A

Project Report

On

Bank ATM System Simulator

By

Pratik Kadam (19)

Avinash Prajapati (39)

Under the Guidance of

Prof. Shashikant Ahire

For

Mini Project

In partial fulfillment of

MASTER OF COMPUTER APPLICATION

Semester I

UNIVERSITY OF MUMBAI

NCRD’s Sterling Institute of Management Studies

Nerul, NaviMumbai

2023-2024

NCRD's Sterling Institute of Management Studies

Nerul, NaviMumbai

Certificate of Approval

This is to certify that the Summer project titled ***BANK ATM SYSTEM SIMULATOR*** successfully completed by **Pratik Kadam & Avinash Prajapati** for Semester-I (Academic year 2023-24) in partial fulfillment of **Masters of Computer Application, University of Mumbai,** Mumbai through the NCRD's Sterling Institute of Management Studies Nerul, Navi Mumbai, carried out by him/her under our guidance and supervision.

Date: / /20

Internal Guide	**HOD**
Prof. Shashikant Ahire	Dr. Pragati Goel

Examiner

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: / /20

Pratik Kadam

Avinash Prajapati

ACKNOWLEDGMENT

It gives us immense pleasure in presenting this summer report for the project ***BANK ATM SYSTEM SIMULATOR***. I profoundly thank our **Director Dr. Murlidhar Dhanawade,** for giving us support throughout the course and thus made us capable of being worthy of recognition and extended every facility to us for making and completing this project smoothly.

I would like to express my sincere thanks to **Dr. Pragati Goel, Professor & HOD (MCA)** for his constant encouragement, which made this project a success.

I owe deep gratitude to ***Prof. Shashikant Ahire*** my project guide, for rendering his/her valuable guidance with a touch of inspiration and motivation. He/She has guided me quite a lot in negotiating through the hurdles by giving plenty of early ideas and which resulted in the present fine work.

I would like to thank all the faculty members & staff of NCRD's Sterling Institute of Management Studies, Nerul, Navi Mumbai, for providing us sufficient information which helped me to complete my project successfully. Their guidance has always inculcated confidence in me. And last but not the least, I wish to thank all my friends and well-wishers who are directly or indirectly linked with the success of my project.

Pratik Kadam (19)
Avinash Prajapati (39)
FY MCA A

TABLE OF CONTENTS

ABSTRACT

The Bank Account Management System is an application for maintaining a person's account in a bank. In this project I tried to show the working of a banking account system and cover the basic functionality of a Bank Account Management System. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also, to enable the user’s work space to have additional functionalities which are not provided under a conventional banking project.

The Bank Account Management System undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This project is developed using Java language. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budget.

The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment. The project analyzes the system requirements and then comes up with the requirements specifications. It studies other related systems and then come up with system specifications. The system is then designed in accordance with specifications to satisfy the requirements. The system design is then implemented with Java. The system is designed as an interactive and content management system. The content management system deals with data entry, validation confirm and updating whiles the interactive system deals with system interaction with the administration and users. Thus, above features of this project will save transaction time and therefore increase the efficiency of the system

1. INTRODUCTION

1.1 Introduction

1.1.1 Problem Definition

The “Bank Account Management System” 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.

1.1.2 Objectives of Project

The primary aim of this “Bank Account Management System” is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a 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.

1.1.3 Scope of Project

Anybody who is an Account holder in this bank can become a member of Bank Account Management System. 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.

1.2 Technical Details

1.2.1.Overview of the Front End

- Console Interface:
    - The frontend provides a simple console-based interface for user interaction.
    - Users can choose options such as creating accounts and exiting the system.
- Features:
    - Create Account:
        - Users can create a new bank account by providing account number, account holder name, and initial balance.
    - Exit:
        - Users can exit the system.
- User Input Handling:
    - Utilizes the Scanner class for accepting user input.
    - Ensures proper validation of input data to maintain data integrity.

1.2.2. Back end Overview:

- Java with JDBC:
    - Backend logic is implemented in Java, using JDBC for database connectivity.
    - JDBC is employed to establish a connection to the MySQL database, execute SQL queries, and manage transactions.
- Database Connection:
    - The backend connects to a MySQL database using a JDBC URL, username, and password.
    - Checks if the accounts table exists; if not, creates it.
- Data Operations:

    - Provides methods for creating a new bank account.
    - Executes SQL queries to insert account information into the database.
- Exception Handling:
    - Incorporates exception handling to deal with SQL-related errors.
- Security Measures:
    - Assumes secure database credentials handling for real-world applications.
    - In a production environment, password encryption and other security measures would be implemented.

1.2.3. MySQL Workbench Connectivity:

- Database Structure:
    - Assumes a simple database structure with a single table named accounts.
    - The accounts table contains columns for account number, account holder name, and balance.
- SQL Script:
    - Provided an SQL script to create the necessary accounts table.

2. SYSTEM STUDY & PLANNING

2.1 System Study

2.1.1 Existing System

Error-Prone Data Entry: Manual processes lead to data inaccuracies.

1. Limited Accessibility: Retrieving information is time-consuming and decentralized.
2. Slow Processing: Manual handling results in slower task execution.
3. Security Risks: Sensitive data is vulnerable to loss or unauthorized access.
4. Inefficiency: Labor-intensive processes reduce overall efficiency.
5. Poor Customer Experience: Delays in tasks contribute to a subpar customer experience.
6. Reporting Challenges: Generating reports is time-consuming and error-prone.
7. Audit Complexity: Manual systems complicate auditing processes.
8. Human Dependency: Heavy reliance on manual labor introduces inconsistencies.
9. Scalability Issues: Difficulty in handling increased transaction volumes.

2.1.2 Disadvantages of Existing System

Data Inaccuracy: Manual processes are prone to errors, leading to inaccuracies in account information.

1. Limited Accessibility: Retrieving information is time-consuming, and there is a lack of centralized data storage.
2. Slow Processing: Manual handling results in slower processing times for account-related tasks.
3. Security Risks: Sensitive customer data is at risk of loss, theft, or unauthorized access.
4. Inefficiency: Labor-intensive processes increase workload and reduce overall efficiency.

5. Poor Customer Experience: Delays in account-related processes contribute to a less satisfactory customer experience.
6. Reporting Challenges: Generating reports is time-consuming and error-prone, impacting decision-making.
7. Difficulty in Auditing: Manual systems make auditing challenging, increasing the risk of oversight.
8. Dependency on Manual Labor: Heavy reliance on human intervention introduces the potential for inconsistent processes.
9. Scalability Issues: The manual system may struggle to handle increased transaction volumes and account additions.

2.1.3 Proposed System

1.Technology Stack:

- Backend: Java with JDBC.
- Database: MySQL.
- Frontend: Graphical User Interface (GUI).

2. Database Design:
    - Single table (accounts) for account information.
3. Account Management:
    - Create, update, and delete accounts.
    - Ensure data validation for accuracy.
4. Security Measures:
    - Secure JDBC connection.
    - Data encryption for sensitive information.
5. User-Friendly Interface:
    - GUI for an intuitive user experience.

- Error handling for user feedback.

6. Reporting and Analytics:
    - Real-time reports for account details.
    - Export functionality for analysis.
7. Scalability:
    - Optimized database structure.
    - Transaction handling for concurrency.
8. Audit Trail:
    - Log management for tracking changes.

2.2 System Planning & Schedule

2.2.1 S/W development Model

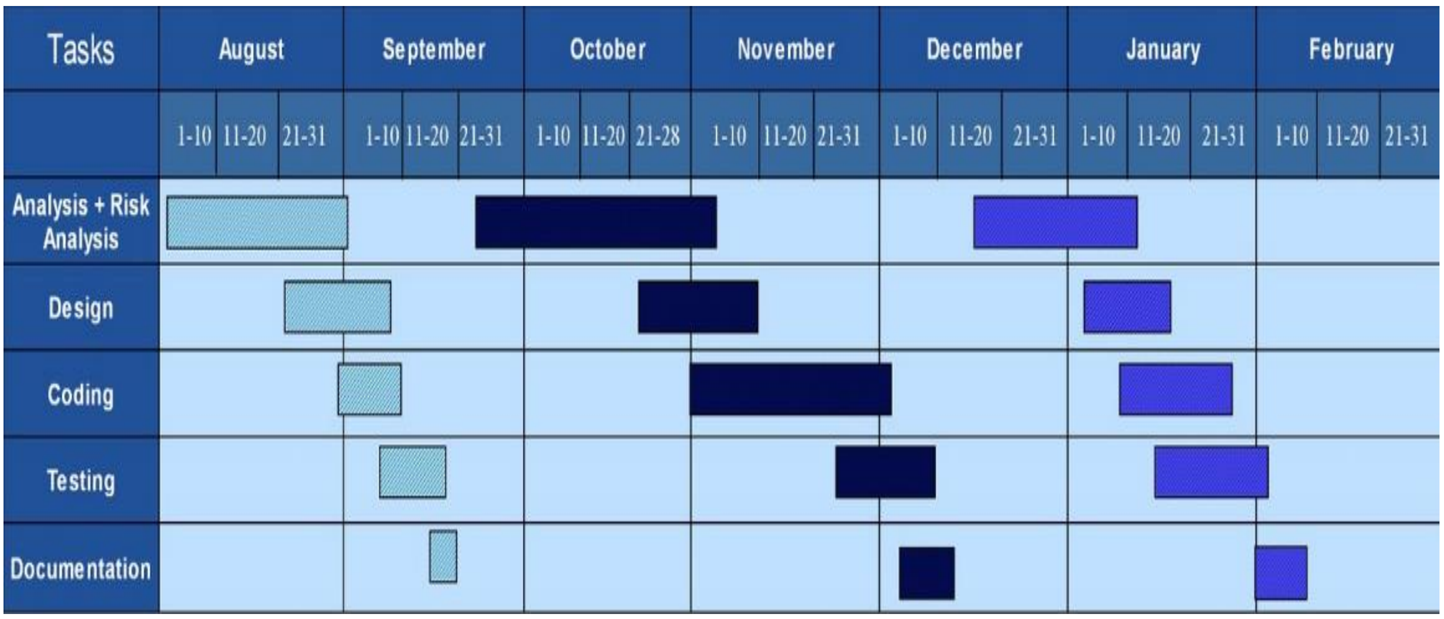
The Agile Model is chosen for the Bank Management System project due to its adaptability to changing requirements and emphasis on iterative development. In the context of the project:

1. Flexibility and Adaptability:
    - Agile allows for frequent changes to project requirements, which is crucial in the dynamic banking sector.
    - The iterative nature accommodates evolving needs and priorities.
2. Iterative Development:
    - The project is divided into small iterations (sprints), typically lasting two to four weeks.
    - Each iteration delivers a potentially shippable product increment.
3. User Involvement:
    - Stakeholders, including end-users, are actively involved throughout the development process.

