B.TECH. (CSE)
V Semester
UE21CS341A –Software Engineering

PROJECT REPORT
on

# *ATM-Simulator*

Submitted by :

| | |
|---|---|
| PES1UG21CS652 | Swaraj MK |
| PES1UG21CS618 | Srikrishna B |

| | |
|---|---|
| PES1UG21CS662 | Tanmay Praveen Udupa |
| PES1UG21CS659 | T P Shriambikesh |

August – Dec 2023
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BENGALURU – 560100, KARNATAKA, INDIA

*Table of Contents*

## Project Proposal

This project aims to develop an ATM simulator that can be used to learn about the basic functions of an ATM machine. The simulator will allow users to perform transactions such as cash withdrawal, cash deposit, balance inquiry, and fund transfer. It will also prevent unauthorized access. The ATM simulator will be tested using a variety of test cases. The test cases will cover all the possible scenarios that a user might encounter when using an ATM machine. The ATM simulator is a useful tool for students and people who are interested in learning about ATM machines. It can also be used by banks and financial institutions to train their employees on the use of ATM machines.

Here are some of the specific objectives of the ATM simulator project:

- To implement the functions of an ATM machine, such as cash withdrawal, balance inquiry, and fund transfer, etc.
- To implement a feature to prevent unauthorized access.
- To test the ATM simulator using a variety of test cases.

UE21CS341A - Software Engineering

# PROJECT PLAN DOCUMENT
# ATM SIMULATOR

Team #: __T_____

| | | | |
|---|---|---|---|
| PES1UG21CS652 | Swaraj MK | PES1UG21CS662 | Tanmay Praveen Udupa |
| PES1UG21CS618 | Srikrishna B | PES1UG21CS659 | T P Shriambikesh |

## Lifecycle

Lifecycle to be followed:

For an ATM Simulator project, the Agile development lifecycle is a suitable choice due to the following reasons:

Iterative Development: Agile allows for iterative development, which is beneficial for a complex project like an ATM Simulator where requirements may evolve.

Customer Collaboration: Regular feedback and involvement of stakeholders ensure that the ATM Simulator aligns with user expectations and business needs.

Flexibility: Agile methodologies provide flexibility to adapt to changing requirements, allowing the team to respond to unforeseen challenges.

Continuous Improvement: The regular retrospective meetings encourage continuous improvement in processes and product quality.

Early and Regular Delivery: Agile promotes the early and regular delivery of increments, providing tangible value to users throughout the development process.

## Tools Used for this Project

Planning Tools: Microsoft Excel

Design Tools: Draw.io, ERD Plus and Visual Paradigm

Development Tools: Flask, Python, sqlite

Version Control: Git Version Control

## Deliverables classified as reuse/build components

Determine all the deliverables and categorize them as reusable/build components, and provide justification for each categorization

Deliverables classified as reuse/build components

User interface (UI) design (Reusable component)

The UI design is a core component of the ATM simulator and can be reused in other software projects, such as other banking applications or training simulators.

Account management module (Reusable component)

The account management module is a self-contained unit that can be easily integrated into other software systems that require user account management functionality.

Balance inquiry module (Reusable component)

The balance inquiry module is a simple and straightforward component that can be reused in other banking applications or financial data management systems.

Cash withdrawal module (Reusable component)

The cash withdrawal module is a critical component of the ATM simulator and can be reused in other banking applications or financial transaction processing systems.

Cash deposit module (Reusable component)

The cash deposit module is another essential component of the ATM simulator and can be reused in other banking applications or financial transaction processing systems.

Error handling module (Reusable component)

The error handling module is a crucial component that can be reused in various software systems to effectively handle and report errors.

## Work Breakdown Structure

| Milestone description | Category | Assigned to | Progress | Start | Days |
|---|---|---|---|---|---|
| **Project Initiation** | | | | | |
| Project Kick-Off | On Track | All Members | **100%** | 9/6/2023 | 1 |
| Identify Stakeholders | On Track | All Members | **100%** | 9/7/2023 | 1 |
| **Requirements Analysis** | | | | | |
| Define Use Cases | Low Risk | Swaraj MK | **100%** | 9/10/2023 | 1 |
| Gather Requirements | Low Risk | Tanmay U | **100%** | 9/14/2023 | 1 |
| Prioritise Requirements | Med Risk | Srikrishna,Shriambhikesh | **100%** | 9/17/2023 | 2 |
| **System Design and Plan** | | | | | |
| Architecture Design | Med Risk | tanmay U, Srikrishna | **100%** | 9/19/2023 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| Database Design | Low Risk | Swaraj mk, Shriambhikesh | **100%** | 9/27/2023 | 4 |
| User Interface Design | Low Risk | All Members | **100%** | 9/28/2023 | 5 |
| Security Planning | High Risk | All Members | **100%** | 10/1/2023 | 6 |
| Infrastructure Planning | Low Risk | Shriambhikesh | **100%** | 10/6/2023 | 2 |
| **Development : 5)** | **1) Frontend(first 3),** | **2) Backend(last** | | | |
| Implement User Login | High Risk | All Members | **100%** | 10/7/2023 | 2 |
| Create user Interface | Med Risk | swaraj mk, Srikrishna | **100%** | 10/8/2023 | 2 |
| Implement transaction Proessing | Med Risk | tanmay U, Shriambhikesh | **100%** | 10/10/2023 | 1 |
| Implement User Database | Low Risk | tanmay U, Shriambhikesh | **100%** | 10/15/2023 | 3 |
| Configure User Database | Low Risk | swaraj mk, Srikrishna | **100%** | 10/16/2023 | 2 |
| Implement transaction Database | Med Risk | swaraj mk, Srikrishna | **100%** | 10/20/2023 | 3 |
| Configure transaction Database | Med Risk | All Members | **100%** | 10/22/2023 | 5 |
| Create views and routes | Med Risk | srikrishna | **100%** | 10/25/2023 | 4 |
| **Testing** | | | | | |
| testing frontend | Low Risk | Shriambhikesh | **100%** | 11/2/2023 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| testing backend | Low Risk | Tanmay | **100%** | 11/4/ 2023 | 2 |

## Effort Estimation (in person-months)

(1 year = 260 working days, then 1 month = 260/12 = 21.66 working days. So 7 full working days for one person would be 7/21.66 = 0.323 person-months.)

