ATM & BANKING SYSTEM

Project Report Submitted

To

Gujarat University

In partial fulfilment of the requirements for

the award to the Degree of

5 YEAR INTEGRATED MASTER OF SCIENCE (COMPUTER SCIENCE)

SEMESTER – V

GUIDED BY:

DR HARDIK JOSHI

SUBMITTED BY:

LAMIN JANKA (50021)

MOHIT JARAWALA (50022)

DEPARTMENT OF COMPUTER SCIENCE
GUJARAT UNIVERSITY, AHMEDABAD

YEAR: 2023-24

Department Of Computer Science

Gujarat University

Certificate

Roll No : _______ Seat No : _______

This is to certify that Mr. /Ms. ________________________________ student of Fifth Semester of 5 years Integrated M.Sc (Computer Science) has duly completed his/her project titled ________________________________ ____________________________ for the semester ending in December 2024, towards partial fulfillment of degree of 5 years Integrated M.Sc (Computer Science).

Date of Submission Internal Project Guide

Course Coordinator

Head of Department

Department Of Computer Science

Gujarat University

Certificate

Roll No : ________   Seat No : _______

This is to certify that Mr. /Ms. __________________________________ student of Fifth Semester of 5 years Integrated M.Sc (Computer Science) has duly completed his/her project titled ____________________________________ ______________________________ for the semester ending in December 2024, towards partial fulfillment of degree of 5 years Integrated M.Sc (Computer Science).

Date of Submission   Internal Project Guide

Course Coordinator

Head of Department

Acknowledgements

We would like to express our sincere gratitude to **Dr. Hardik Joshi**, our project guide, for his invaluable guidance, constructive feedback, and continuous support throughout the development of this project. His expertise and encouragement were instrumental in overcoming challenges and ensuring the successful completion of this work.

We are also deeply thankful to **Dr. Bhumika Shah** for her insightful suggestions and assistance during the course of this project. Her input has enriched our understanding and contributed significantly to its quality.

Our heartfelt appreciation also goes to **Dr. Jyoti Pareek**, Head of the Department of Computer Science, Gujarat University, for providing us with the opportunity to work on this project and for fostering an environment of academic excellence and innovation.

Lastly, we extend our gratitude to all the faculty members and staff of the Department of Computer Science for their constant encouragement, resources, and support, which have been vital in shaping our academic journey.

Thank you all for your unwavering belief in our potential and for contributing to the success of this project.

Table of contents

About Project

Project Profile

Title	Description
Project title	ATM & Banking Services
Aim of project	The ATM & Banking Management System is designed to simulate real-world banking operations and ATM functionalities, including customer account management, balance inquiry, money withdrawal, deposit, and statement generation.
Project duration	6 months
Team size	2 members
Team Members	Lamin Janka(50021) & Mohit Jariwala(500**)
Tools and technologies used	Java(Swing)
Project Guide	Dr. Hardik Joshi

Introduction

The ATM and Banking System project is a comprehensive solution designed to enhance the efficiency and convenience of banking operations. This system focuses on automating routine banking tasks, ensuring secure financial transactions, and providing users with seamless access to banking services through an intuitive interface. The primary objective of the project is to create a robust and scalable system that caters to the needs of both customers and banking personnel**.**

Project Objectives

1. **Customer Convenience:**
   The system enables customers to perform various banking operations, such as checking account balances, transferring funds, withdrawing cash, and viewing transaction history, without the need for direct interaction with bank staff. This ensures 24/7 accessibility and reduces dependency on traditional banking hours.
2. **Operational Efficiency:**
   By automating core banking functionalities, the system minimizes manual intervention, reduces errors, and speeds up banking operations. This improves overall service quality and enhances customer satisfaction.
3. **Security:**
   The project incorporates advanced security measures, including encryption, secure authentication, and real-time monitoring, to safeguard user data and financial transactions from potential threats.
4. **Scalability:**
   The design is flexible, allowing for future enhancements such as integration with mobile banking applications, support for multiple languages, and inclusion of additional banking services like loans and fixed deposits.

Key Features

1. **ATM Module:**
   - Cash withdrawal and deposit.

    - Balance inquiry and transaction history.
    - PIN generation and modification.

2. **Banking System Module:**
    - Account management for customers, including account creation, deletion, and modification.
    - Transaction processing, including fund transfers and bill payments.
    - Administrative features for bank staff, such as report generation and user management.

3. **User Authentication:**
   The system uses multi-factor authentication to ensure secure access, including PINs, passwords, and potentially biometric authentication in future iterations.

4. **Database Management:**
   A robust database is implemented to manage customer details, account information, transaction records, and system logs efficiently and securely.

Tools & Technologies Used

- **Programming Language:**
    - Java
- **Database:**
    - MySQL
- **Development Tools(IDE):**
    - Visual Studio Code
    - IntelliJ
- **Libraries/Frameworks:**
    - JDBC – (For database Connection (DB: Mysql))
    - Java Swing – (For UI implementation)
    - AWT – (for UI Implementation)
    - Calander – (for Date Functionality)
- **Encryption**:
    - Standard encryption methods for securing sensitive data like PINs

    - METHOD: MD-5

- **Operating System:**
    - Windows (for development and testing)
- **Other Development Tools:**
    - Launch4j – (Export from .jar file to .exe file)
    - Inno Setup Compiler – (Export final Setup of Application)

System Engineering

The **ATM and Banking System** project is designed to deliver a secure, efficient, and user-friendly platform for performing banking operations. This section provides a detailed breakdown of the system engineering process, including its overall architecture, requirement analysis, feasibility study, and proposed system specifications.

1. System Overview

The ATM and Banking System integrates core banking functionalities with an intuitive interface to facilitate everyday transactions. The system automates processes such as cash withdrawals, fund transfers, balance inquiries, and administrative tasks, thereby reducing manual effort and enhancing customer convenience.

The primary users of this system are:

- **Bank Customers**: For self-service banking activities.
- **Bank Administrators**: For managing accounts, generating reports, and maintaining system operations.