- Regular feedback ensures that the system aligns with user expectations.

4. Prioritization of Features:
   - Features are prioritized based on their importance and value to the end-users.
   - High-priority features are addressed early in the development cycle.
5. Continuous Delivery:
   - Users can see tangible progress and provide feedback in shorter time frames.
6. Collaboration and Communication:
   - Agile encourages close collaboration between developers, testers, and stakeholders.\

2.2.2 Gantt Chart



3. SYSTEM DESIGN

3.1 Software Requirement Specification(SRS)

3.1.1 Introduction of SRS

1. Project Overview:

- The Bank Management System is a software solution designed to streamline and automate basic banking operations. It aims to provide a secure and efficient platform for managing customer accounts, transactions, and related activities.

2. Purpose:

- The primary purpose of the system is to enhance the overall efficiency of banking processes, ensuring accurate and timely account management while prioritizing data security and user experience.

3. Scope:

- The scope of the SRS encompasses the development of both backend and frontend components, including account creation, updates, and secure database management. It does not include advanced banking features in this initial phase.

4. Stakeholders:

- The main stakeholders include end-users (bank employees and customers), developers, project managers, and system administrators responsible for maintaining the application.

5. Features:

- Core features include account creation, account updates, account deletion, secure database connectivity, user authentication, and a graphical user interface (GUI) for an intuitive user experience.

6. Assumptions and Constraints:

- The system assumes a stable network connection and adherence to basic security protocols.
- Constraints include a predefined budget and time frame for development and deployment.

7. Functional Requirements:

- Detailed functionalities encompass user authentication, account management, database operations, error handling, and reporting capabilities.

8. Non-functional Requirements:

- Performance: The system should handle a predefined number of concurrent users efficiently.
- Security: Implement encryption for sensitive data, secure database connections, and access controls.
- Usability: Ensure a user-friendly GUI with intuitive navigation and error feedback.

9. External Interfaces:

- The system will interact with a MySQL database using JDBC for backend operations. The GUI will provide the interface for end-users.

10. Future Enhancements:

- The system is designed to allow for future enhancements, including additional banking features (e.g., funds transfer) and improved reporting capabilities.

11. Conclusion:

- The SRS serves as a comprehensive guide for the development of the Bank Management System, outlining its purpose, scope, features, and requirements. It provides a roadmap for developers to create a robust and user-friendly banking solution.

3.1.2 Technology Requirements

3.1.2.1 Hardware to be used

1. Development Machines:
    - High-performance computers with sufficient RAM and processing power for software development.

2. Database Server:
    - A dedicated server to host the MySQL database, ensuring data security and reliability.
3. Client Machines:
    - Computers for end-users to access the graphical user interface (GUI) of the Bank Management System.
4. Network Infrastructure:
    - Stable network connectivity to facilitate communication between the client machines and the database server.
5. Backup System:
    - An efficient backup system to regularly back up the database and ensure data integrity.
6. Security Measures:
    - Implementation of security measures such as firewalls, antivirus software, and encryption to protect sensitive data.
7. Printers/Scanners:
    - If needed, printers and scanners for generating reports or handling physical documentation.
8. Testing Environment:
    - Separate machines for testing purposes to simulate real-world scenarios and identify potential issues.

3.1.2.2 Software/tools to be used

1. Java Development Kit (JDK):
    - Required for backend development using Java.
2. Integrated Development Environment (IDE):

- An IDE such as IntelliJ IDEA or Eclipse for coding and project management.

3. MySQL Database:
   - Database management system for storing and retrieving account information.
4. JDBC (Java Database Connectivity):
   - Enables Java applications to interact with the MySQL database.
5. Graphical User Interface (GUI) Library:
   - JavaFX or Swing for developing the frontend GUI.
6. Version Control System:
   - Git for version control and collaboration among developers.
7. Build Tool:
   - Maven or Gradle for managing project builds and dependencies.
8. Testing Framework:
   - JUnit for unit testing Java code.
9. Deployment Tools:
   - Tools like Apache Tomcat for deploying Java web applications.
10. Project Management and Collaboration:
    - Platforms like Jira, Trello, or Asana for project tracking and collaboration.
11. Documentation Tools:
    - Markdown or tools like Confluence for technical and user documentation.
12. Continuous Integration/Continuous Deployment (CI/CD) Tool:
    - Jenkins or GitLab CI for automating the build and deployment process.

3.2 Detailed life Cycle of the Project

3.2.1 Modules

Account Management:

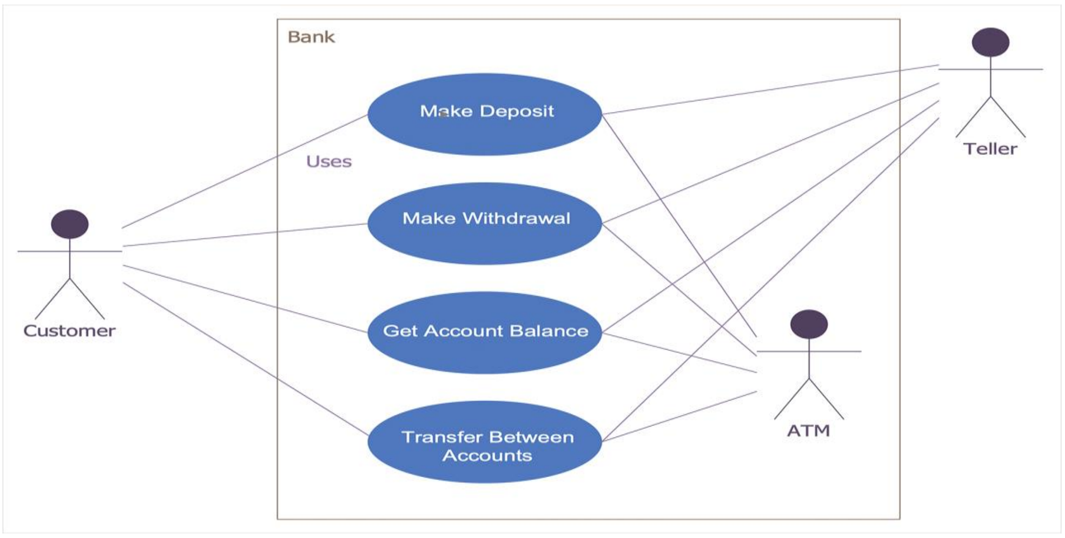
- Functionality: Account creation, updates, and deletion.
- Objective: Efficient management of customer accounts.

2. Database Connectivity:
    - Functionality: Secure JDBC connection, CRUD operations.
    - Objective: Establish and maintain a secure link with the MySQL database.
3. Graphical User Interface (GUI):
    - Functionality: User authentication, intuitive account interface.
    - Objective: Provide a user-friendly and secure interface for end-users.
4. Reporting and Analytics:
    - Functionality: Real-time reports, export functionality.
    - Objective: Enable users to access and analyze account-related data easily.
5. Security:
    - Functionality: Data encryption, access controls.
    - Objective: Ensure the confidentiality and integrity of sensitive information.
6. Integration:
    - Functionality: Backend-Frontend integration, system testing.
    - Objective: Seamlessly combine backend and frontend components for a fully functional system.

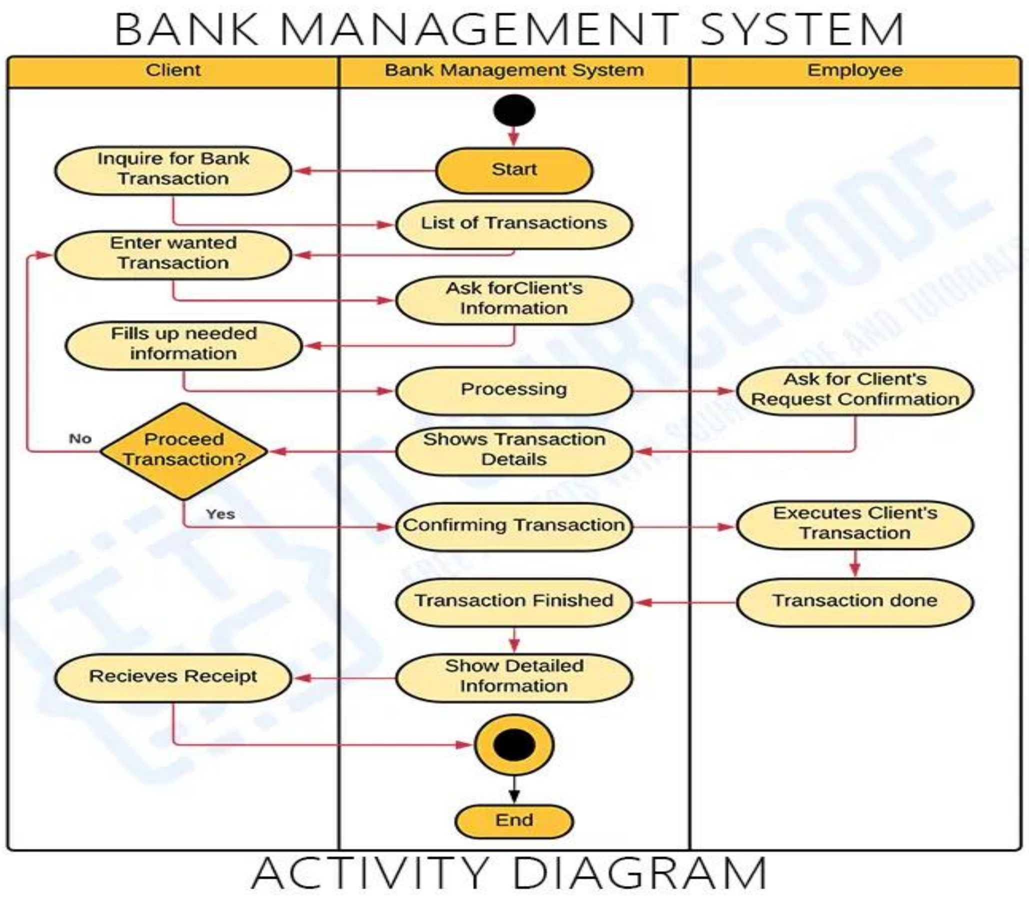
3.2.2 Object Oriented Analysis & Design Diagrams

Designing Object-Oriented Analysis & Design (OOAD) diagrams for a project involves creating visual representations of the system's structure and behavior.