Project Initiation :

2 working days

2/21.66 = 0.0923 person-months

Requirement Analysis :

4 working days

4/21.66 = 0.184 person-months

System Design and Plan :

19 working days

19/21.66 = 0.877 person-months

Development(frontend and backend) :

22 working days

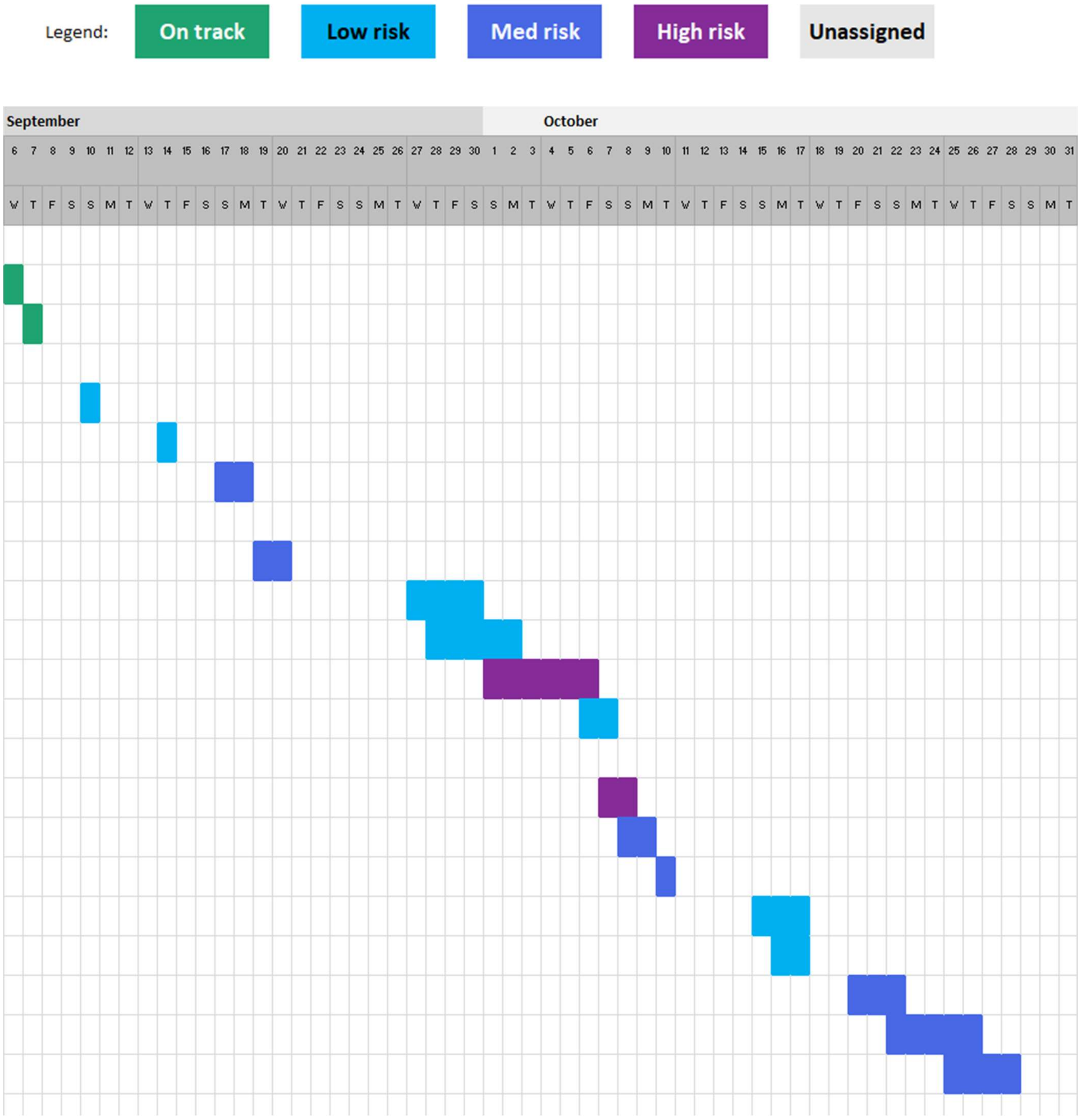
22/21.66 = 1.015 person-months

Testing :

4 working days

4/21.66 = 0.184 person-months

## Gantt Chart


Legend:
On track
Low risk
Med risk
High risk
Unassigned
September
October
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T

# Software Requirements Specification

# ATM Simulator

**Version 1.2**

**Prepared by**

**T P Shriambhikesh (PES1UG21CS659)**

**Tanmay Praveen Udupa (PES1UG21CS662)**

**Srikrishna B (PES1UG21CS618)**

**Swaraj MK (PES1UG21CS652)**

# Table of Contents

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| ATM-simulator | 7-9-2023 | Initial draft | 1.0 |
| ATM-simulator | 26-10-2023 | Modified Section 5 : System features | 1.1 |
| ATM-simulator | 16-11-2023 | Modified Appendix-A: RTM | 1.2 |

# 1. Introduction

## 1.1 Purpose

The aim of this document is to establish the software prerequisites for an Automated Teller Machine Simulator. An Automated Teller Machine is an electronic banking outlet that allows the end-users to complete basic transactions without the aid of a branch representative. This simulation software is designed to mimic the functionality of a real-world Automated Teller Machine, allowing users to perform various banking operations in a controlled environment.

