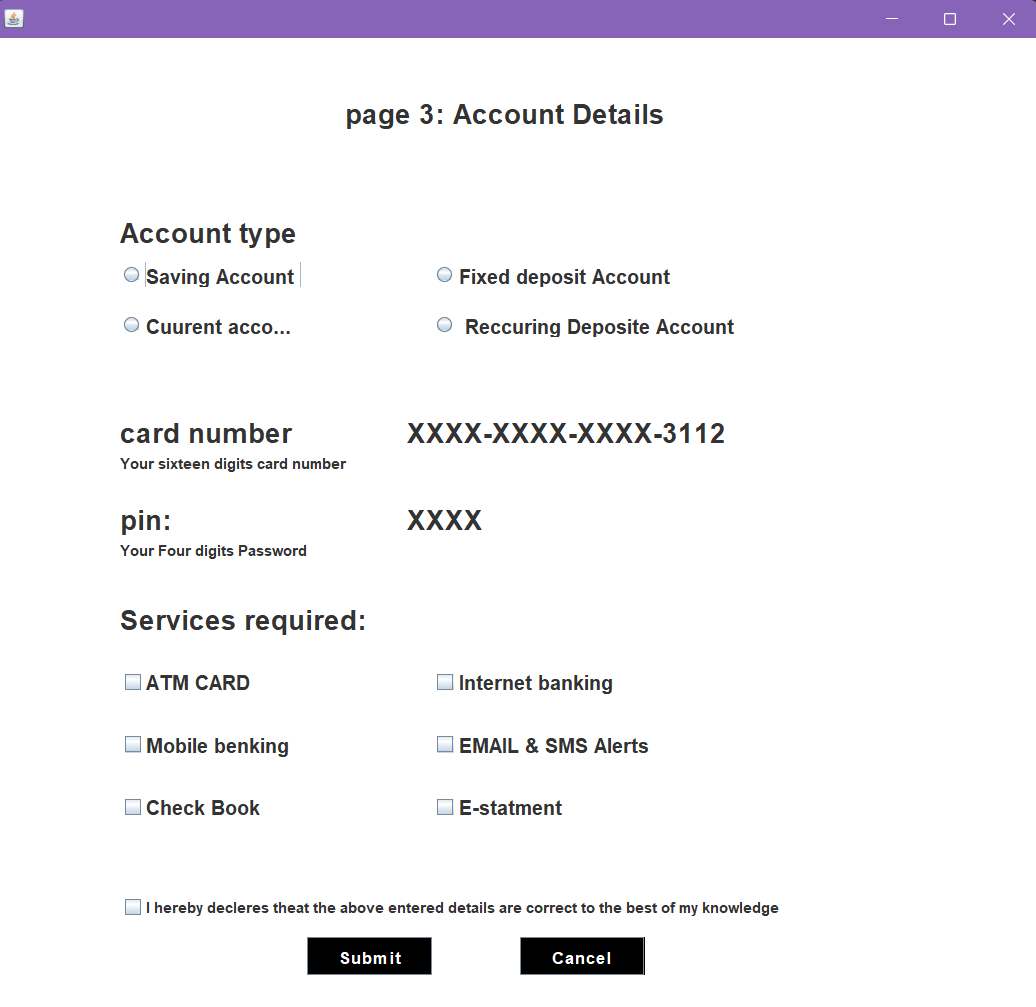
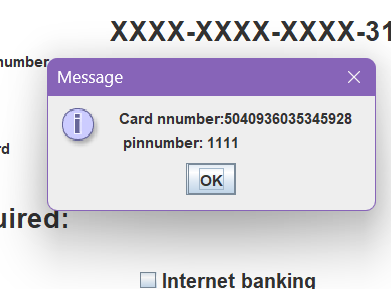