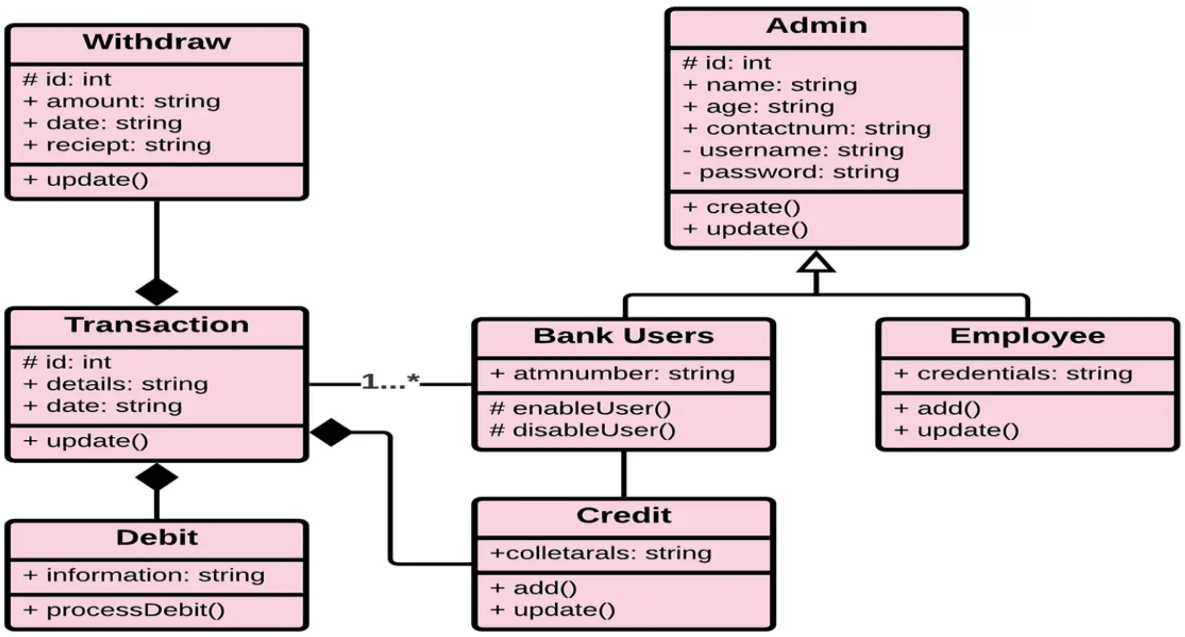
3.2.2.1 Use Case Diagram



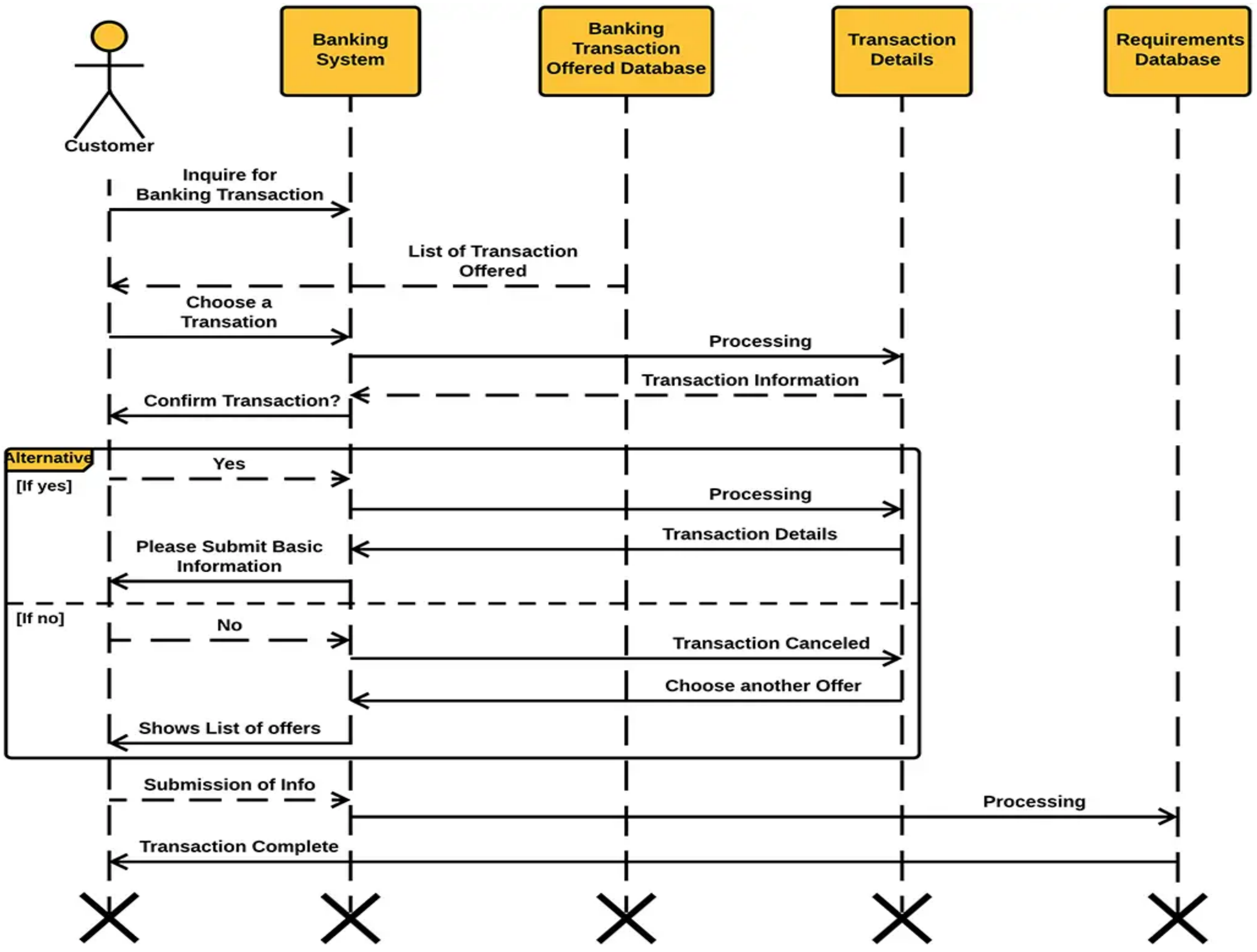
3.2.2.2 Activity Diagram



3.2.2.3 Class Diagram



3.2.2.4 Sequence Diagram



3.2.2.5 Flowchart/DFD/ER diagram

1. Flowchart: Account Creation Process

```
Start --> User Input: Account Details
                |
                v
          Validate Input
                |
[Valid Input?] No -----> Display Error
              | Yes
                v
         Create Account
                |
                v
               End
```

2. Data Flow Diagram (Level 0):

```
          +---------------------+
         |   Bank Management  |
          |      System       |
          +----------+----------+
                     |
      +------------+------------+
          |        |        |
          v        v        v
  +---+---+  +--+---+  +----+---+
  | User |  |Database|  | Report|
  |Input |  |System |  |Module|
  +---+---+  +---+----+  +----+--+
          |        |        |
      +------------+------------+
                     |
                     v
                   +---+
                   |End|
                   +---+
```

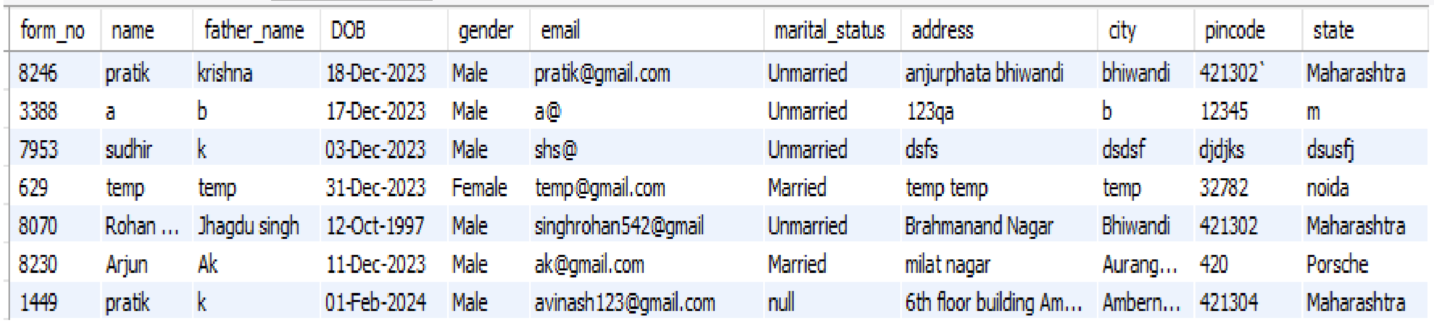
3.2.3 Database

Creating a database for a bank ATM system simulator involves designing tables to store information such as customer details, account information, transaction history, and ATM details. Below is a simplified example of a relational database schema for a basic bank ATM system:

1. Customers Table:
    - CustomerID (Primary Key)
    - FirstName
    - LastName
    - DateOfBirth
    - Address
    - ContactNumber
    - Email
    - PIN (encrypted)
2. Accounts Table:
    - AccountNumber (Primary Key)
    - CustomerID (Foreign Key referencing Customers)
    - AccountType (Savings, Checking, etc.)
    - Balance
    - OpenDate
3. Transactions Table:
    - TransactionID (Primary Key)
    - AccountNumber (Foreign Key referencing Accounts)
    - TransactionType (Deposit, Withdrawal, Transfer, etc.)
    - Amount