## 1.2 Intended Audience

1. **Software Developers and Testers:** Used to develop, test, and debug ATM software and applications. These professionals use simulators to ensure that the software works correctly and to identify and rectify any issues or bugs.
2. **Banking and Financial Institutions:** Banks and financial institutions may use ATM simulators as training tools for their employees, including customer service representatives, tellers, and ATM maintenance personnel. This helps trainees understand how ATMs work and how to troubleshoot common issues.
3. **Educational Institutions:** Used in educational settings, such as universities, colleges, and technical schools, to teach students about banking systems, software development, and financial technology. Students can gain hands-on experience using a simulator.
4. **ATM Manufacturers and Maintenance Personnel:** Companies that manufacture ATMs and ATM components may use simulators to train their technicians and service personnel on how to install, repair, and maintain ATMs.
5. **Independent Software Developers:** Independent software developers or development teams may create ATM simulators for educational or personal use to learn about ATM systems, practice coding, or experiment with new features.
6. **General Public:** Some ATM simulators are designed for the general public to familiarize themselves with the ATM interface and process. This can help users become more comfortable with using ATMs in real-life scenarios.

### 1.3 Product Scope

**Objectives and Goals**

1. To provide users with a realistic experience of using an ATM.
2. To teach users how to perform all of the basic ATM functions, such as withdrawing cash, checking balances, and transferring funds.
3. To help users learn how to use an ATM safely and securely.
4. To gather data on how people interact with ATMs in order to develop new ATM features and services.

**Relationship to Corporate Goals or Business Strategies**

1. **Improving customer service:** ATM simulators can be used to train employees on how to use ATMs so that they can better assist customers who need help.
2. **Reducing fraud:** ATM simulators can be used to train employees and customers on how to identify and prevent ATM fraud.
3. **Increasing customer satisfaction:** ATM simulators can be used to develop new ATM features and services that meet the needs of customers.

### 1.4 References

https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

## 2. Overall Description

### 2.1 Product Perspective

The automated teller machine simulator is a self contained software product that is meant to mimic the functionality of a real-world ATM. It is not a replacement of the existing ATM but a standalone software system meant for educational and training purposes. The aim of this software system is to provide the users, including bank employees, customers and students, with a realistic platform for learning and practicing in a controlled environment.

While the ATM simulator is a self contained software product , it can be integrated into larger training or educational systems that focus on banking and financial services. The primary external interface is the user interface(UI) that allows users to interact with software.

## 2.2 Product Functions

1. User Authentication
    a. Allow users to log in securely with a username and password
    b. Implement authentication checks to ensure user's identity
2. Account Management
    a. Enable users to create, edit, and delete accounts
    b. Maintain account details, including balance and transaction history
3. Balance Inquiry
    a. Provide users with the ability to check their account balance.
4. Cash Withdrawal
    a. Allow users to withdraw cash from their accounts
    b. Validate withdrawal limits and available balance
5. Cash Deposit
    a. Allow users to deposit cash into their accounts
    b. Validate deposited amounts and updated the account balance.
6. Receipt Generation
    a. Generate transaction receipt for users
    b. Include details such as transaction data, time, and account balance.
7. Error Handling
    a. Detect and handle errors gracefully, providing informative messages to the users
8. Security Features
    a. Implement security measures to protect user data and transactions.
    b. Encrypt sensitive information and use secure communication protocol.

9. Simulation Environment
    a. Create realistic simulation of an ATM, including screen layout and interactive elements.

## 2.3 Operating Environment

1. **Hardware Platform**
    a. The ATM simulator must be designed to run on standard personal computers (PCs) and laptops.
    b. Minimum hardware requirements - a CPU with speed sufficient to run the chosen programming language and framework, adequate amount of RAM and a display screen.

2. **Operating system and version**
    a. Must be compatible with Windows 10 and above
    b. A Linux distribution with compatibility for the chosen development tools and libraries

## 2.4 User Classes and Characteristics

| User | Frequency of use | Technical Expertise | Security/privilege Levels | Educational Level |
|---|---|---|---|---|
| Regular Users (eg. Bank Employees) | Frequent Users | Varying technical expertise levels | standard user privileges | Have a basic understanding of banking operations |
| Novice Users | Infrequent or Individuals new to banking operations | Limited technical expertise | standard user privileges | students/individuals in the early learning stage of financial operations |

| | | | | |
|---|---|---|---|---|
| | | | | |

The software should prioritize providing a user-friendly experience for both frequent users and those who are new to the system, while also accommodating advanced users and administrators who require in-depth functionality and training capabilities.

## 2.5 Design and Implementation Constraints

1. **Regulatory Compliance:** The ATM simulator must adhere to regulatory standards and compliance requirements specific to the banking and financial industry.
2. **Hardware Limitations:** The software should be designed to work efficiently within the hardware constraints of typical personal computers.
3. **Security Considerations:** The software should adhere to established security protocols and encryption standards. It must implement secure user authentication, data protection, and secure communication.
4. **Parallel operations:** If the ATM simulator is designed to support multiple users, it must be able to handle multiple user sessions simultaneously without data corruption or conflicts.
5. **Performance and Response Time:** The software must meet performance expectations, including response time for user interactions. This is a critical constraint, especially in a real time constraint.

## 2.6 Assumptions and Dependencies

1. **Assumptions:**
    a. **Database availability:** It is assumed that the required database system will be available and properly configured for storing user account data. Any issues with the database system may impact the project.
    b. **Operation system stability:** The software system assumes the stability and proper functioning of the specified operating systems. Changes in these operating system environments might affect software's operation.
    c. **Compliance with regulatory standards:** It is assumed that the software will be

designed and developed to comply with relevant regulatory and security standards.

   d. **Availability of required development tools:** The availability of specified programming languages, frameworks, libraries and development tools as specified in the SRS is assumed. Any changes to tool availability may impact the development process.

2. **Dependencies**
   a. **Third-Party Libraries and Components:** The project may rely on third-party libraries and components for specific functionalities (e.g., security libraries, GUI frameworks, and database connectors). Any changes or issues with these dependencies can affect the project's development and functionality.
   b. **Hardware and Network Infrastructure:** The software's performance and functionality depend on the hardware and network infrastructure of the host computer. The project is dependent on the availability and proper functioning of this infrastructure.
   c. **External Services:** If the ATM Simulator interfaces with external banking or financial services, it is dependent on the availability and stability of those services. Changes or issues with external services can impact the software's functionality.
   d. **Accessibility Standards:** If the project aims to adhere to accessibility standards (e.g., WCAG) for users with disabilities, it is dependent on the accessibility features provided by the underlying operating systems and technologies.