2. Requirement Analysis and Feasibility Study

This phase involved gathering and analysing the requirements of all stakeholders to ensure the system meets their needs efficiently.

a. Functional Requirements:

- **Customer Features**:
    - Withdraw and deposit cash.
    - Check account balance and transaction history.
    - Transfer funds securely.
    - Generate or reset PINs.
- **Administrative Features**:

    - Create, modify, or delete customer accounts.
    - Generate detailed transaction and audit reports.
    - Monitor system logs for suspicious activities.

b. Non-Functional Requirements:

- **Performance**: The system should handle multiple concurrent transactions without significant delays.
- **Security**: All data should be encrypted, and authentication mechanisms should be robust.
- **Reliability**: The system should ensure high availability and minimal downtime.
- **Scalability**: The architecture should support future expansion, such as integration with mobile banking platforms.

c. Feasibility Study:

The feasibility study evaluated the following:

- **Technical Feasibility**: Ensured the tools and technologies chosen (Java, MySQL, etc.) meet the project's requirements.
- **Economic Feasibility**: Confirmed the project budget is justified by the expected benefits.
- **Operational Feasibility**: Determined that bank staff and customers can easily adopt the system with minimal training.

3. Proposed System

The proposed system aims to overcome the limitations of traditional banking systems by providing the following:

a. Objectives:

- To simplify and secure banking operations.
- To reduce manual workload and processing times.
- To enhance customer satisfaction with a seamless user experience.

b. Hardware and Software Platforms:

- **Development Environment**:
    - Hardware:
        - Minimum 4 GB RAM, 500 GB storage, and a multi-core processor.
    - Software:
        - IDE: IntelliJ IDEA or Eclipse for development.
        - Database: MySQL.
        - Server: Localhost
        - Executable final software: Launch4j
        - Executable final software Bundled with JRE: Inno Setup Compiler
        - Operating System: Windows 10 or higher.
- **Deployment Environment**:

- Hardware:
  - Secure servers with RAID storage for redundancy.
- Software:
  - Server OS: Linux or Windows Server.
  - Middleware for transaction processing.

System Design

System Architecture

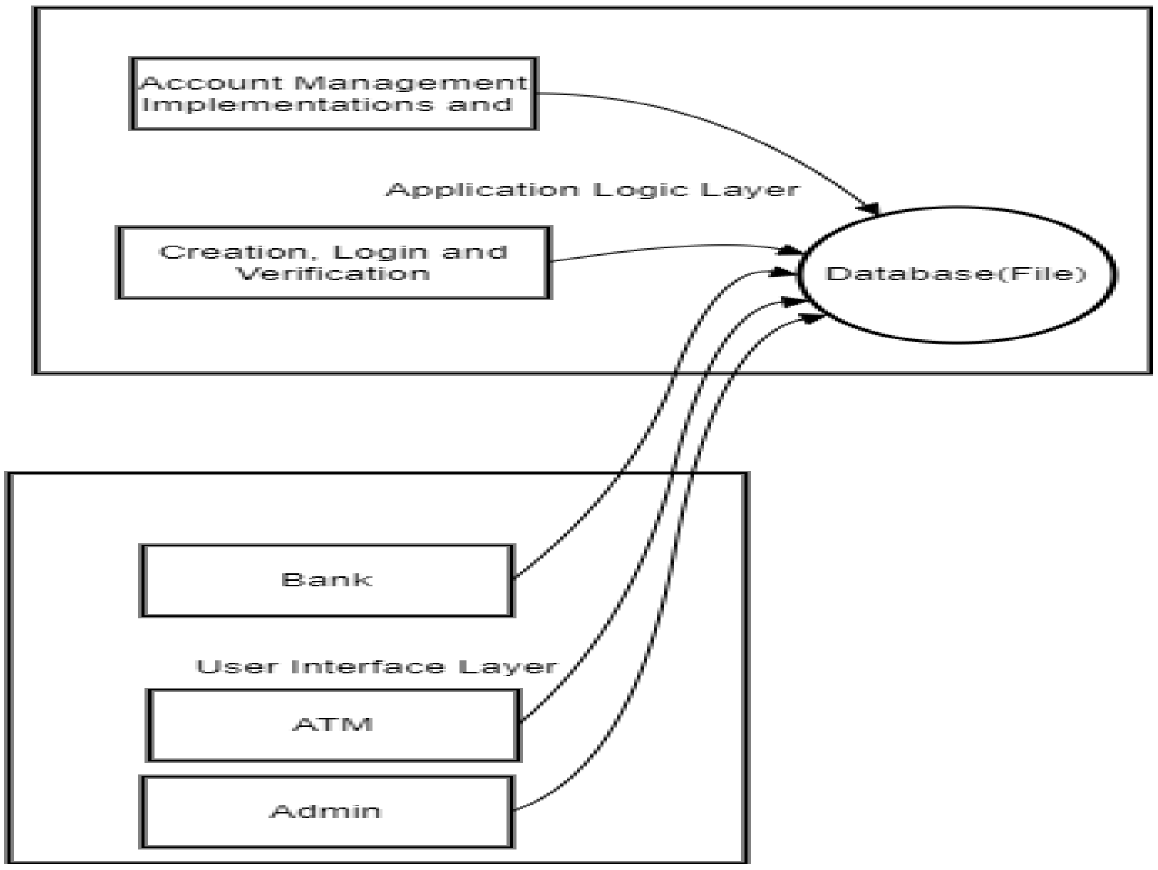

Account Management Implementations and
Application Logic Layer
Creation, Login and Verification
Database(File)
Bank
User Interface Layer
ATM
Admin