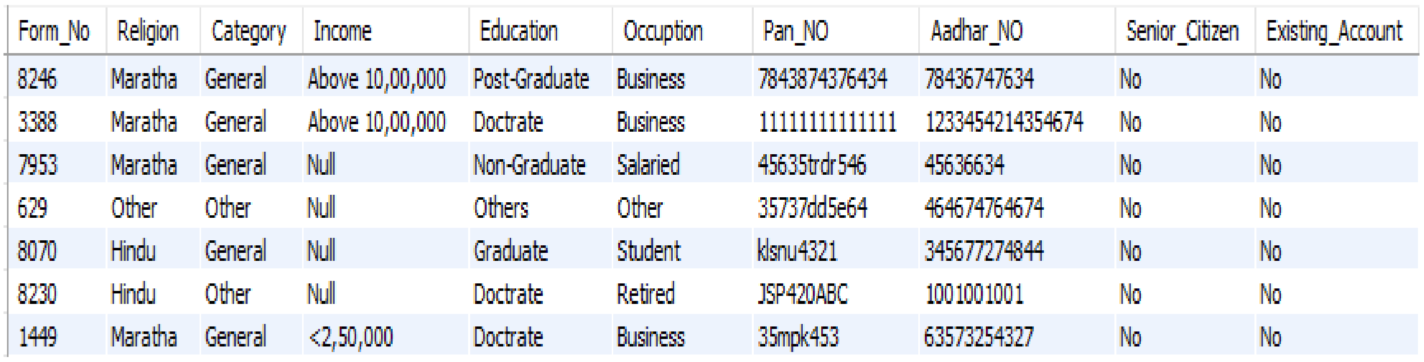
3.2.3.1 Database Table

Personal Details



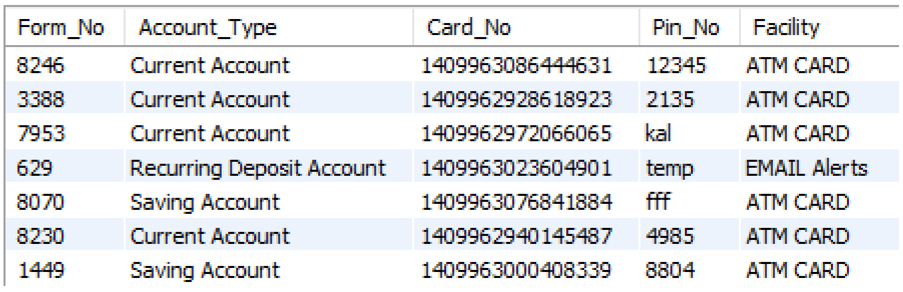
form_no	name	father_name	DOB	gender	email	marital_status	address	city	pincode	state
8246	pratik	krishna	18-Dec-2023	Male	pratik@gmail.com	Unmarried	anjurphata bhiwandi	bhiwandi	421302`	Maharashtra
3388	a	b	17-Dec-2023	Male	a@	Unmarried	123qa	b	12345	m
7953	sudhir	k	03-Dec-2023	Male	shs@	Unmarried	dsfs	dsdsf	djdjks	dsusfj
629	temp	temp	31-Dec-2023	Female	temp@gmail.com	Married	temp temp	temp	32782	noida
8070	Rohan ...	Jhagdu singh	12-Oct-1997	Male	singhrohan542@gmail	Unmarried	Brahmanand Nagar	Bhiwandi	421302	Maharashtra
8230	Arjun	Ak	11-Dec-2023	Male	ak@gmail.com	Married	milat nagar	Aurang...	420	Porsche
1449	pratik	k	01-Feb-2024	Male	avinash123@gmail.com	null	6th floor building Am...	Ambern...	421304	Maharashtra

Additional Details



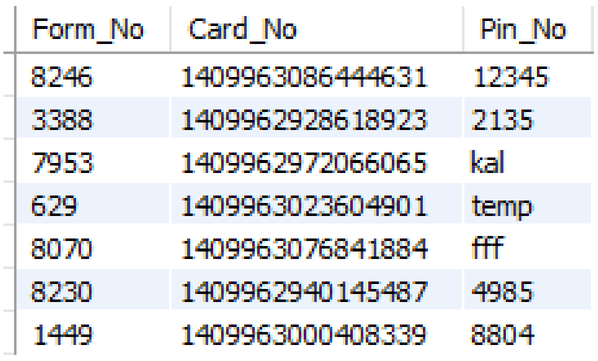
Form_No	Religion	Category	Income	Education	Occuption	Pan_NO	Aadhar_NO	Senior_Citizen	Existing_Account
8246	Maratha	General	Above 10,00,000	Post-Graduate	Business	7843874376434	78436747634	No	No
3388	Maratha	General	Above 10,00,000	Doctrate	Business	11111111111111	123345421435467 4	No	No
7953	Maratha	General	Null	Non-Graduate	Salaried	45635trdr546	45636634	No	No
629	Other	Other	Null	Others	Other	35737dd5e64	464674764674	No	No
8070	Hindu	General	Null	Graduate	Student	klsnu4321	345677274844	No	No
8230	Hindu	Other	Null	Doctrate	Retired	JSP420ABC	1001001001	No	No
1449	Maratha	General	<2,50,000	Doctrate	Business	35mpk453	63573254327	No	No

Account Details



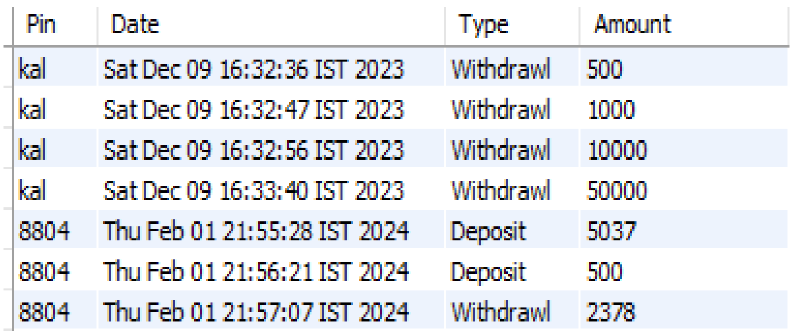
Form_No	Account_Type	Card_No	Pin_No	Facility
8246	Current Account	1409963086444631	12345	ATM CARD
3388	Current Account	1409962928618923	2135	ATM CARD
7953	Current Account	1409962972066065	kal	ATM CARD
629	Recurring Deposit Account	1409963023604901	temp	EMAIL Alerts
8070	Saving Account	1409963076841884	fff	ATM CARD
8230	Current Account	1409962940145487	4985	ATM CARD
1449	Saving Account	1409963000408339	8804	ATM CARD

Card Generation



Form_No	Card_No	Pin_No
8246	1409963086444631	12345
3388	1409962928618923	2135
7953	1409962972066065	kal
629	1409963023604901	temp
8070	1409963076841884	fff
8230	1409962940145487	4985
1449	1409963000408339	8804

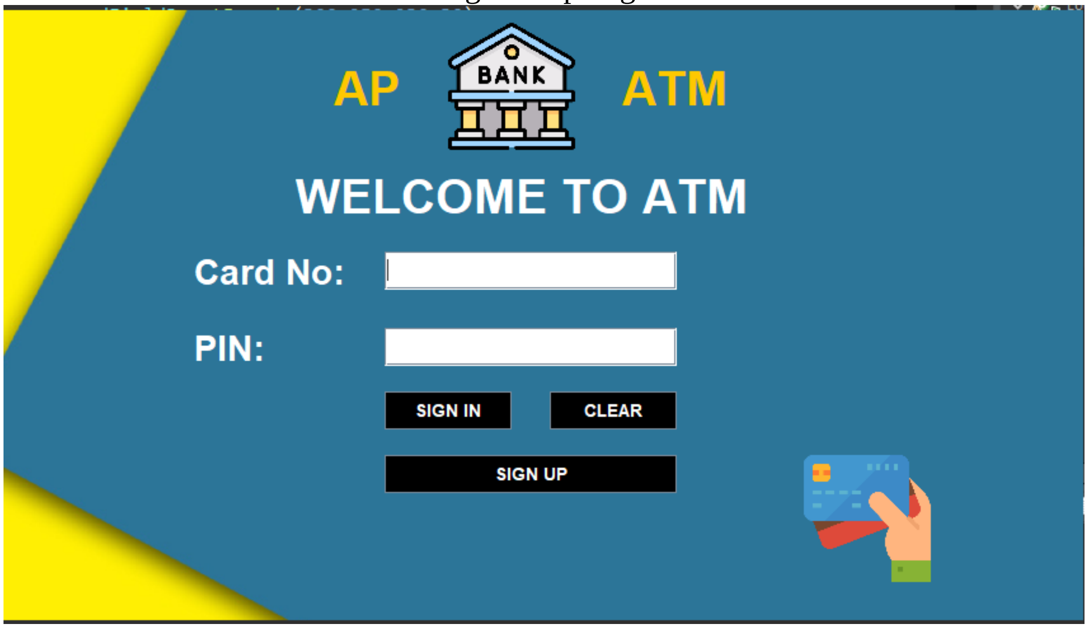
Transaction Status



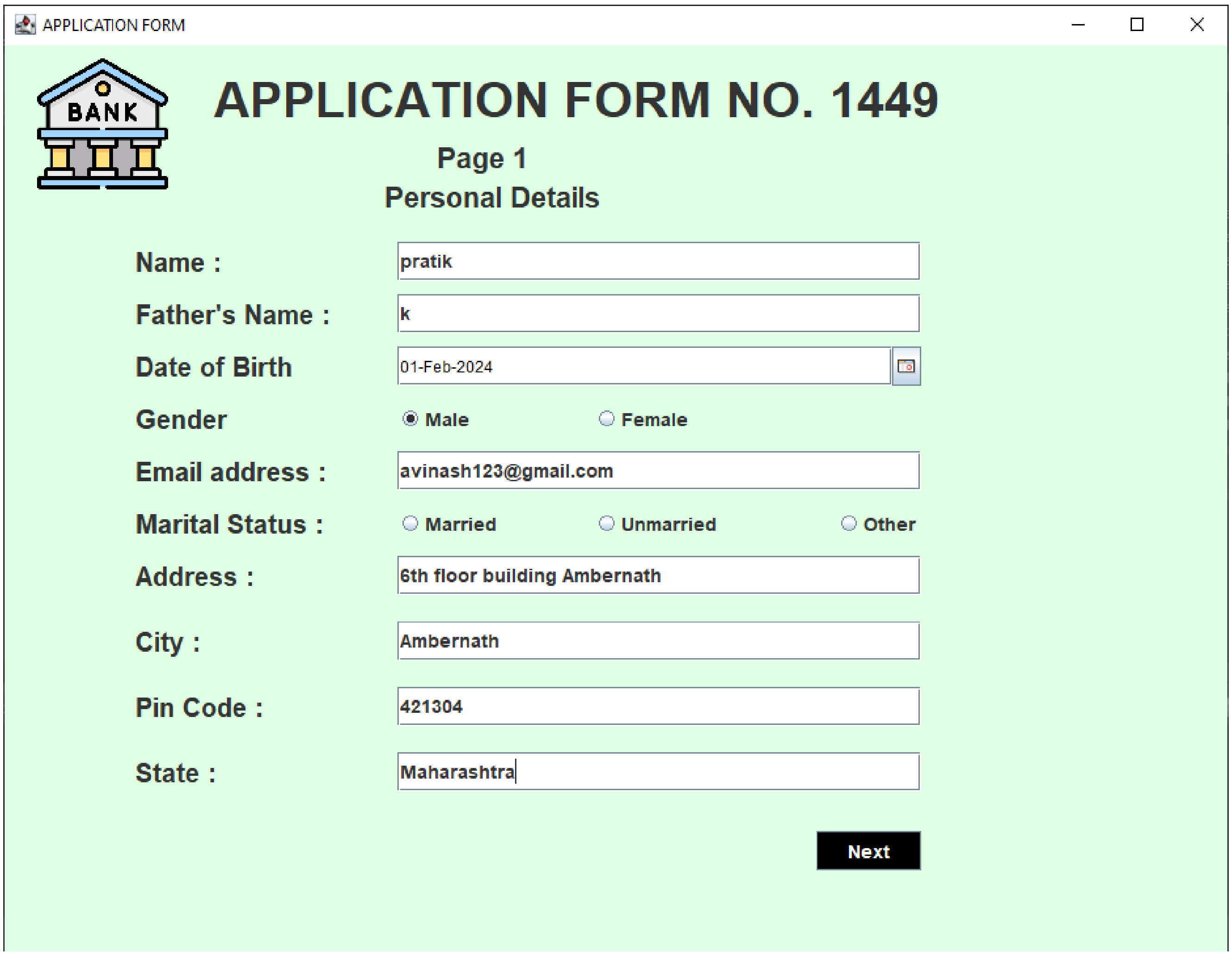
Pin	Date	Type	Amount
kal	Sat Dec 09 16:32:36 IST 2023	Withdrawl	500
kal	Sat Dec 09 16:32:47 IST 2023	Withdrawl	1000
kal	Sat Dec 09 16:32:56 IST 2023	Withdrawl	10000
kal	Sat Dec 09 16:33:40 IST 2023	Withdrawl	50000
8804	Thu Feb 01 21:55:28 IST 2024	Deposit	5037
8804	Thu Feb 01 21:56:21 IST 2024	Deposit	500
8804	Thu Feb 01 21:57:07 IST 2024	Withdrawl	2378