## 3. External Interface Requirements

### 3.1 User Interfaces

The ATM simulator software should have a user interface that is easy to use and navigate. It should use standard buttons and functions that users are familiar with, such as "Help", "Back", and "Next". The interface should support keyboard shortcuts for common tasks, such as withdrawing cash or transferring money. Error messages should be displayed in a clear and concise way, and provide users with instructions on how to resolve the error.

The following are some software components for which a user interface is needed:

1. **Main screen:** This screen should allow users to select from the most common ATM transactions, such as withdrawing cash, depositing cash, and checking their balance.
2. **Withdraw cash screen:** This screen should allow users to enter the amount of cash they want to withdraw, and select the type of account they want to withdraw from.
3. **Deposit cash screen:** This screen should allow users to enter the amount of cash they want to deposit, and select the type of account they want to deposit into.
4. **Check balance screen:** This screen should display the user's current account balance.

## 3.2 Software Interfaces

The ATM simulator software will interact with the following software components:

1. **Database:** The ATM simulator will interact with a database to store and retrieve customer account information.
2. **Operating system:** The ATM simulator will run on an operating system such as Windows or Linux.
3. **Tools:** The ATM simulator will use a variety of tools to develop and deploy the software, such as a compiler, a linker, and a debugger.
4. **Libraries/Frameworks:** The ATM simulator will use a variety of libraries and frameworks to provide common functionality, such as a GUI library and a database library.
5. **Integrated commercial components:** The ATM simulator may use integrated commercial components to provide additional functionality, such as a fraud detection component.

The following data items or messages will come into the ATM simulator system:

1. **Customer account information:** This information will be retrieved from the database when the customer inserts login credentials.
2. **Transaction requests:** The customer will enter transaction requests through the ATM's user

interface.

The following data items or messages will go out of the ATM simulator system:

1. **Transaction results:** The ATM simulator will display the results of the customer's transaction on the ATM's screen.
2. **Transaction data:** The ATM simulator will update the database with the results of the customer's transaction.

The ATM simulator will need to communicate with the following services:

1. **Database service:** The ATM simulator will need to communicate with the database service to retrieve and update customer account information.
2. **Printer service:** The ATM simulator may need to communicate with the printer service to print receipts or other documents.

The nature of the communications between the ATM simulator and other software components will be synchronous. This means that the ATM simulator will wait for a response from the other component before proceeding.

The following data will be shared across software components:

1. **Customer account information:** This information will be shared between the ATM simulator and the database.
2. **Transaction data:** This information will be shared between the ATM simulator and the database. The data sharing mechanism will be implemented using a database. This means that the ATM simulator and the database will share a common pool of data.

## 3.3 Communications Interfaces

The ATM simulator software will require the following communications functions:

1. **Database communication:** The ATM simulator will need to communicate with a database to retrieve and update customer account information. This communication will use a database-specific protocol, such as JDBC or ODBC.
2. **Client-server communication over HTTPS:** To ensure secure data transmission between the client user and the server, HTTP or HTTPS is to be used for communication between the client and the Flask application.
3. **Electronic Forms:** HTML forms to be used to create the electronic forms for user input.

**3.4 Synchronization mechanisms:**

The ATM simulator will use a variety of synchronization mechanisms to ensure that data is shared correctly between software components. For example, the ATM simulator will use a database transaction to ensure that customer account information is updated correctly.

**3.5 Data Transfer Rates:**

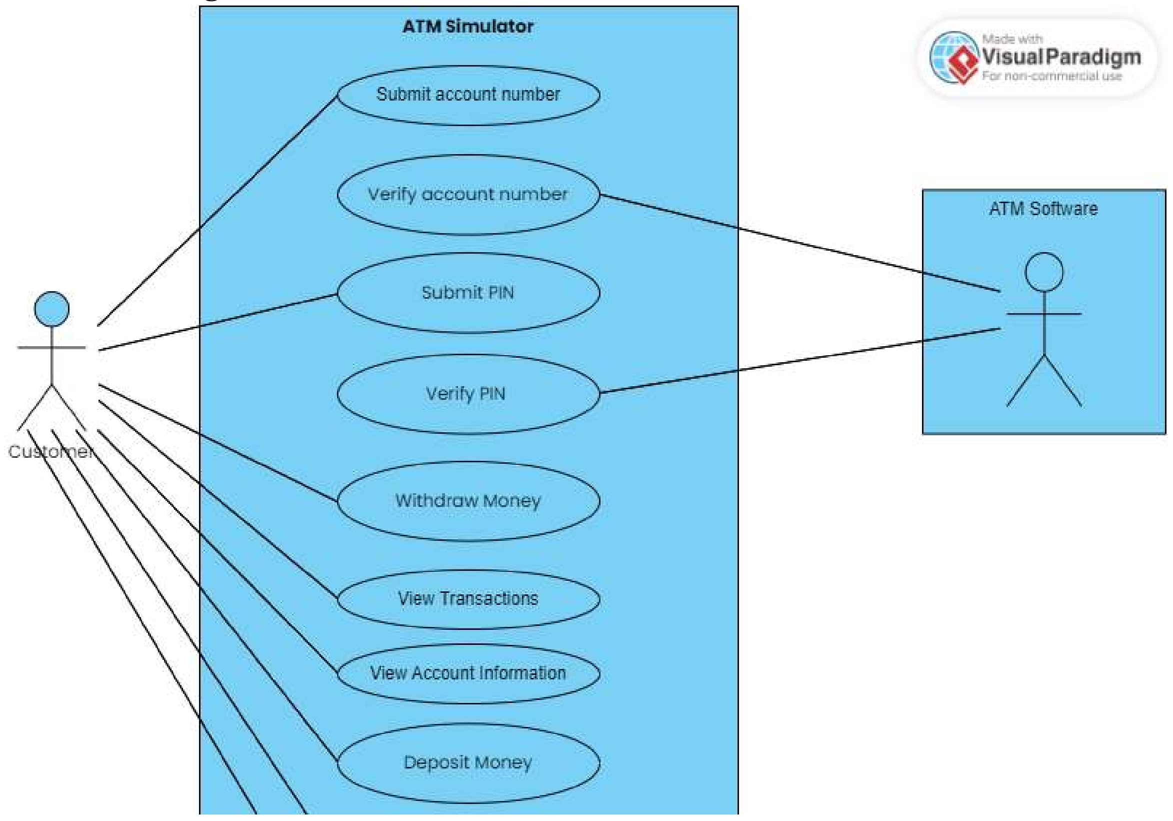
The data transfer rates will largely depend on factors such as server performance, network conditions, and database queries.

