**Project Document:ATM Simulation System project**

Table of Contents

1. Introduction

1.1. Purpose of the Document

1.2. Project Overview

1.3. Scope

2. System Requirements

2.1. Functional Requirements

3. Architecture

3.1. High-Level Architecture

3.2. Class Diagram

3.3. Sequence Diagrams

4. User Interface

5. Technologies Used

6. Testing

6.1. Test Cases

6.2. Unit Testing

7. Conclusion

8. References

**1. Introduction**

**1.1. Purpose of the Document**

The purpose of this document is to provide an overview of the ATM Simulation System project developed using Core Java. It outlines the system's requirements, architecture, features, user interface, testing procedures.

**1.2. Project Overview**

The ATM Simulation System is a desktop-based application that allows users to withdraw money from their account, transfer money to another account, account management (PIN Change, PIN reset), view account balance, and more. The system aims to provide a secure and user-friendly interface for users to access ATM capabilities.

Note: for core java track participants will do this project as console-based event driven code.

**1.3. Scope**

The scope of the ATM Simulation System project includes the following functionalities:

- User PIN entering (assume card swipe as external to the system).

- Withdrawal processing.

- Transfer processing

- PIN Change, PIN Reset

- Display balance

**2. System Requirements**

**2.1. Functional Requirements**

1. User PIN auth: Users should be able enter PIN and authentication, (assume PIN is pre-provisioned in the DB). Load account post authentication.

2. Withdrawal processing: accept amount, validate amount, accept denomination, disburse

4. transfer processing. accept amount, validate amount, accept recipient account, validation, debit, credit

5. PIN management (reset PIN)

6. Display balance

**3. User Interface**

\*(Briefly describe the CLI Options.)\*

**4. Technologies Used**

\*(List the programming languages, libraries, frameworks, and tools used to develop the system.)\*

**5. Testing**

**5.1. Test Cases**

\*(Provide a set of test cases that will be executed during the testing phase. Include positive and negative test scenarios.)\*

**6. Conclusion**

\*(Summarize the key points and goals achieved through the development of the Online Banking System.)\*

**7. References**

\*(Include any references to external resources, documentation, or libraries used during the project.)\*

**1. Introduction**

**1.1. Purpose of the Document**

The purpose of this document is to provide an overview of the ATM Simulation System project developed using Core Java. It outlines the system's requirements, architecture, features, user interface, and testing procedures.

**1.2. Project Overview**

The ATM Simulation System project is a Core Java-based application designed to simulate the functionalities of an Automated Teller Machine (ATM). The system aims to replicate real-world ATM operations, providing users a virtual experience of withdrawing, and managing their bank accounts. The ATM will service one customer at a time. A customer will be required to have a personal identification number (PIN) – both of which will be sent to the database for validation as part of each transaction. The customer will then be able to perform one or more transactions. Also, customers must be able to make a balance inquiry of any account linked to the card.

**1.3. Scope**

The scope of the ATM Simulation System project includes the following functionalities:

- User PIN entering (assume card swipe as external to the system).

- Withdrawal processing.

- Transfer processing

- PIN Change, PIN Reset

-Transaction history

- Display balance

**2.1.1. System Requirements:** The system must fulfill the following requirements:

* User authentication: Validate users through PIN authentication.
* Account management: Support account balance inquiries, withdrawals, and transfers.
* Transaction history: Maintain a record of user transactions.
* Multiple accounts: Handle transactions for different bank accounts.
* User-friendly interface: Provide an intuitive graphical user interface (GUI).
* Security: Ensure secure transactions and protect user data.
* Error handling: Address potential errors and edge cases gracefully.

**2.1.2 Functional Requirements:** The ATM Simulation System Users offers the following features:

* Account Balance Inquiry: Users can check their account balance.
* Cash Withdrawal: Users can withdraw a specified amount of money.
* Cash transfer: Users can transfer funds into their accounts.
* Change PIN: Users can change their PIN for security purposes.
* Transaction History: can view their recent transaction history.
* Exit: Users can safely exit the system.