3.2.4 I/O Screen Layout

Sign In/Up Page



Personal Details


APPLICATION FORM

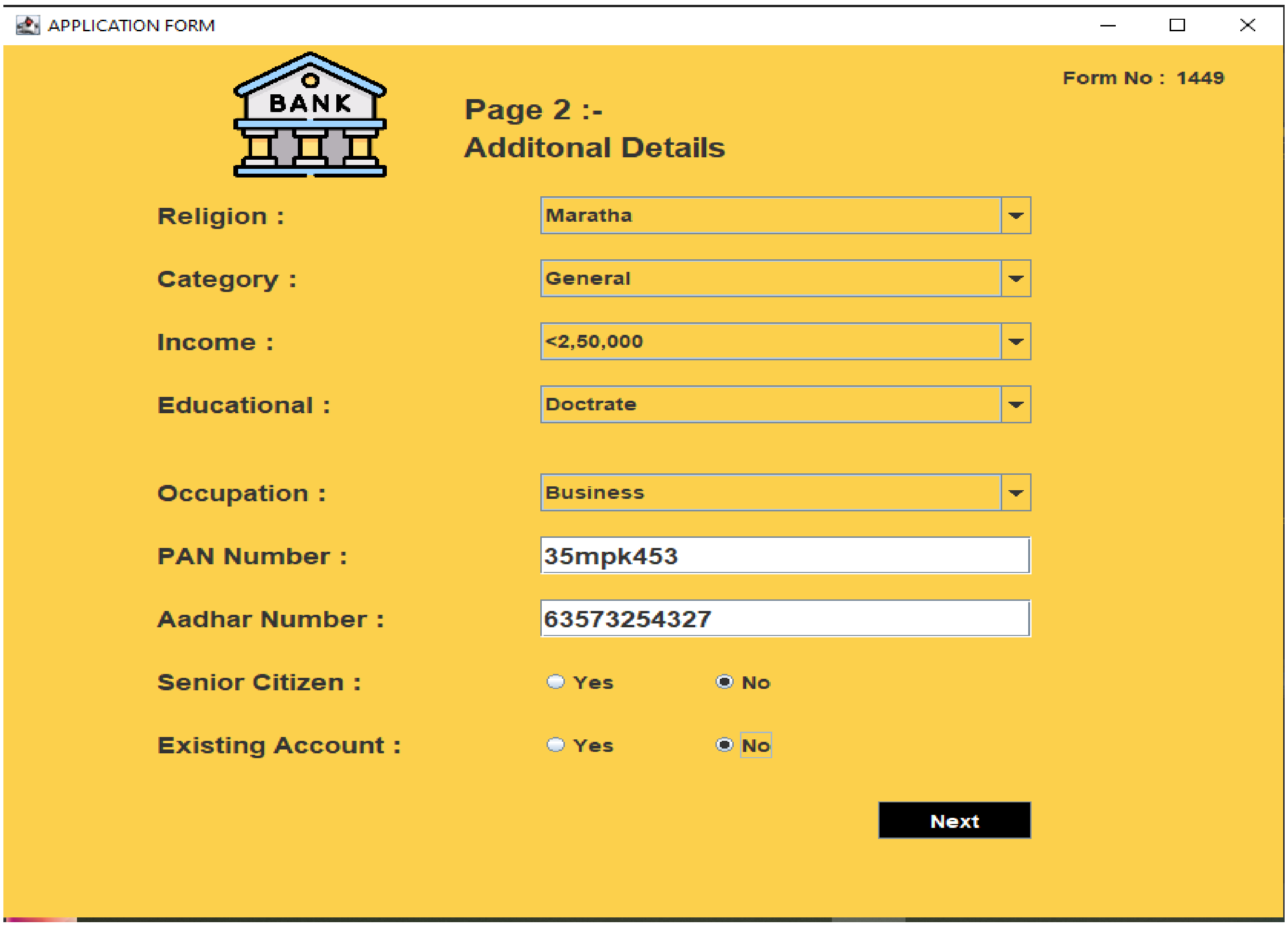
APPLICATION FORM NO. 1449

Page 1

Personal Details

Name :	pratik
Father's Name :	k
Date of Birth	01-Feb-2024
Gender	[✓ Male] Female
Email address :	avinash123@gmail.com
Marital Status :	Married Unmarried Other
Address :	6th floor building Ambernath
City :	Ambernath
Pin Code :	421304
State :	Maharashtra

Additional Details

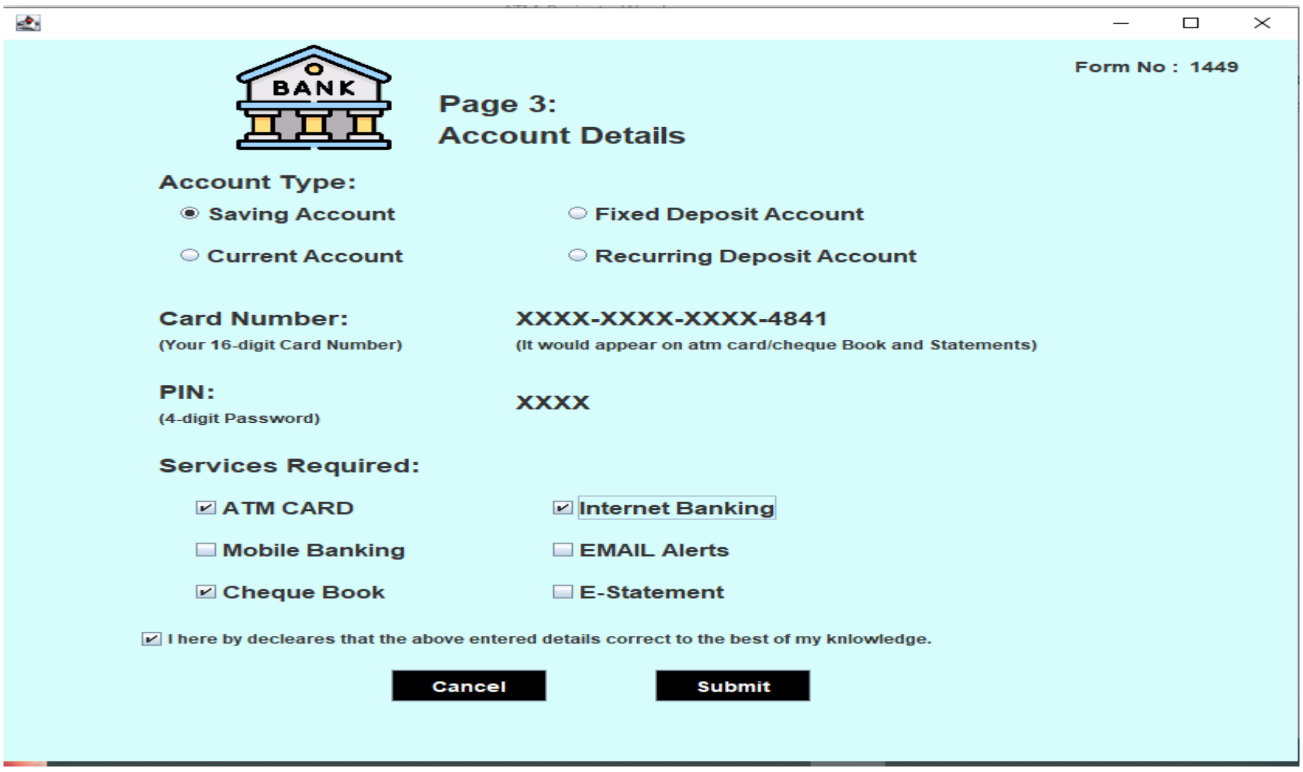


Form No : 1449

Page 2 :-
Additonal Details

Religion :	Maratha
Category :	General
Income :	<2,50,000
Educational :	Doctrate
Occupation :	Business
PAN Number :	35mpk453
Aadhar Number :	63573254327
Senior Citizen :	Yes [✓ No]
Existing Account :	Yes [✓ No]

Account Details



Form No : 1449

Page 3:
Account Details

Account Type:
[✓ Saving Account] Fixed Deposit Account
Current Account Recurring Deposit Account

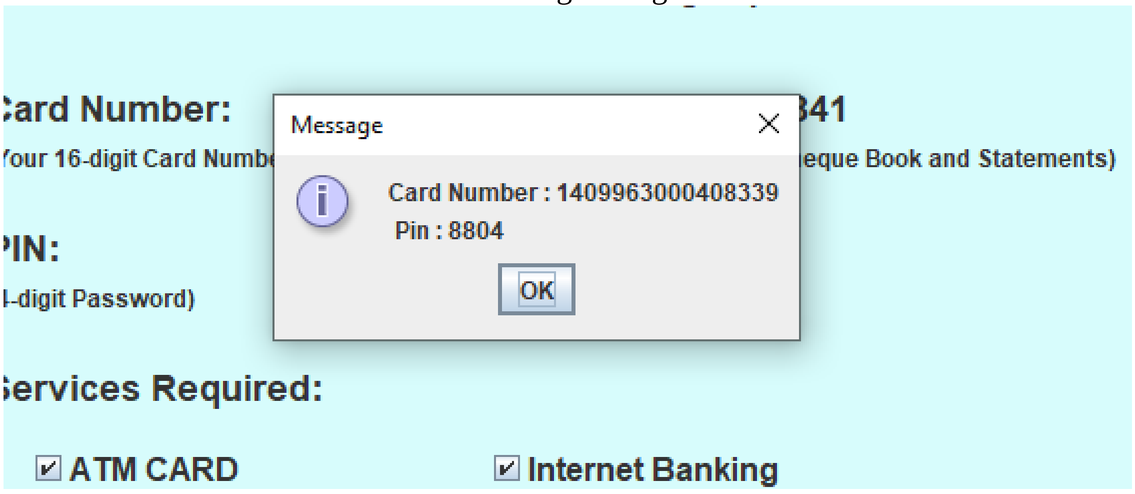
Card Number: XXXX-XXXX-XXXX-4841
(Your 16-digit Card Number) (It would appear on atm card/cheque Book and Statements)

PIN: XXXX
(4-digit Password)

Services Required:
[✓ ATM CARD] [✓ Internet Banking]
Mobile Banking EMAIL Alerts
[✓ Cheque Book] E-Statement

[✓ I here by decleares that the above entered details correct to the best of my knlowledge.]