**3.6 Security considerations:**

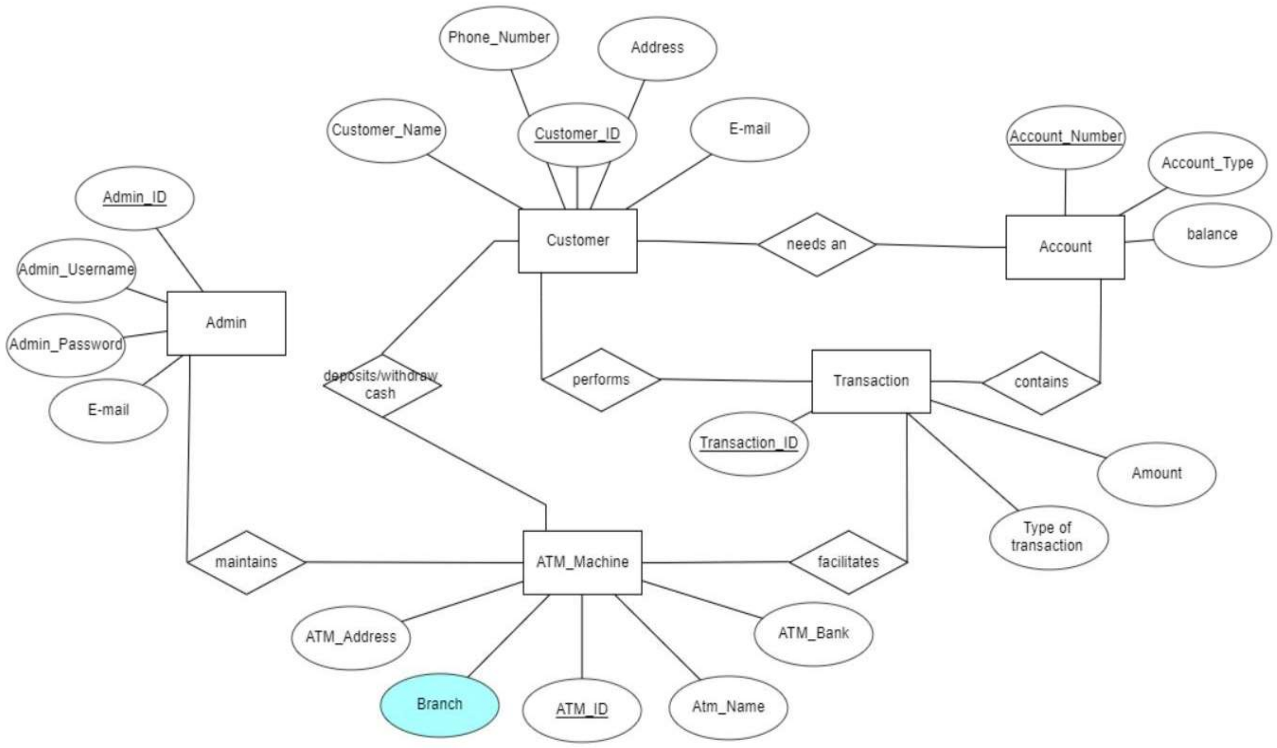
1. Using strong passwords and authentication mechanisms
2. Implementing input validation to protect against cross-site scripting attacks

# 4. Analysis Models

## 4.1 Use case diagram



## 4.2 ER Diagram



## 5. System Features

### 5.1 Customer Features

#### 5.1.1 Description and Priority

Customers can check information about their account, and perform operations from the account. (High priority)

#### 5.1.2 Stimulus/Response Sequences

User logs in to the account. Displays options to perform operations / view account information.

#### 5.1.3 Functional Requirements

1. Account Information Retrieval:
    a. Users must be able to retrieve their account balance, account number, and account type (savings or current).
    b. Customers should be able to view a history of their cash deposits and withdrawals,

showing transaction details, dates, and amounts.
   c. If the account doesn't exist, the system should provide a clear error message.

2. Withdrawal from Account:
   a. Users should be able to withdraw funds from their account.
   b. The system must validate that the withdrawal amount is within the available balance.
   c. If the withdrawal amount exceeds the balance, the system should provide an error message and prevent the transaction.
   d. After a successful deposit, the system should generate a receipt that includes transaction details, date, time, and a reference number.
3. Deposit to account:
   a. Customers should be able to deposit cash into their savings or current account by specifying the deposit amount.
   b. The system should validate the deposit amount, ensuring it's not negative or zero.
   c. Invalid or negative deposit amounts should result in an error message.
   d. After completing the deposit transaction, the system should display a confirmation message showing the deposited amount and updated account balance.
   e. There may be daily or per-transaction deposit limits for customers. The system should enforce these limits and provide an error message if exceeded.
   f. After a successful deposit, the system should generate a receipt that includes transaction details, date, time, and a reference number.