Figure 1: System Architecture

Flowchart

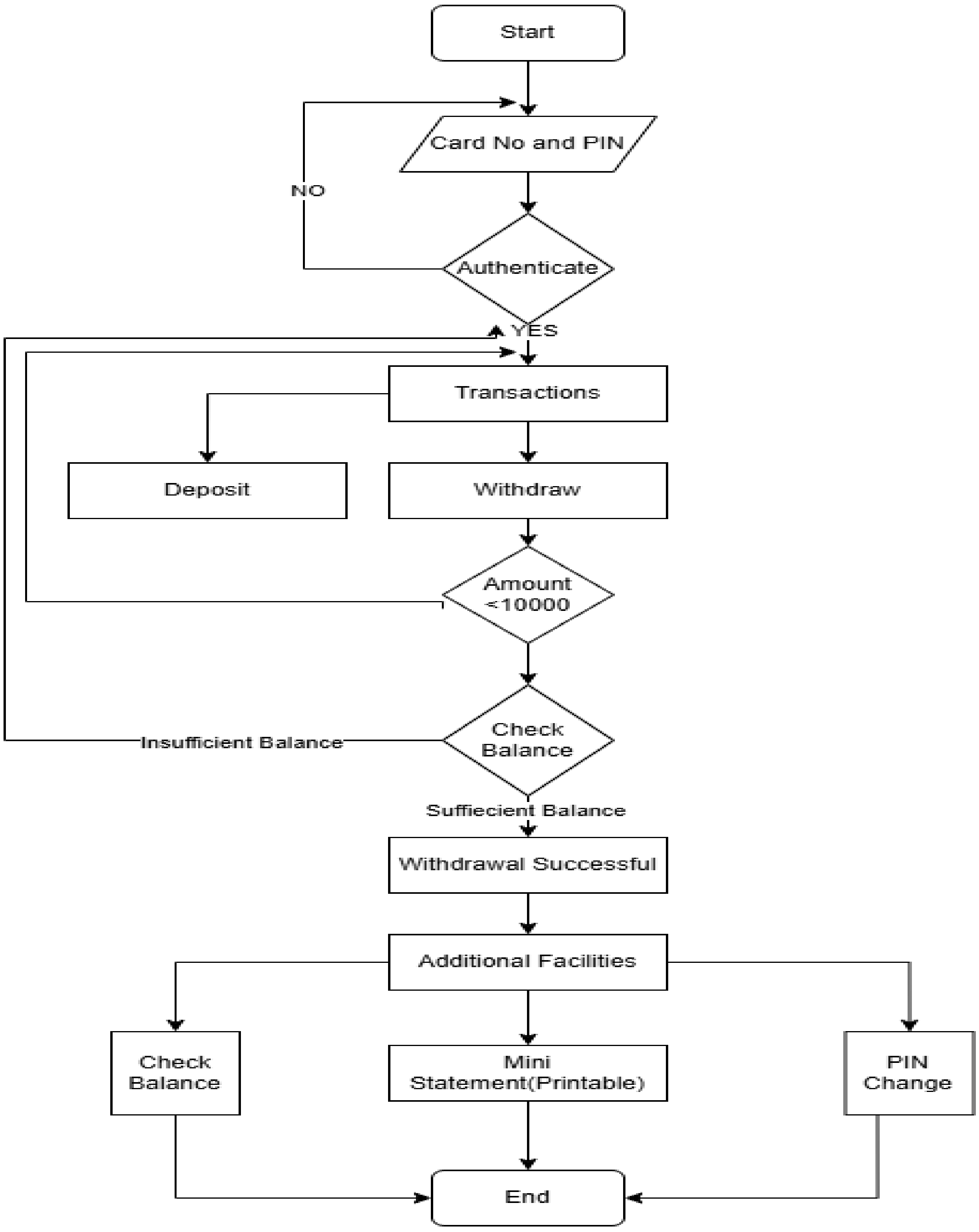


Figure 2:Flow Chart

Use Case

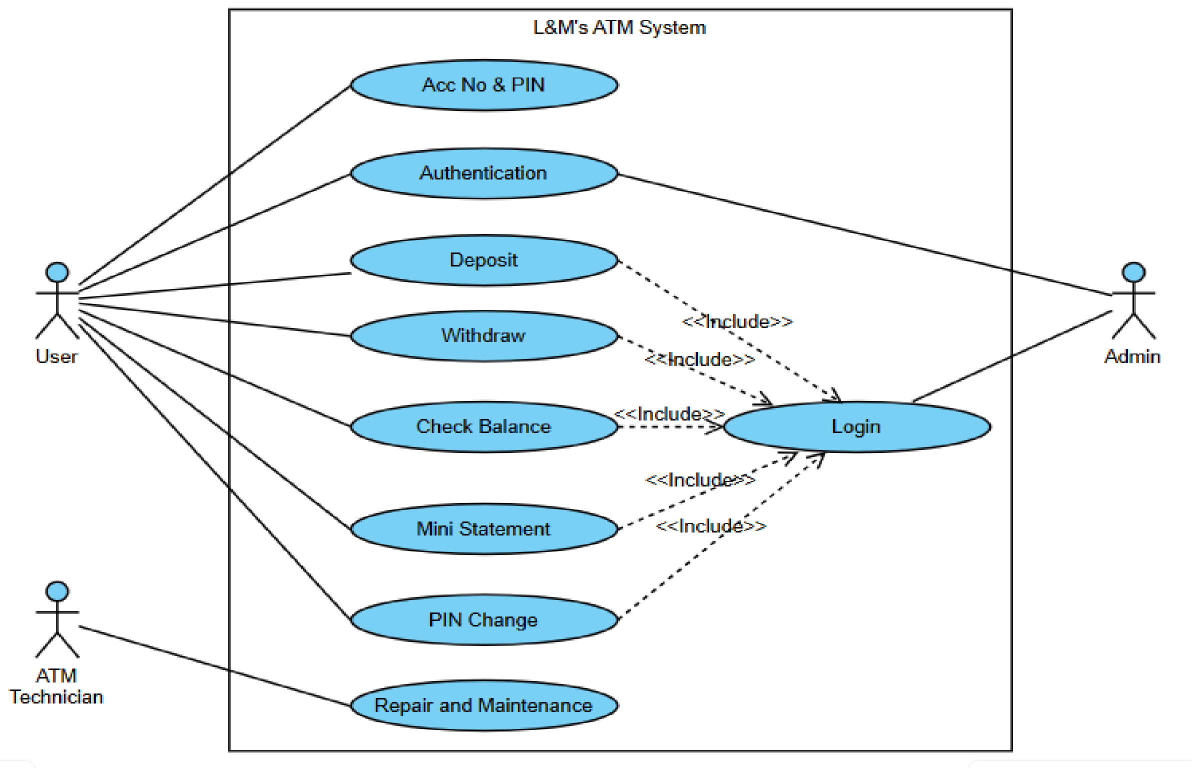
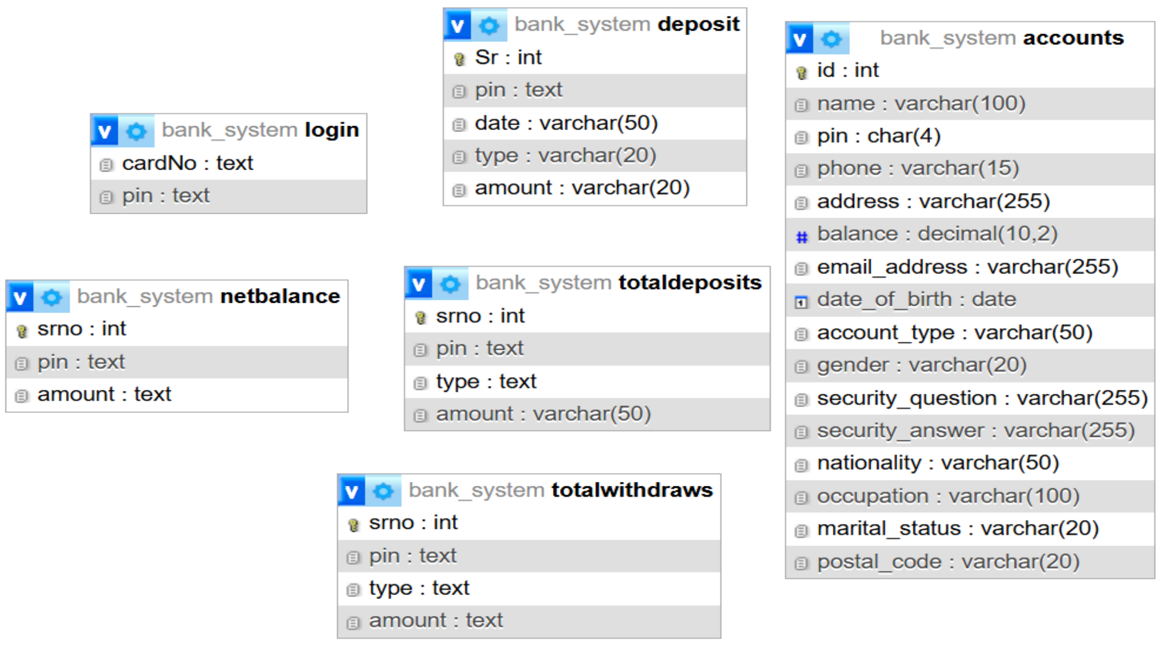