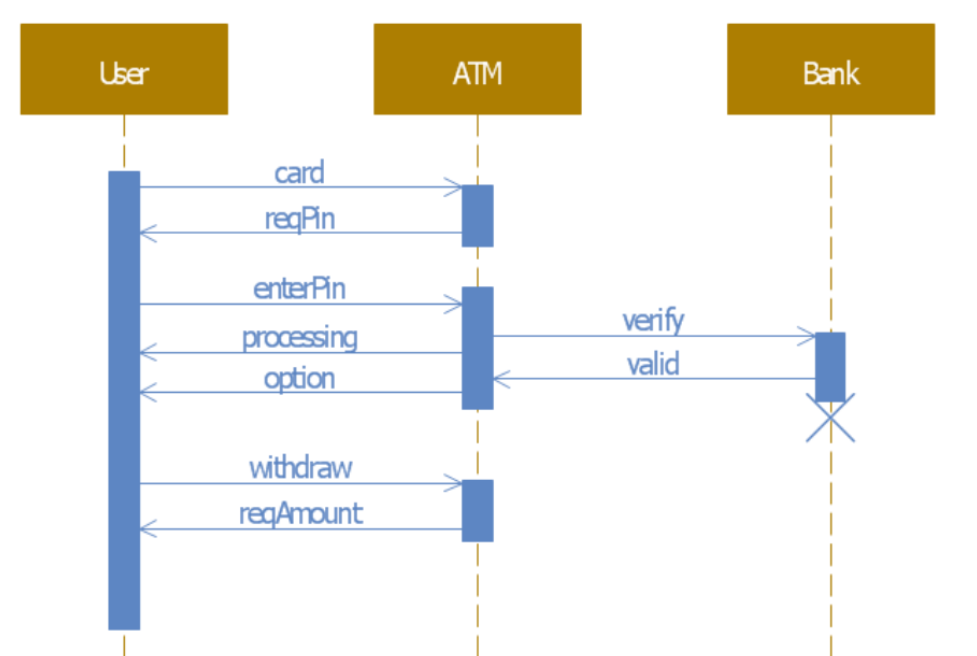
**2.1.3. System Architecture**

The ATM Simulation System follows a layered architecture:

* Presentation Layer: Handles the user interface and interaction.
* Business Logic Layer: Manages core ATM functionalities and communicates with the database.
* Data Access Layer: Connects to the database to retrieve and update user account information.

**3. Architecture**

**Sequence Diagrams**



**4. User Interface:**

Here's a brief description of the CLI options available in the project:

* Account Authentication:
  + Users are prompted to enter their account number and PIN to authenticate themselves.
  + The system verifies the entered credentials against the database.
* Main Menu:
  + After successful authentication, users are presented with a main menu.
  + The main menu lists available transactions and options.
* Balance Inquiry:
  + Users can select the "Balance Inquiry" option to view their current account balance.
  + The system fetches the account balance from the database and displays it to the user.
* Cash Withdrawal:
  + Users can choose the "Cash Withdrawal" option to initiate a withdrawal.
  + They enter the amount they wish to withdraw.
  + The system checks if the withdrawal is within the available balance and updates the account balance accordingly.
* Cash Transfer:
  + Users can select the "Cash transfer" option to transfer funds into their accounts.
  + They input the amount they want to transfer.
  + The system adds the transfered amount to the account balance.
* Change PIN:
  + Users can change their PIN by selecting the "Change PIN" option.
  + They enter their current PIN and then provide a new PIN for security purposes.
  + The system updates the PIN in the database.
* Transaction History:
  + Users can choose the "Transaction History" option to view their recent transactions.
  + The system retrieves and displays a list of recent transactions from the database.
* Exit:
  + Users can select the "Exit" option to safely exit the system.
  + The system closes the CLI and concludes the user's session.

The system utilizes Core Java for implementing the logic behind these options and SQL for retrieving and updating data in the database. This approach offers users a convenient and efficient means of conducting banking transactions and managing their accounts through the command line.

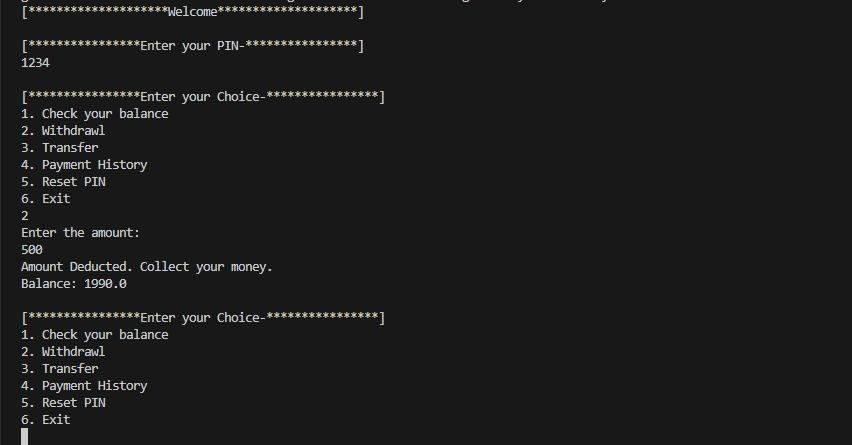
**5. Technologies Used**

* Mysql database
* Eclipse jdk and driver manager

**6. Testing Procedures:** Testing is crucial to ensure the system's functionality and reliability:

* Unit Testing: Individual components and methods are tested in isolation.
* Integration Testing: Interactions between different components are tested.
* User Acceptance Testing: Real users perform end-to-end testing.
* Security Testing: Verify the system's security measures against unauthorized access.
* Error Handling Testing: Test how the system handles unexpected scenarios.

**sample test case**

****

**7. Conclusion**: The ATM Simulation System project has effectively combined Core Java's programming capabilities with MySQL's data management features to create a comprehensive simulation of ATM operations. Through its user-friendly interface, secure data management, error handling, documentation practices, and collaborative development, the project offers an impactful demonstration of technology integration for practical purposes.

**8. References**

- MySQL Documentation

- Java Documentation

- Stack Overflow (for troubleshooting)