## 6. Other Non Functional Requirements

### 6.1 Performance Requirements

1. **Response Time:** The system should respond to user interactions within 2 seconds for standard operations such as balance inquiry and cash withdrawal.
2. **Concurrent User Capacity:** The ATM simulator should support a minimum of 100 concurrent users without experiencing significant performance degradation.

## 6.2 Safety Requirements

1. User Data Protection
    a. The system must employ data encryption to protect user PINs and other sensitive information.
    b. In the event of an unexpected system crash, all transaction data should be saved and recoverable upon system restart.

## 6.3 Security Requirements

1. Authentication
    a. User authentication must be based on a secure method, such as two-factor authentication or strong password policies.
2. Authorization
    a. Users should have access only to their own account and associated operations.
    b. Administrative functions, such as PIN reset, should be restricted to authorized personnel.
3. Audit Trail
    a. The system should maintain an audit trail of all user actions, login attempts, and other critical events.
    b. The audit trail should be securely stored and accessible only to authorized personnel.
4. Secure Communication
    a. All data transmitted between the client and server must be encrypted using industry-standard encryption protocols, such as HTTPS.

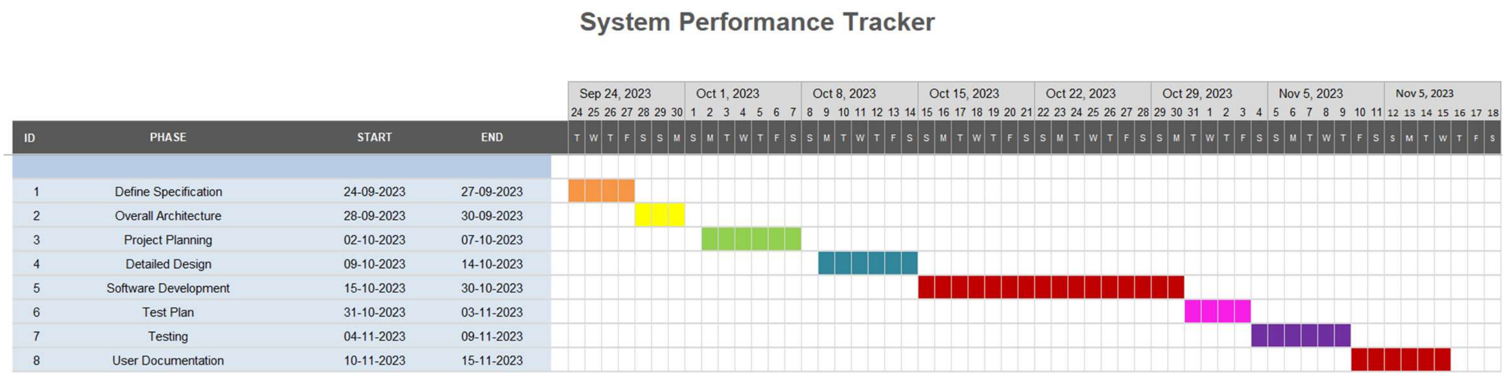
## 6.4 Software Quality Attributes

1. Reliability: The system should have a high level of reliability, with an expected uptime of at least 99.9%
2. Maintainability: The codebase should be well-documented, and software updates and maintenance should be straightforward
3. Scalability: The system should be designed to scale to accommodate future feature enhancements and user growth.

## 6.5 Business Rules

1. Transaction Limits
    a. The system should enforce transaction limits based on the user's account type and banking regulations.
    b. Users must be informed of these limits during the transaction process.
2. Logging and Reporting
    a. The system should generate daily, weekly, and monthly transaction reports for administrative purposes.
    b. Users should have access to their transaction history for a minimum of 30 days.
3. User Support
    a. The system must provide users with clear instructions and error messages.
    b. Contact information for customer support or help resources should be readily accessible within the application.

## 7. Gantt Chart



| ID | PHASE | START | END |
|---|---|---|---|
| 1 | Define Specification | 24-09-2023 | 27-09-2023 |
| 2 | Overall Architecture | 28-09-2023 | 30-09-2023 |
| 3 | Project Planning | 02-10-2023 | 07-10-2023 |
| 4 | Detailed Design | 09-10-2023 | 14-10-2023 |
| 5 | Software Development | 15-10-2023 | 30-10-2023 |
| 6 | Test Plan | 31-10-2023 | 03-11-2023 |
| 7 | Testing | 04-11-2023 | 09-11-2023 |
| 8 | User Documentation | 10-11-2023 | 15-11-2023 |

## Appendix A: Requirement Traceability Matrix

| Sl.no | Requirement | Testcase_ID | Test Result | Defect status |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | Account Information Retrieval | BAL_001 | PASS | Nil |
| 2 | Withdrawal from account | WITH_001 | PASS | Nil |
| 3 | Deposit to account | DEP_001 | PASS | Nil |
| 4 | View Transaction Logs | TRANSAC_001 | PASS | Nil |

**UE20CS303 - Software Engineering**

**DESIGN DOCUMENT**

*ATM SIMULATOR*

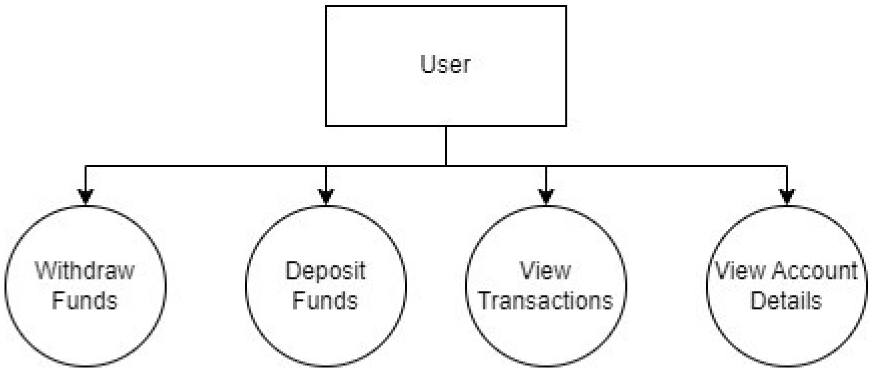
Team #: __T_____

| | | | |
|---|---|---|---|
| PES1UG21CS652 | Swaraj MK | PES1UG21CS662 | Tanmay Praveen Udupa |
| PES1UG21CS618 | Srikrishna B | PES1UG21CS659 | T P Shriambikesh |