Figure 3: Use Case Diagram

Database Design – (Without Registration)



bank_system **deposit**
Sr : int
pin : text
date : varchar(50)
type : varchar(20)
amount : varchar(20)

bank_system **accounts**
id : int
name : varchar(100)
pin : char(4)
phone : varchar(15)
address : varchar(255)
balance : decimal(10,2)
email_address : varchar(255)
date_of_birth : date
account_type : varchar(50)
gender : varchar(20)
security_question : varchar(255)
security_answer : varchar(255)
nationality : varchar(50)
occupation : varchar(100)
marital_status : varchar(20)
postal_code : varchar(20)

bank_system **login**
cardNo : text
pin : text

bank_system **netbalance**
srno : int
pin : text
amount : text

bank_system **totaldeposits**
srno : int
pin : text
type : text
amount : varchar(50)

bank_system **totalwithdraws**
srno : int
pin : text
type : text
amount : text

Figure 4:Database Design (With Hidden Dependencies through PIN)

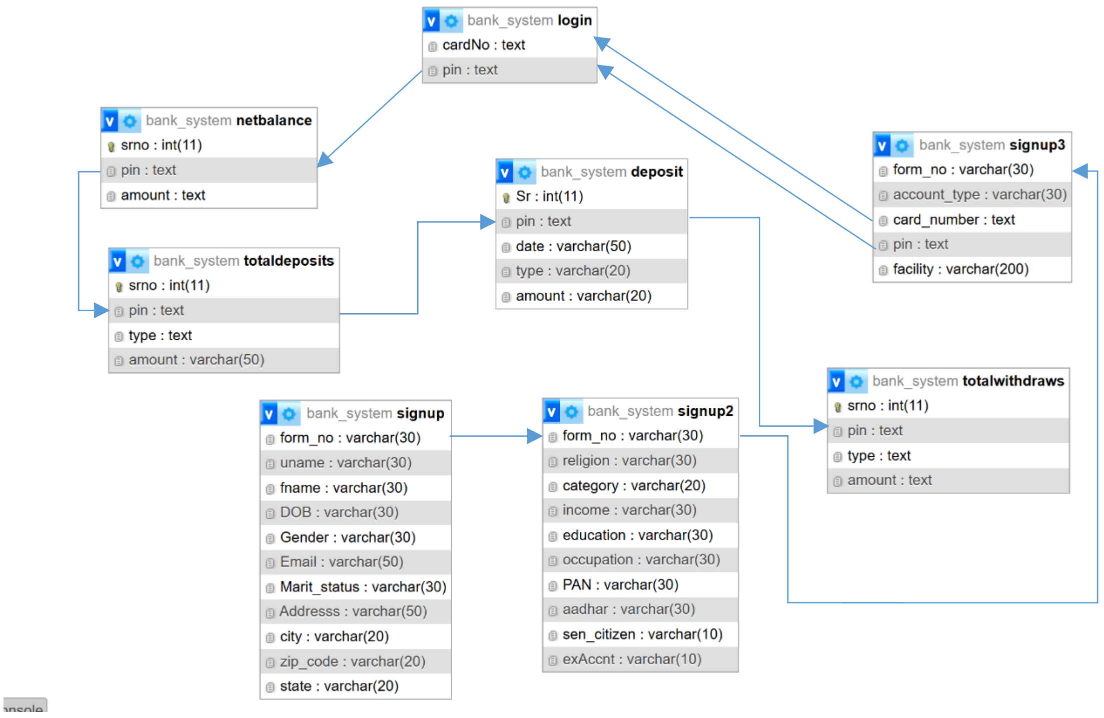


Figure 5: Database Design (with Registration of New Customer)

(***Note:** The Blue Arrows Showing dependencies are Hidden Dependencies for the purpose of Security and the PIN here is actually Hashed PIN. **)**