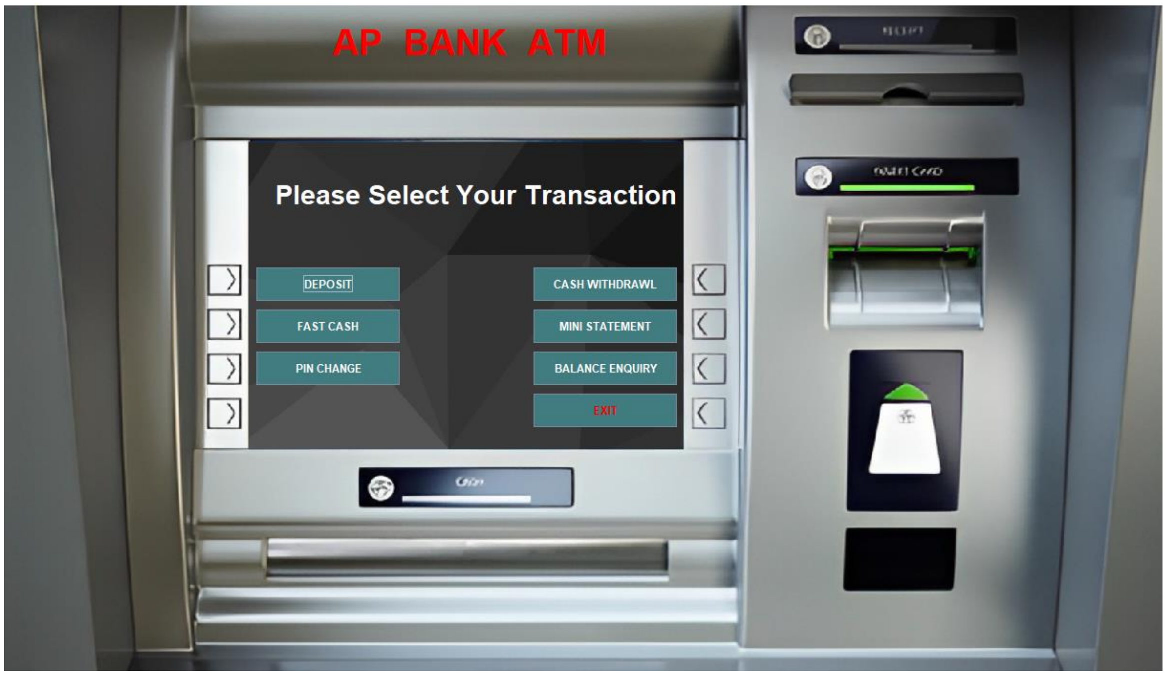
Card Registering

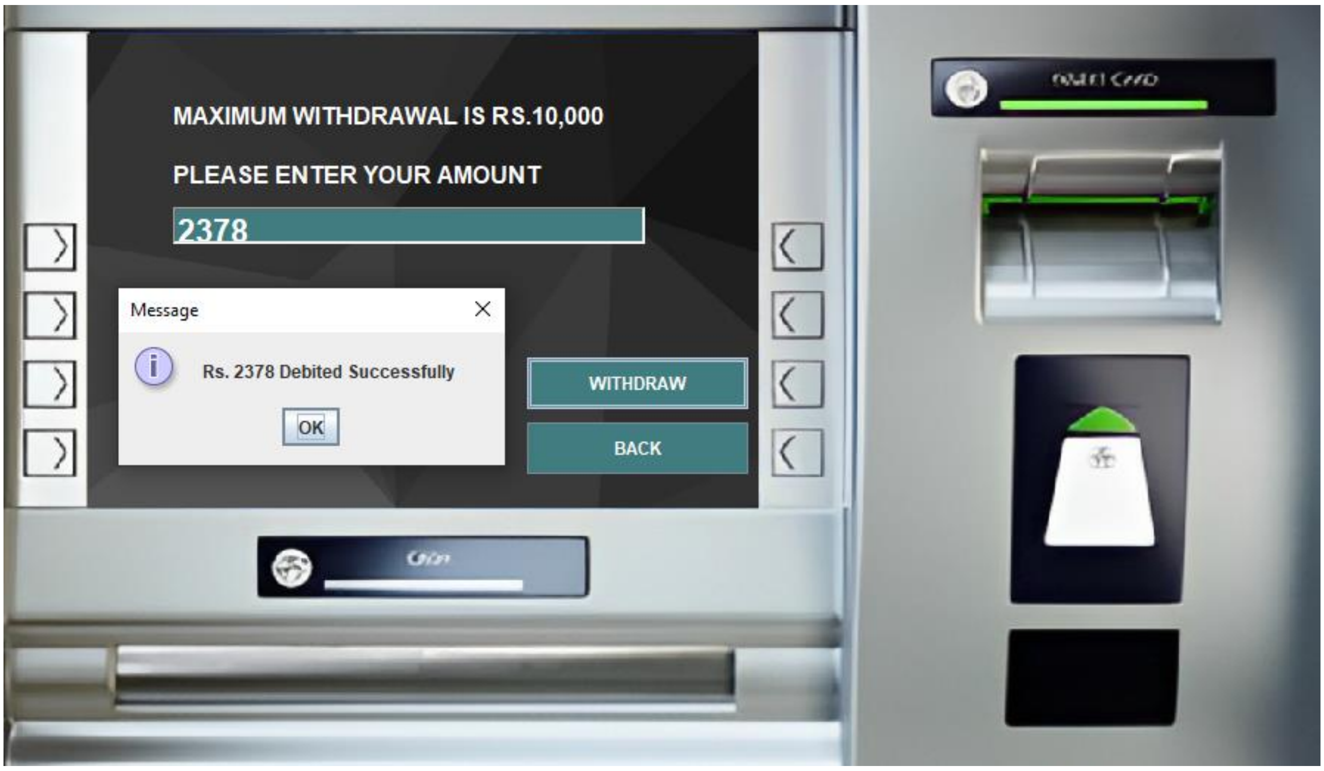


ard Number: 41
four 16-digit Card Numb eque Book and Statements)
PIN:
-digit Password)
ervices Required:
[✓ ATM CARD] [✓ Internet Banking]

Message

Card Number : 1409963000408339
Pin : 8804


AP BANK ATM
Please Select Your Transaction
DEPOSIT
FAST CASH
PIN CHANGE
CASH WITHDRAWL
MINI STATEMENT
BALANCE ENQUIRY
EXIT