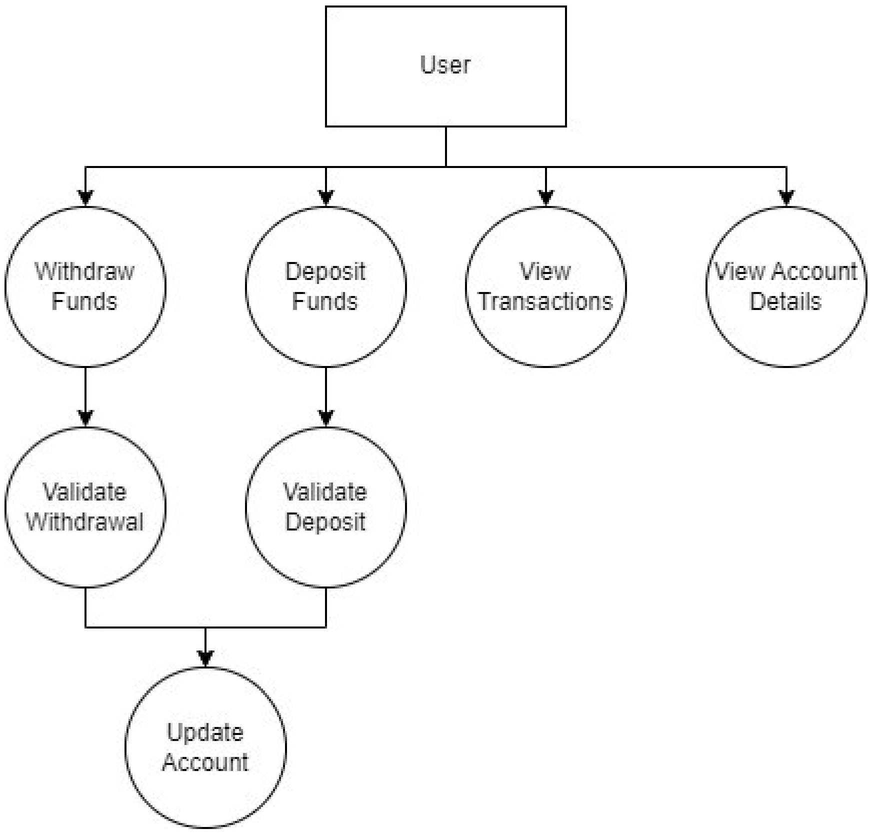
## *Design Diagrams*

### Diagrams of Levels of DFD

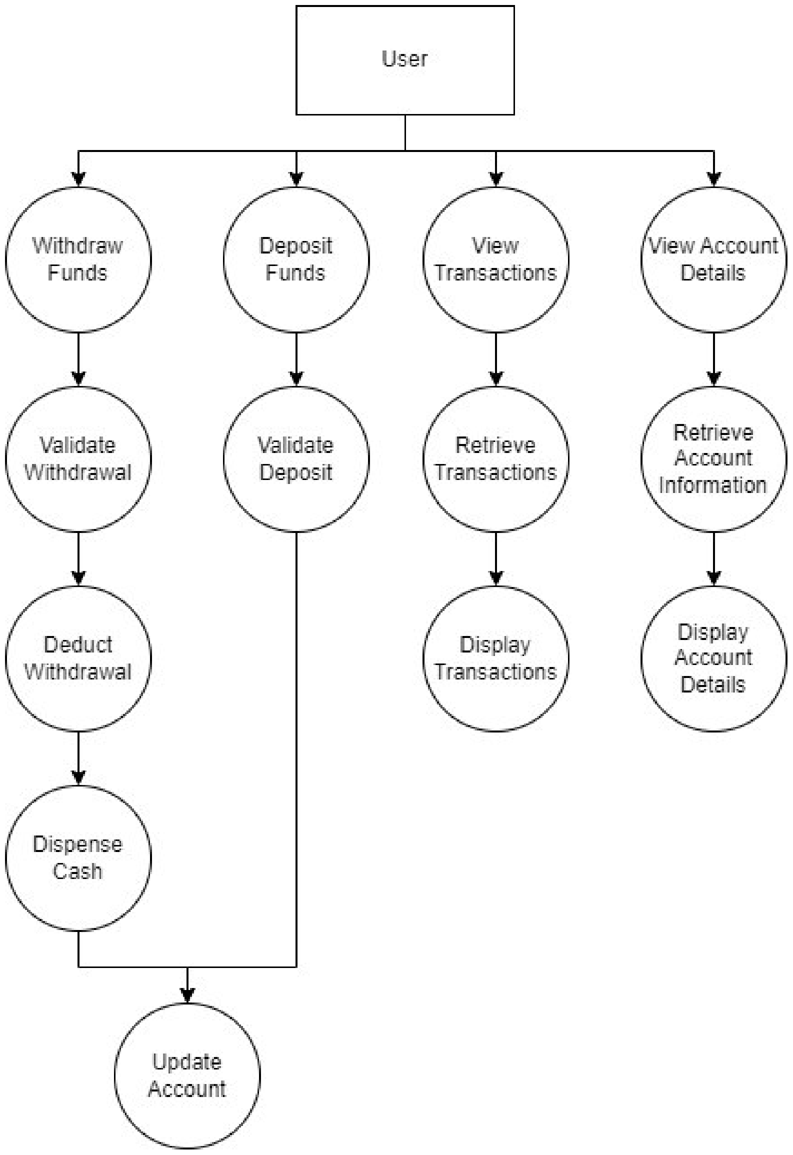
#### 0-level DFD



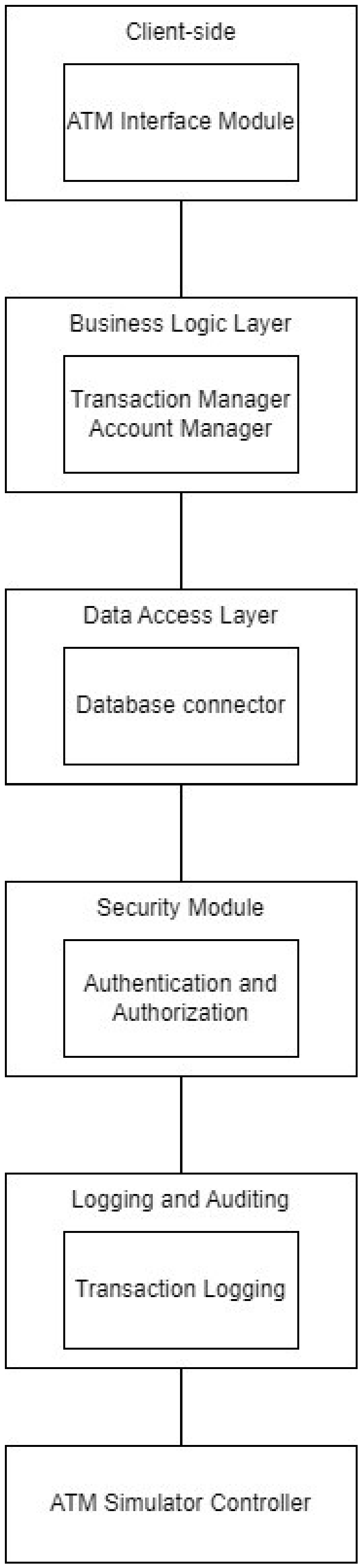
## 1-level DFD


User
Withdraw Funds
Deposit Funds
View Transactions
View Account Details
Validate Withdrawal
Validate Deposit