Deposit(Main tansaction Log)

Column	Type
Sr	Int
pin	Text
date	Varchar(50)
type	Varchar(20)
amount	Varchar(20)

Deposit(Main transaction Log)

Sr	Pin	date	type	amount
1	cf866614b6b18cda13fe699a3a65661b	Fri Oct 18 23:11:51 IST 2024	DEPOSIT	5000.00
2	cf866614b6b18cda13fe699a3a65661b	Tue Oct 22 20:17:02 IST 2024	DEPOSIT	2000.00
12	cf866614b6b18cda13fe699a3a65661b	Wed Oct 23 00:00:38 IST 2024	WITHDRAWAL	450.00
13	122e27d57ae8ecb37f3f1da67abb33cb	Wed Oct 23 00:02:39 IST 2024	WITHDRAWAL	300.00

Login

Column	Type
card	text
pin	Text

card	Pin
4567890123456789	3b712de48137572f3849aabd5666a4e3
4496729582401836	122e27d57ae8ecb37f3f1da67abb33cb
9168427588269011	1ce3e6e3f452828e23a0c94572bef9d9
5813956490174441	cf866614b6b18cda13fe699a3a65661b

TotalDeposits

Column	Type
Srno	Int
pin	Text
type	Text
amount	Varchar(50)

Srno	Pin	type	amount
3	cf866614b6b18cda13fe699a3a65661b	DEPOSIT	8500
8	122e27d57ae8ecb37f3f1da67abb33cb	DEPOSIT	65600
9	b53477c2821c1bf0da5d40e57b870d35	DEPOSIT	9050

Total Withdrawals

Srno	Pin	type	amount
4	cf866614b6b18cda13fe699a3a65661b	WITHDRAWAL	900
5	122e27d57ae8ecb37f3f1da67abb33cb	WITHDRAWAL	25800

Column	Type
Srno	Int
pin	Text
type	Text
amount	Text

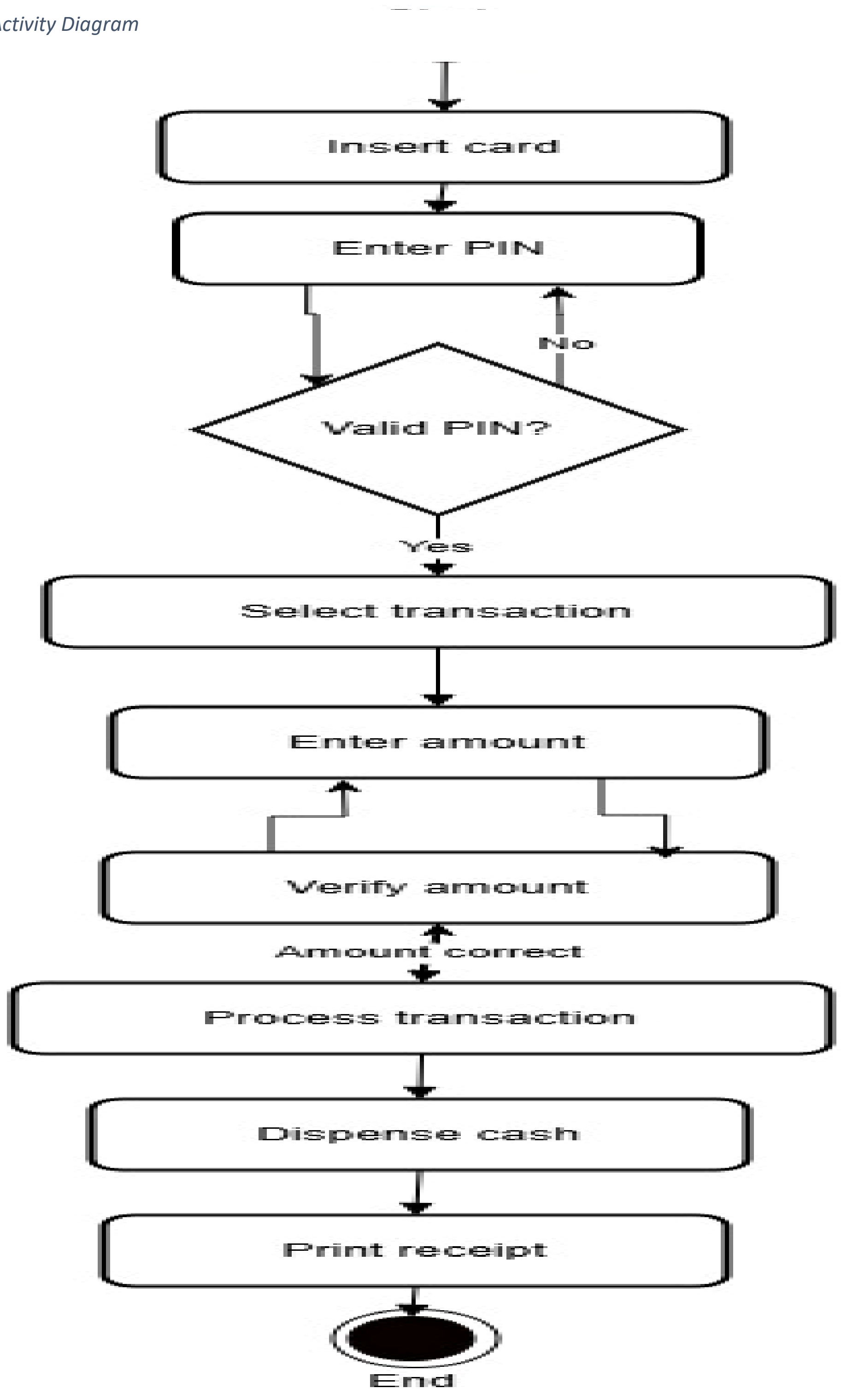
Net Balance

Srno	pin	amount
4	cf866614b6b18cda13fe699a3a65661b	7600.0
5	122e27d57ae8ecb37f3f1da67abb33cb	39800.0