**Signup pages:-**

Page 1:- Page 2 :-

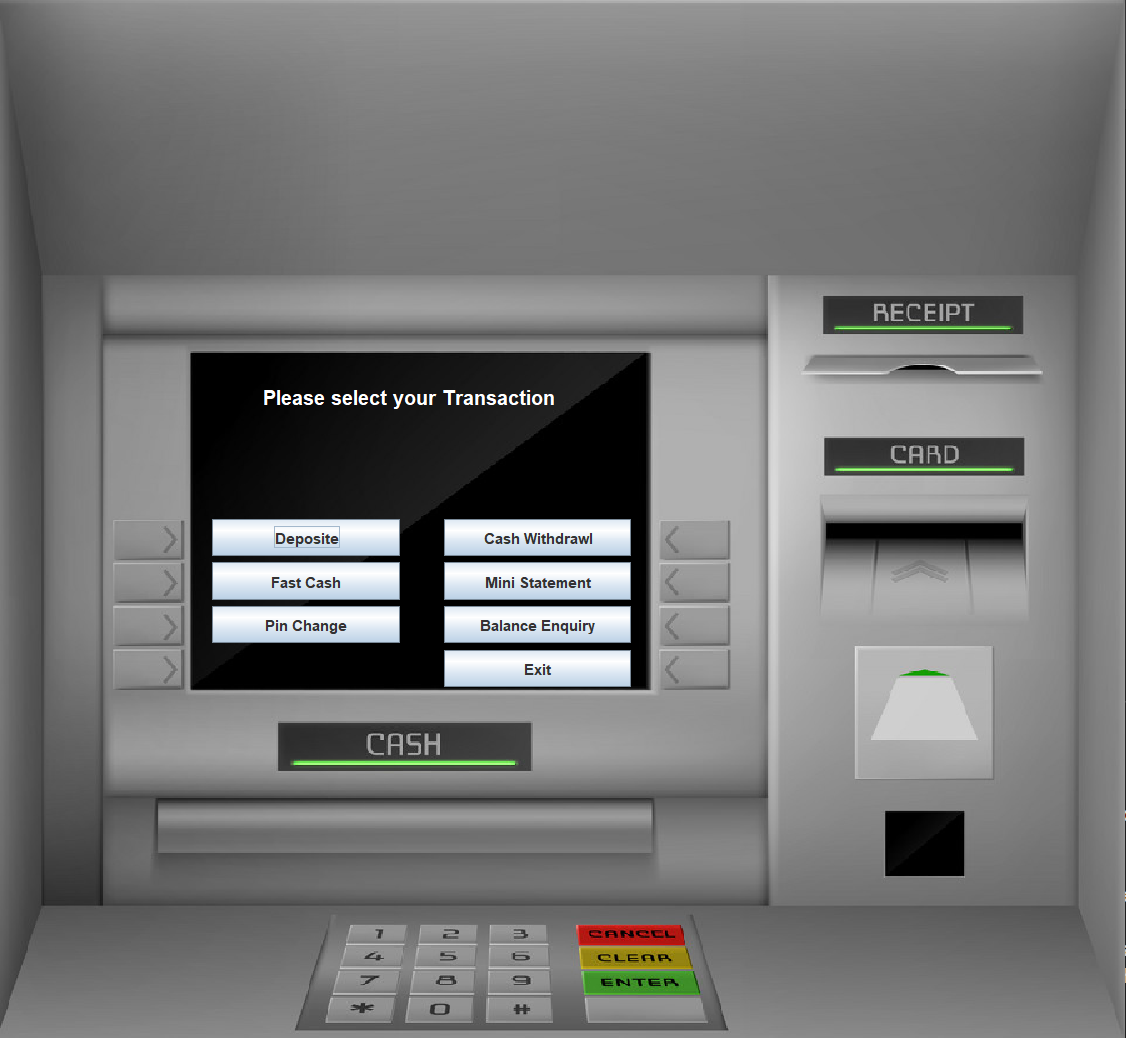
** **

Page 3:-

(Message pop up after completion ofsignup)

**Transaction Page:-**

****

## **7. Conclusion and Discussion**

**7.1 Overall Analysis of Project Viabilities :**

* This project is developed to nurture the needs of a user in a banking sector by embedding all the tasks of transactions taking place in a bank. Future version of this project will still be much enhanced than the current version. Writing and depositing checks are perhaps the most fundamental ways to move money in and out of a checking account, but advancements in technology have added ATM and debit card transactions. All banks have rules about how long it takes to access your deposits, how many debit card transactions you're allowed in a day, and how much cash you can withdraw from an ATM. Access to the balance in your checking account can also be limited by businesses that place holds on your funds.
* Banks are providing internet banking services also so that the customers can be attracted. By asking the bank employs we came to know that maximum numbers of internet bank account holders are youth and business man. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable in a volatile and competitive marketplace of today. If proper training should be given to customer by the bank employs to open an account will be beneficial secondly the website should be made friendlier from where the customers can directly make and access their accounts. Thus, the Bank Management System it is developed and executed successfully.

### **7.2 Summary**

In summary, A banking system is like a safe place for your money. You can put your money in a bank, and they keep it safe for you. When you need some of your money, you can take it out Banks also let you do other things with your money. You can earn some extra money by putting it in a savings account. And if you need to buy something but don't have enough money, the bank can lend you some (like a loan or credit card), but you have to pay it back later.

People use banks to save money, borrow money, and sometimes invest money. Banks also help businesses with their finances, like paying their employees or getting money for new projects.

So, in simple words, a banking system is where you keep your money safe, earn some extra, borrow when needed, and do other important money stuff.

### **7.3 Limitations and Future Enhancement**

* In case of anyone gets confused how to operate the application the website no guidelines.
* Less Security.

Fash cash, Mini statement is been provided in future.

### **7.4 Conclusion**

This project is developed to nurture the needs of a user in a banking sector by embedding all the tasks of transactions taking place in a bank. Future version of this project will still be much enhanced than the current version. Writing and depositing checks are perhaps the most fundamental ways to move money in and out of a checking account, but advancements in technology have added ATM and debit card transactions. All banks have rules about how long it takes to access your deposits, how many debit card transactions you're allowed in a day, and how much cash you can withdraw from an ATM. Access to the balance in your checking account can also be limited by businesses that place holds on your funds.

Banks are providing internet banking services also so that the customers can be attracted. By asking the bank employs we came to know that maximum numbers of internet bank account holders are youth and business man. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable in a volatile and competitive marketplace of today. If proper training should be given to customer by the bank employs to open an account will be beneficial secondly the website should be made friendlier from where the customers can directly make and access their accounts. Thus, the Bank Management System it is developed and executed successfully.

## **Reference**

1. YouTube- www.youtube.com
2. Google- www.google.com
3. GitHub- www.github.com