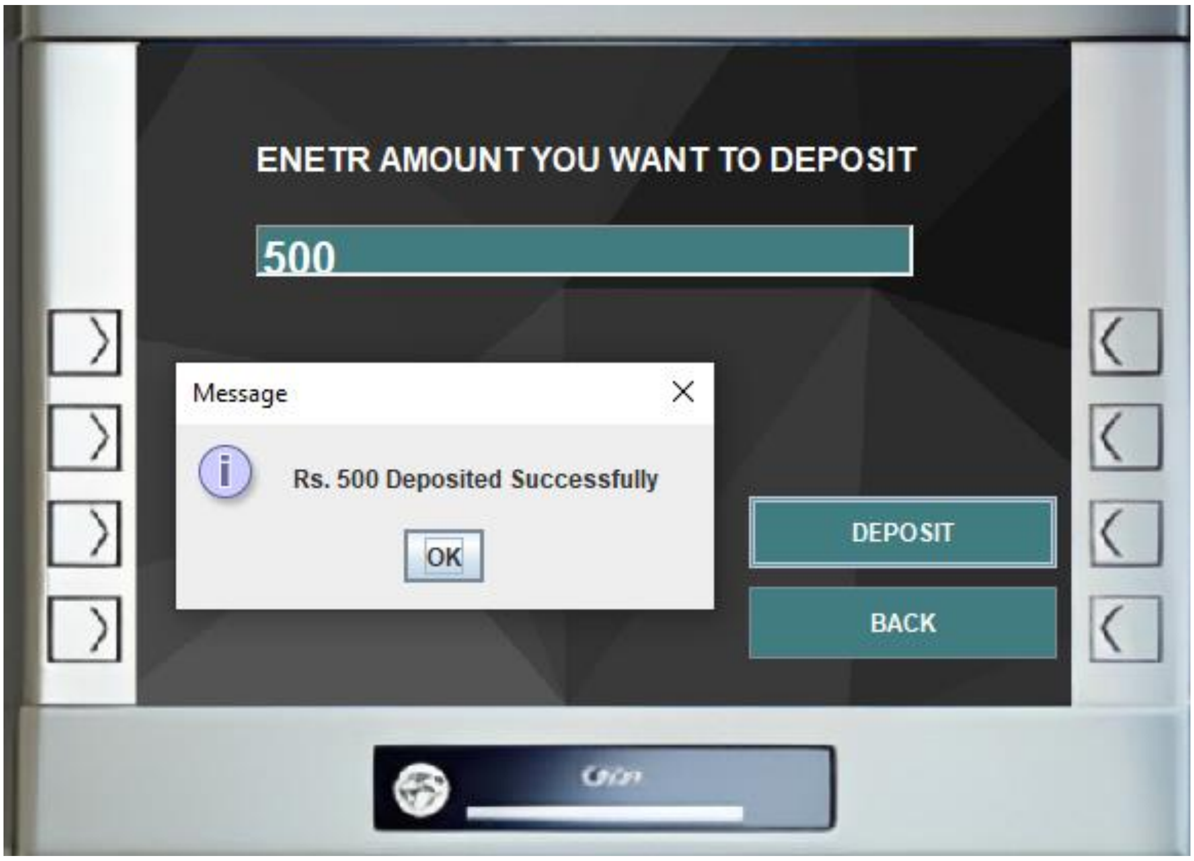
MAXIMUM WITHDRAWAL IS RS.10,000

PLEASE ENTER YOUR AMOUNT

2378

Message

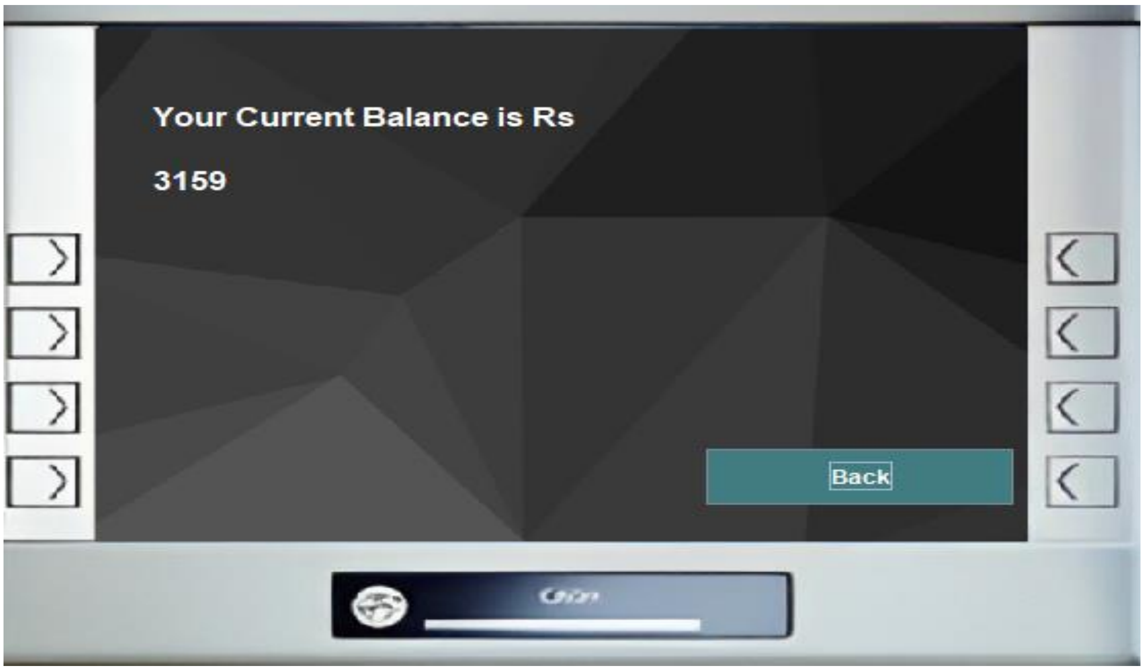
Rs. 2378 Debited Successfully


ENETR AMOUNT YOU WANT TO DEPOSIT

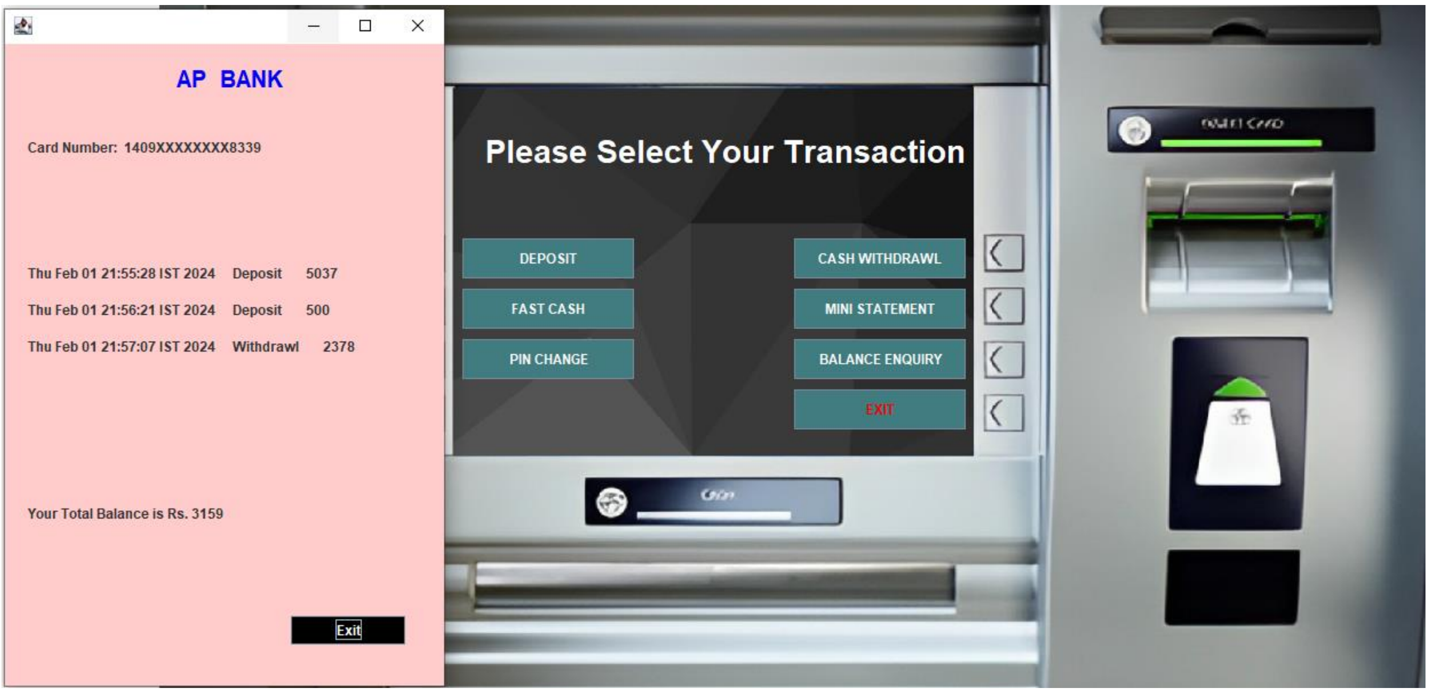
500

Message

Rs. 500 Deposited Successfully


Your Current Balance is Rs

3159


AP BANK

Card Number: 1409XXXXXXXX8339

```
Thu Feb 01 21:55:28 IST 2024   Deposit     5037
Thu Feb 01 21:56:21 IST 2024   Deposit     500
Thu Feb 01 21:57:07 IST 2024   Withdrawl   2378
```

Your Total Balance is Rs. 3159

4. CODING

```
{ Login Page Code}
package AP_BANK_ATM;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;

public class Login extends JFrame implements ActionListener {
    JLabel label01, label0, label1, label2, label3;
    JTextField textField2;
    JPasswordField passwordField3;
    JButton button1,button2,button3;
    Login(){
        super("AP Bank Management System");

        label0 = new JLabel("AP");
        label0.setForeground(Color.orange);
        label0.setFont(new Font("AvantGarde", Font.BOLD, 38));
        label0.setBounds(260,35,450,50);
        add(label0);

        label01 = new JLabel("ATM");
        label01.setForeground(Color.orange);
        label01.setFont(new Font("AvantGarde", Font.BOLD, 38));
        label01.setBounds(487,35,450,50);
        add(label01);

        ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("icon/bank.png"));
        Image i2 =
i1.getImage().getScaledInstance(100,100,Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel image = new JLabel(i3);
        image.setBounds(350,10,100,100);
        add(image);

        ImageIcon ii1 = new
ImageIcon(ClassLoader.getSystemResource("icon/card.png"));
        Image ii2 =
ii1.getImage().getScaledInstance(100,100,Image.SCALE_DEFAULT);
        ImageIcon ii3 = new ImageIcon(ii2);
        JLabel iimage = new JLabel(ii3);
        iimage.setBounds(630,350,100,100);
        add(iimage);

        label1 = new JLabel("WELCOME TO ATM");
        label1.setForeground(Color.white);
        label1.setFont(new Font("AvantGarde", Font.BOLD, 38));
        label1.setBounds(230,125,450,40);
        add(label1);

        label2 = new JLabel("Card No:");
        label2.setFont(new Font("Ralway", Font.BOLD, 28));
        label2.setForeground(Color.WHITE);
        label2.setBounds(150,190,375,30);
        add(label2);
```

```
            textField2 = new JTextField(15);
            textField2.setBounds(300,190,230,30);
            textField2.setFont(new Font("Arial", Font.BOLD,14));
            add(textField2);

            label3 = new JLabel("PIN: ");
            label3.setFont(new Font("Ralway", Font.BOLD, 28));
            label3.setForeground(Color.WHITE);
            label3.setBounds(150,250,375,30);
            add(label3);

            passwordField3 = new JPasswordField(15);
            passwordField3.setBounds(300,250,230,30);
            passwordField3.setFont(new Font("Arial", Font.BOLD, 14));
            add(passwordField3);

            button1 = new JButton("SIGN IN");
            button1.setFont(new Font("Arial", Font.BOLD, 14));
            button1.setForeground(Color.WHITE);
            button1.setBackground(Color.BLACK);
            button1.setBounds(300,300,100, 30);
            button1.addActionListener(this);
            add(button1);

            button2 = new JButton("CLEAR");
            button2.setFont(new Font("Arial", Font.BOLD, 14));
            button2.setForeground(Color.WHITE);
            button2.setBackground(Color.BLACK);
            button2.setBounds(430,300,100, 30);
            button2.addActionListener(this);
            add(button2);

            button3 = new JButton("SIGN UP");
            button3.setFont(new Font("Arial", Font.BOLD, 14));
            button3.setForeground(Color.WHITE);
            button3.setBackground(Color.BLACK);
            button3.setBounds(300,350,230, 30);
            button3.addActionListener(this);
            add(button3);

            ImageIcon iii1 = new
ImageIcon(ClassLoader.getSystemResource("icon/backbg.png"));
            Image iii2 =
iii1.getImage().getScaledInstance(850,480,Image.SCALE_DEFAULT);
            ImageIcon iii3 = new ImageIcon(iii2);
            JLabel iiimage = new JLabel(iii3);
            iiimage.setBounds(0,0,850,480);
            add(iiimage);
            setLayout(null);
            setSize(850,480);
            setLocation(350,150);
            setUndecorated(true);
            setVisible(true);
        }
        @Override
        public void actionPerformed(ActionEvent e) {
            try{
                if (e.getSource()==button1){
                    Connn c = new Connn();
                    String cardno = textField2.getText();
                    String pin = passwordField3.getText();
```

```
                      String q = "select * from login where Card_No =
'"+cardno+"' and  Pin_NO = '"+pin+"'";
                      ResultSet resultSet = c.statement.executeQuery(q);
                      if (resultSet.next()){
                          setVisible(false);
                          new main_Class(pin);
                      }else {
                          JOptionPane.showMessageDialog(null,"Incorrect
Card Number or PIN");
                      }

                  }else if (e.getSource() == button2){
                      textField2.setText("");
                      passwordField3.setText("");
                  }else if (e.getSource() == button3){
                      new Signup();
                      setVisible(false);
                  }
              }catch (Exception E){
                  E.printStackTrace();
              }

          }
          public static void main(String[] args) {
              new Login();
          }
      }
{Main Class}

Package AP_BANK_ATM;


Import javax.swing.*;

Import java.awt.*;

Import java.awt.event.ActionEvent;

Import java.awt.event.ActionListener;


Public class main_Class extends JFrame implements ActionListener {

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

    String pin;

    Main_Class(String pin){

        This.pin = pin;


        ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("icon/atm2.png"));

        Image i2 =
i1.getImage().getScaledInstance(1550,830,Image.SCALE_DEFAULT);

        ImageIcon i3 = new ImageIcon(i2);

        JLabel l3 = new JLabel(i3);

        L3.setBounds(0,0,1550,830);
```

```
Add(l3);

JLabel label1 = new JLabel(“AP  BANK  ATM”);
JLabel label = new JLabel(“Please Select Your Transaction”);
Label.setBounds(430,180,700,35);
Label.setForeground(Color.WHITE);
Label.setFont(new Font(“System”,Font.BOLD,28));
L3.add(label);

Label1.setBounds(490,15,700,45);
Label1.setForeground(Color.red);
Label1.setFont(new Font(“System”,Font.BOLD,38));
L3.add(label1);

B1 = new JButton(“DEPOSIT”);
B1.setForeground(Color.WHITE);
B1.setBackground(new Color(65,125,128));
B1.setBounds(410,274,150,35);
B1.addActionListener(this);
L3.add(b1);

B2 = new JButton(“CASH WITHDRAWL”);
B2.setForeground(Color.white);
B2.setBackground(new Color(65,125,128));
B2.setBounds(700,274,150,35);
B2.addActionListener(this);
L3.add(b2);

B3 = new JButton(“FAST CASH”);
B3.setForeground(Color.WHITE);
B3.setBackground(new Color(65,125,128));
B3.setBounds(410,318,150,35);
B3.addActionListener(this);
L3.add(b3);

B4 = new JButton(“MINI STATEMENT”);
```

```
    B4.setForeground(Color.WHITE);
    B4.setBackground(new Color(65,125,128));
    B4.setBounds(700,318,150,35);
    B4.addActionListener(this);
    L3.add(b4);

    B5 = new JButton("PIN CHANGE");
    B5.setForeground(Color.white);
    B5.setBackground(new Color(65,125,128));
    B5.setBounds(410,362,150,35);
    B5.addActionListener(this);
    L3.add(b5);

    B6 = new JButton("BALANCE ENQUIRY");
    B6.setForeground(Color.WHITE);
    B6.setBackground(new Color(65,125,128));
    B6.setBounds(700,362,150,35);
    B6.addActionListener(this);
    L3.add(b6);

    B7 = new JButton("EXIT");
    B7.setForeground(Color.red);
    B7.setBackground(new Color(65,125,128));
    B7.setBounds(700,406,150,35);
    B7.addActionListener(this);
    L3.add(b7);

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

}

@Override
Public void actionPerformed(ActionEvent e) {
        If (e.getSource()==b1){
           New Deposit(pin);
            setVisible(false);
        }else if (e.getSource()==b7){
            System.exit(0);
        } else if (e.getSource()==b2) {
            New Withdrawl(pin);
            setVisible(false);
        } else if (e.getSource()==b6) {
            New BalanceEnquriy(pin);
            setVisible(false);
        } else if (e.getSource()==b3) {
            New FastCash(pin);
            setVisible(false);
        } else if (e.getSource()==b5) {
            New Pin(pin);
            setVisible(false);
        } else if (e.getSource()==b4) {
            New mini(pin);
        }
    }

    Public static void main(String[] args) {
        New main_Class("");
    }
}
```