Column	Type
Srno	Int
pin	Text
amount	Text

Activity Diagram

Figure 6:Activity Diagram



Level 0 Data-Flow Diagram

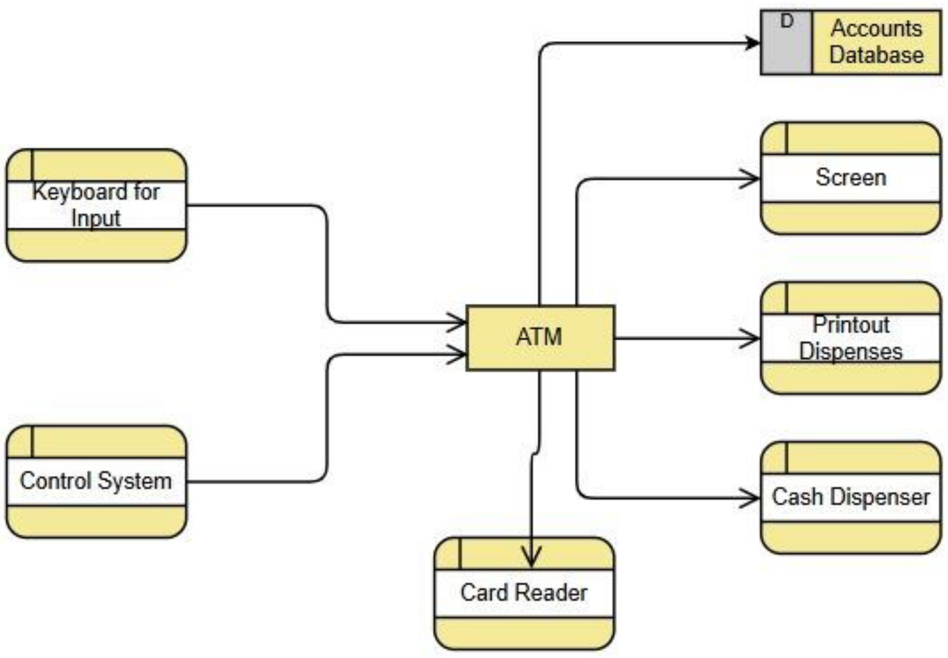


Figure 7: Data Flow Diagram (Level 0)

Level 1 Data Flow Diagram

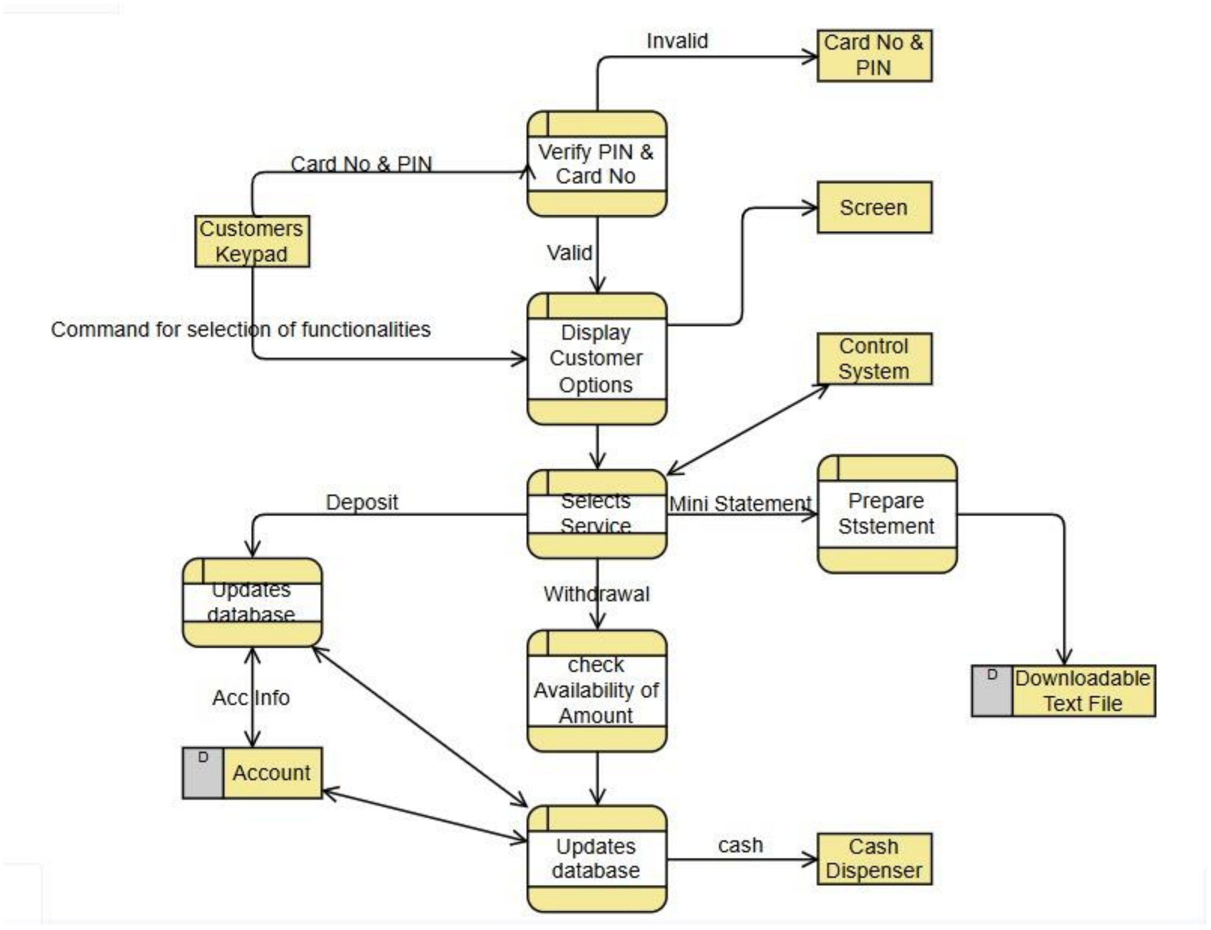


Figure 8:Level 1 Data flow Diagram

System Security

The **ATM and Banking System** is designed with robust security measures to ensure the safety and confidentiality of sensitive financial data and transactions. Security is a critical aspect of this project, given its application in managing customer accounts and financial operations. The implemented security mechanisms aim to protect against unauthorized access, data breaches, and fraudulent activities.

1. Authentication Mechanisms

The system employs secure authentication methods to verify the identity of users before granting access:

- **Customer Authentication**:
    - Users log in with their username number and a secure Personal Identification Number (PIN).
    - Password hashing ensures that even if the database is compromised, sensitive credentials remain protected.

2. Data Encryption

Data security is reinforced through encryption techniques:

- **Data**: Sensitive information, such as account details and transaction records, is encrypted using advanced encryption algorithms like MD5 (message-digest).

3. Access Control

Role-based access control ensures that users have access only to the functionalities relevant to their role:

- **Customers** can perform transactions, view balances, and manage their accounts.
- **Administrators** have additional privileges, such as monitoring transactions, and managing user accounts.

3. Database Functional Dependencies

Functional Denpendencies created through HIDDEN integration of PIN(Hashed)

4. Audit Logging

The system maintains detailed logs of all activities to track user actions and detect any suspicious behaviour:

- **Transaction Logs**: Every deposit, withdrawal, and transfer is recorded with timestamps, account details, and amounts.
- **System Access Logs**: Tracks logins, and administrative operations.
- These logs can be used for monitoring, forensic analysis, and generating alerts in case of anomalies.

5. System Security Testing

Rigorous testing was performed to validate the system's security:

- **Stress Testing**: The system was tested under heavy transactional loads to ensure resilience and maintain security protocols under pressure.

System Testing

System testing ensures that the **ATM and Banking System** functions as intended, meets its requirements, and performs reliably under various conditions. The testing process validates the system's functionalities, identifies potential defects, and verifies compliance with security and performance standards.

1. Objectives of System Testing

- To ensure all functionalities, such as cash withdrawal, fund transfer, and account management, work correctly.
- To validate the system's performance under normal and peak loads.
- To verify that the system meets security and usability standards.
- To identify and fix defects before deployment.

2. Testing Schemes Applied

Various testing schemes were employed to ensure comprehensive validation of the system.

a. Functional Testing

- **Objective**: To validate that the system performs all intended functions correctly.
- **Test Cases**:
  - Successful login for customers and administrators.
  - Performing transactions like withdrawals, deposits, and fund transfers.
  - PIN changes and password resets.

    - Generating reports for administrators.

b. Integration Testing

- **Objective**: To ensure that different components of the system work seamlessly together.
- **Test Cases**:
    - Interaction between the user interface, business logic, and database.
    - Real-time updates of account balances after transactions.

c. Performance Testing

- **Objective**: To evaluate the system's performance under different conditions.
- **Test Cases**:
    - Simulating multiple concurrent user transactions.
    - Measuring response times for account queries and transaction processing.

d. Security Testing

- **Objective**: To identify and address vulnerabilities in the system.
- **Test Cases**:
    - Attempting unauthorized access to accounts.
    - Validating encryption of sensitive data like passwords and transaction details.
    - Testing the robustness of the authentication mechanism.

e. User Acceptance Testing

- **Objective**: To ensure the system meets user expectations.
- **Methodology**: Customers and administrators tested the system in a controlled environment to confirm usability and functionality.

3. Sample Test Cases

Test Case ID	**Test Scenario**	**Expected Outcome**	**Result**
TC1	Customer login with valid credentials	User is logged in successfully.	Passed
TC2	Fund transfer between two valid accounts	Funds are transferred and balances updated.	Passed
TC3	Withdrawal exceeding account balance	Transaction is denied with an error message.	Passed

TC4	Administrator generates daily transaction report	Report is generated correctly.	Passed
TC5	Login attempt with incorrect PIN	Access is denied with an error message.	Passed

4. Results and Conclusion

The system underwent rigorous testing across all scenarios and passed all critical test cases successfully. The results demonstrate that the **ATM and Banking System** is functionally complete, secure, and ready for deployment in a real-world banking environment.

Sample Coding

Customer Login

```
try {
    if (e.getSource() == Btn1) {
        if (textField2.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please Fill all the Fields");
        } else {
            Conn con1 = new Conn();
            con1.ConnectMain();
            String HashedPIN = PIN_hashing(pin);
            this.pin = HashedPIN;
            //pin = hash pin
            int row_count = con1.loginCheck(card_no, this.pin);
//              int row_count = con1.loginCheck(card_no,PIN);
            if (row_count == 1){
                JOptionPane.showMessageDialog(null, "Login Success press OK to Continue");
                Locale currentLocale = Locale.ENGLISH;
//                  new MainScreen(HashedPIN,currentLocale);
                new MainScreen(this.pin,currentLocale);
                setVisible(false);
            }
            else{
                JOptionPane.showMessageDialog(null, "Login Failed Please Enter Right Credentials");
                textField2.setText("");
                passField3.setText("");
            }
        }
    } else if (e.getSource() == Btn2) {
        textField2.setText("");
        passField3.setText("");
    } else if (e.getSource() == Btn3) {
        new register();
        setVisible(false);
    }
} catch (Exception E) {
    E.printStackTrace();
}
```

Withdrawal

```
public void withdraw(String pin, String amount, Date date) throws SQLException {
    this.ConnectMain();
    PreparedStatement p1 = this.con.prepareStatement("INSERT INTO `deposit`(pin,date,type,amount)
VALUES ('" + pin + "', '" + date + "', 'WITHDRAWAL', '" + amount + "')");
    int status = p1.executeUpdate();

    if (status > 0) {
        System.out.println("Withdrawal Data inserted Successfully");
    } else {
        System.out.println("Insertion Failed!");
    }

    String select1 = "SELECT * FROM totalwithdraws WHERE pin = '" + pin + "' AND type =
'WITHDRAWAL'";
    this.preparedStatement = this.con.prepareStatement(select1);
    preparedStatement.executeQuery(select1);

    int rows = 0;
    resultSet = preparedStatement.executeQuery();
    while (resultSet.next()) {
        rows += 1;
    }
    if (rows >= 1) {
        String update = "UPDATE `totalwithdraws` SET `amount` = (SELECT SUM(amount) FROM deposit
WHERE pin = '" + pin + "' AND type = 'WITHDRAWAL') WHERE pin = '" + pin + "'";
        this.preparedStatement = this.con.prepareStatement(update);
        preparedStatement.executeUpdate();
    } else {
        String insertWithdraw = "INSERT INTO `totalwithdraws` (pin, type, amount) VALUES ('" + pin + "', '"
+ "WITHDRAWAL" + "', '" + "0.0" + "')";
        this.preparedStatement = this.con.prepareStatement(insertWithdraw);
        preparedStatement.executeUpdate();
    }
}
```