5. TESTING

5.1 Methodologies used for testing

In a banking management system project, various testing methodologies are essential to ensure the system's functionality, security, and reliability. Here's a brief summary of the testing methodologies commonly used:

1. Unit Testing:
    - Purpose: Verify individual units or components.
    - Scope: Functions, methods, or procedures.
2. Integration Testing:
    - Purpose: Verify interactions between integrated components.
    - Scope: Combined components.
3. System Testing:
    - Purpose: Validate the entire system's compliance.
    - Scope: System as a whole.
4. Acceptance Testing:
    - Purpose: Ensure the system meets business requirements.
    - Scope: Entire system.
5. Regression Testing:
    - Purpose: Ensure new changes don't impact existing functionalities.
    - Scope: Previously executed test cases.
6. Performance Testing:
    - Purpose: Assess responsiveness and scalability.
    - Types: Load testing, stress testing.

7. Security Testing:
    - Purpose: Identify vulnerabilities.
    - Types: Penetration testing, vulnerability scanning.
8. UI Testing:
    - Purpose: Validate usability and interface.
    - Scope: Graphical User Interface.
9. Database Testing:
    - Purpose: Verify database operations.
    - Scope: Data retrieval, storage, and manipulation.
10. Usability Testing:
    - Purpose: Assess user-friendliness and experience.
    - Scope: Overall user interface.

5.2 Types of Testing

In a banking management system project, various types of testing are crucial to ensure the system's quality and reliability. Here's a brief overview of the types of testing commonly conducted:

1. Unit Testing:
    - Purpose: Verify individual units or components.
    - Scope: Functions, methods, or procedures.
2. Integration Testing:
    - Purpose: Verify interactions between integrated components.
    - Scope: Combined components.
3. System Testing:
    - Purpose: Validate the entire system's compliance.
    - Scope: System as a whole.

4. Acceptance Testing:
    - Purpose: Ensure the system meets business requirements.
    - Scope: Entire system.
5. Regression Testing:
    - Purpose: Ensure new changes don't impact existing functionalities.
    - Scope: Previously executed test cases.
6. Performance Testing:
    - Purpose: Assess responsiveness and scalability.
    - Types: Load testing, stress testing.
7. Security Testing:
    - Purpose: Identify vulnerabilities.
    - Types: Penetration testing, vulnerability scanning.
8. User Interface (UI) Testing:
    - Purpose: Validate usability and interface.
    - Scope: Graphical User Interface.
9. Database Testing:
    - Purpose: Verify database operations.
    - Scope: Data retrieval, storage, and manipulation.
10. Usability Testing:
    - Purpose: Assess user-friendliness and experience.
    - Scope: Overall user interface.

6. CONCLUSION

In conclusion, the Bank ATM System Simulator mini project has successfully demonstrated the development of a simulated environment for managing ATM operations. The simulator emulates key functionalities such as account creation, balance inquiries, deposits, and withdrawals, providing users with a realistic experience of interacting with an ATM.

Throughout the project, a structured software development approach has been followed. The use of Java as the programming language, JDBC for database connectivity, and SQL Workbench for database management has enabled the creation of a robust and functional application.

The project design incorporates visual representations, including flowcharts, Data Flow Diagrams (DFD), and Entity-Relationship (ER) diagrams. These diagrams help in understanding the flow of data, system processes, and relationships between different entities.

Testing has been a critical phase in ensuring the reliability and performance of the system. Various testing methodologies, such as unit testing, integration testing, system testing, and acceptance testing, have been employed. Additionally, performance testing, security testing, and usability testing contribute to the overall quality assurance of the simulator.

The simulator aims to provide users with a user-friendly interface resembling a real ATM system. The inclusion of a main menu, intuitive input screens, and responsive interactions enhances the overall user experience.

In summary, the Bank ATM System Simulator mini project successfully demonstrates the development of a functional and user-friendly ATM simulation. The systematic approach to design, implementation, and testing ensures that the simulator meets its objectives of emulating key banking operations in an ATM environment.

7. LIMITATIONS

1. **Limited Transaction Types:** The simulator may only support basic transactions, omitting more complex services.
2. **Security Constraints:** Security features may not match real-world complexities, potentially overlooking vulnerabilities.
3. **Absence of Real-Time Interaction:** Lack of real-time interaction with live banking systems; transactions may not reflect immediately.
4. **Static Data Usage:** Reliance on static data for accounts and transactions, not reflecting dynamic real-world scenarios.
5. **No Network Simulation:** The simulator might not simulate network interactions, missing real-time communication complexities.
6. **No Physical Interface:** The absence of a physical interface may diminish the realism for users practicing physical interactions.
7. **Authentication Limitations:** Lack of advanced authentication mechanisms, such as biometrics or multi-factor authentication.
8. **Not Mobile-Friendly:** Lack of optimization for mobile devices, limiting accessibility.
9. **No Integration with Physical Devices:** Inability to integrate with physical components like card readers or cash dispensers.
10. **Scope Limitations:** The simulator may focus on a subset of ATM functionalities, not covering the full range of features in diverse ATM systems.

8. FUTURE ENCHANMENTS

- **Expanded Transaction Support:** Include advanced transactions like fund transfers, bill payments, and mobile recharges.
- **Real-Time Interaction:** Enable real-time updates to account balances and transaction history for a more dynamic experience.
- **Enhanced Security Measures:** Implement advanced security features, such as biometric and multi-factor authentication.
- **Dynamic Data Generation:** Generate dynamic data for realistic scenarios, including interest calculations and transaction histories.
- **Network Simulation:** Simulate network interactions for real-time communication complexities.
- **Physical Interaction Simulation:** Integrate with physical components like card readers and cash dispensers for a more immersive experience.
- **User Authentication Options:** Provide various authentication methods, allowing users to explore different approaches.
- **Mobile-Friendly Design:** Optimize the simulator for mobile devices for improved accessibility.
- **Integration with External Systems:** Integrate with core banking systems, payment gateways, and fraud detection systems.
- **User Customization:** Allow users to customize preferences, set transaction limits, and choose authentication methods.
- **Comprehensive Reporting:** Implement detailed transaction reports and statements for user insights.
- **Educational Resources:** Include educational tips on secure transactions, financial literacy, and ATM usage.
- **Multi-User Support:** Enable multiple users to simulate transactions independently.
- **Scenario Testing:** Introduce scenario-based testing for specific situations like card loss or transaction disputes.
- **Integration with Banking Regulations:** Incorporate features related to compliance with banking regulations and standards.

9. REFERENCES

1. Unique Developer YouTube Channel
(https://www.youtube.com/watch?v=zdWfyBXO2iU)

2. Online Bank Account Management System
Website: http://www.slideshare.net (Collect some info for report documents)

3. Learning MYSQL, JavaScript, jQuery, PHP, HTML, CSS3,
Website: http://www.w3schools.com

4. PHP and MySQL video tutorials
Website: http://www.freehinditutorial.com, http://www.youtube.com

5. Veneeva, V. (2006), "E-Banking (Online Banking) and Its Role in Today's Society", Ezine articles

6. JavaScript validation for empty input field
Website:http://stackoverflow.com/questions/3937513/javascript-validation-for-empty-input-field ,

7. JavaScript form validation: Validate Password, Validate Email, Validate Phone Number, http://webcheatsheet.com/javascript/form_validation.php