Deposit

```
public void depositInsert(String pin, String amount, Date dateTime) throws SQLException {
        this.ConnectMain();
        this.preparedStatement = this.con.prepareStatement("INSERT INTO `deposit` (pin,date,type,amount)
VALUES ('" + pin + "', '" + dateTime + "', '" + "DEPOSIT" + "', '" + amount + "')");
        int status = preparedStatement.executeUpdate();
        if (status > 0) {
            System.out.println("deposit Data inserted Successfully");

        } else {
            System.out.println("Insertion Failed!");
        }

        String select1 = "SELECT * FROM totaldeposits WHERE pin = '" + pin + "' AND type = 'DEPOSIT'";
        this.preparedStatement = this.con.prepareStatement(select1);
        preparedStatement.executeQuery(select1);

        int rows = 0;
        resultSet = preparedStatement.executeQuery();
        while (resultSet.next()) {
            rows += 1;
        }
        if (rows >= 1) {
            String update = "UPDATE `totaldeposits` SET `amount` = (SELECT SUM(amount) FROM deposit
WHERE pin = '" + pin + "' AND type = 'DEPOSIT') WHERE pin = '" + pin + "'";
            this.preparedStatement = this.con.prepareStatement(update);
            preparedStatement.executeUpdate();
        } else {
//          String insert = "INSERT INTO totaldeposits (pin, type, amount)\n" +
//                  "SELECT pin, type, amount\n" +
//                  "FROM deposit\n" +
//                  "WHERE pin = '" + pin + "' AND type = 'DEPOSIT';\n";
            String insert = "INSERT INTO totaldeposits (pin, type, amount) VALUES
('"+pin+"','"+"DEPOSIT"+"','"+amount+"')";
            this.preparedStatement = this.con.prepareStatement(insert);
            preparedStatement.executeUpdate(insert);

            String insertWithdraw = "INSERT INTO `totalwithdraws` (pin, type, amount) VALUES ('" + pin + "',
'" + "WITHDRAWAL" + "', '" + "0.0" + "')";
            this.preparedStatement = this.con.prepareStatement(insertWithdraw);
            preparedStatement.executeUpdate(insertWithdraw);
        }
    }
```

Balance Enquiry

```
public void insertIntoNetbalance(String pin) throws SQLException {
    this.ConnectMain();
    PreparedStatement q1 = this.con.prepareStatement("SELECT (amount) FROM totaldeposits where pin =
'" + pin + "'");
    ResultSet rs_deposit = q1.executeQuery();
    String deposit_amt = "";
    while (rs_deposit.next()) {
        deposit_amt = rs_deposit.getString("amount");
    }
    double Total_depositAMT = Double.parseDouble(deposit_amt);

    PreparedStatement q2 = this.con.prepareStatement("SELECT (amount) FROM totalwithdraws where pin
= '" + pin + "'");
    ResultSet rs_withdraw = q2.executeQuery();
    String withdraw_amt = "";
    while (rs_withdraw.next()) {
        withdraw_amt = rs_withdraw.getString("amount");
    }

    double Total_withdrawAMT = Double.parseDouble(withdraw_amt);



    double netBalance = Total_depositAMT - Total_withdrawAMT;
    String netBal = Double.toString(netBalance);

    String select = "SELECT * FROM netbalance WHERE pin = '" + pin + "'";
    this.preparedStatement = this.con.prepareStatement(select);
    preparedStatement.executeQuery(select);

    int rows = 0;
    resultSet = preparedStatement.executeQuery();
    while (resultSet.next()) {
        rows += 1;
    }
    if (rows == 1) {
        String update = "UPDATE `netbalance` SET `amount` = '" + netBal + "' WHERE pin = '" + pin + "'";
        this.preparedStatement = this.con.prepareStatement(update);
        preparedStatement.executeUpdate();
    } else if (rows < 1) {
        PreparedStatement q3 = this.con.prepareStatement("INSERT INTO `netbalance` (pin,amount)
VALUES ('" + pin + "','" + netBal + "')");
        int status = q3.executeUpdate();
        if (status > 0) {
            System.out.println("net balance Data inserted Successfully");
        } else {
            System.out.println("Insertion Failed!");
        }
    }

}
```

Transaction Logging

```
public void MiniStatement(String pin, JTextArea ta) throws SQLException {
        String netbl = check_Bal(pin);
        String select1 = "SELECT l.cardNo as cardno, dep.date as date, dep.type as type, dep.amount as amt "
+
                "FROM login l, deposit dep " +
                "WHERE l.pin = dep.pin AND l.pin = '"+pin+"'";

        this.preparedStatement = this.con.prepareStatement(select1);
        ResultSet rs = preparedStatement.executeQuery();

// Use a StringBuilder to accumulate results
        StringBuilder l1Content = new StringBuilder();
        boolean firstRecord = true; // Flag to manage the first record display

        while (rs.next()) { // Loop through all records
            if (firstRecord) {
                // Set card number, masking it appropriately for the first record
                ta.append("Card Number: " + rs.getString("cardno").substring(0, 4) + "XXXXXXXX" +
rs.getString("cardno").substring(12) + "\n\n\n");
                firstRecord = false; // Change flag after first record
            }
            // Append the current record's details to the StringBuilder
            l1Content.append(rs.getString("date")).append("    ")
                    .append(rs.getString("type")).append("    ")
                    .append(rs.getString("amt")).append("\n\n");
        }
```

Future Work/Enhancements

Multilanguage Support

Implement support for multiple languages to cater to a diverse user base.

In-App Feedback Forms

Implement forms within the application to allow users to submit feedback easily.

Fund Transfers

Enable users to transfer funds between accounts and to other banks

Bill Payments

Integrate bill payment features for utilities, loans, and other recurring payments.

Bibliography

Reference Websites

Diagram illustrations

draw.io

Visualparadigm-online

Image Template

Freepik.com