Update Account

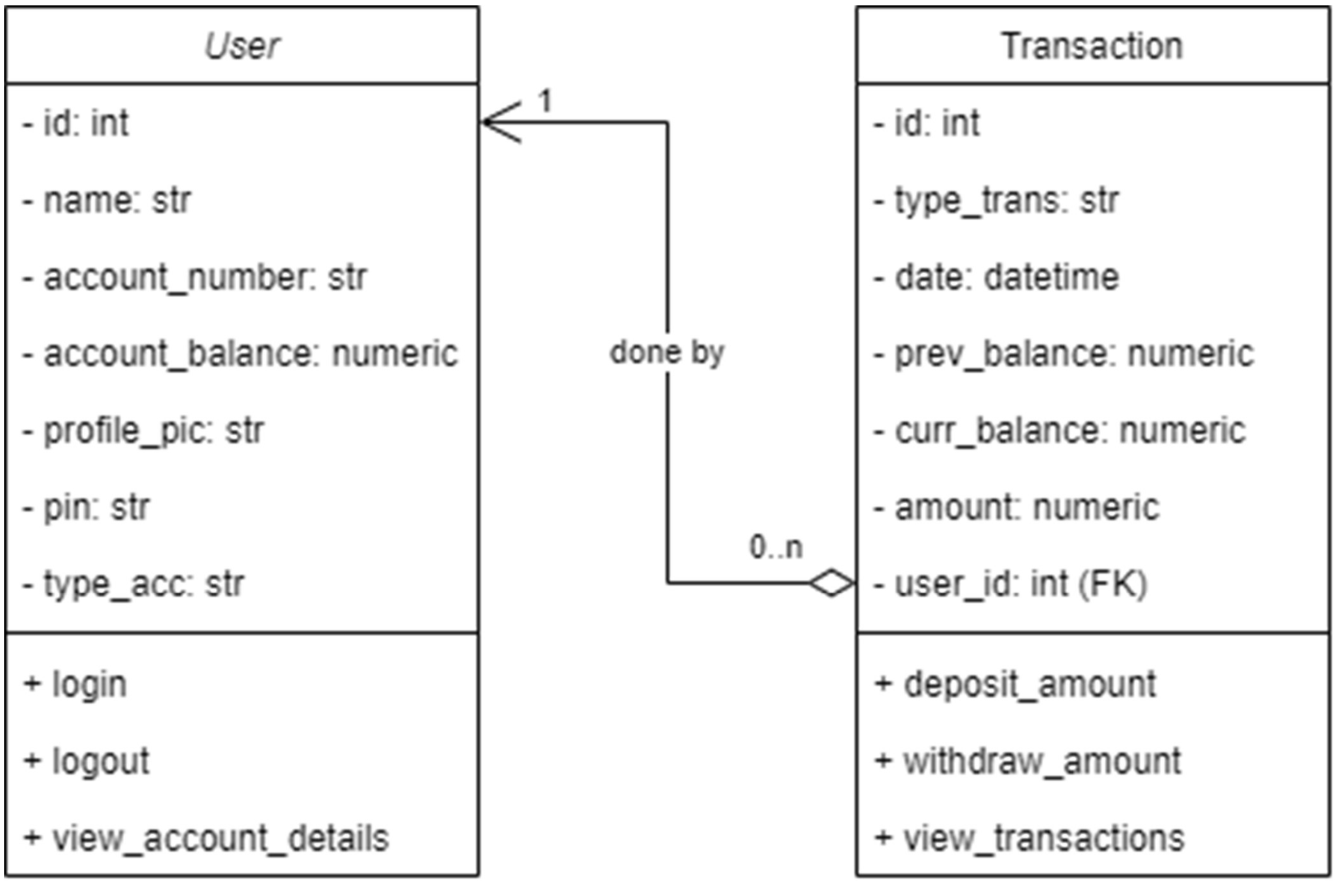
## 2-level DFD


User
Withdraw Funds
Deposit Funds
View Transactions
View Account Details
Validate Withdrawal
Validate Deposit
Retrieve Transactions
Retrieve Account Information
Deduct Withdrawal
Display Transactions
Display Account Details
Dispense Cash
Update Account

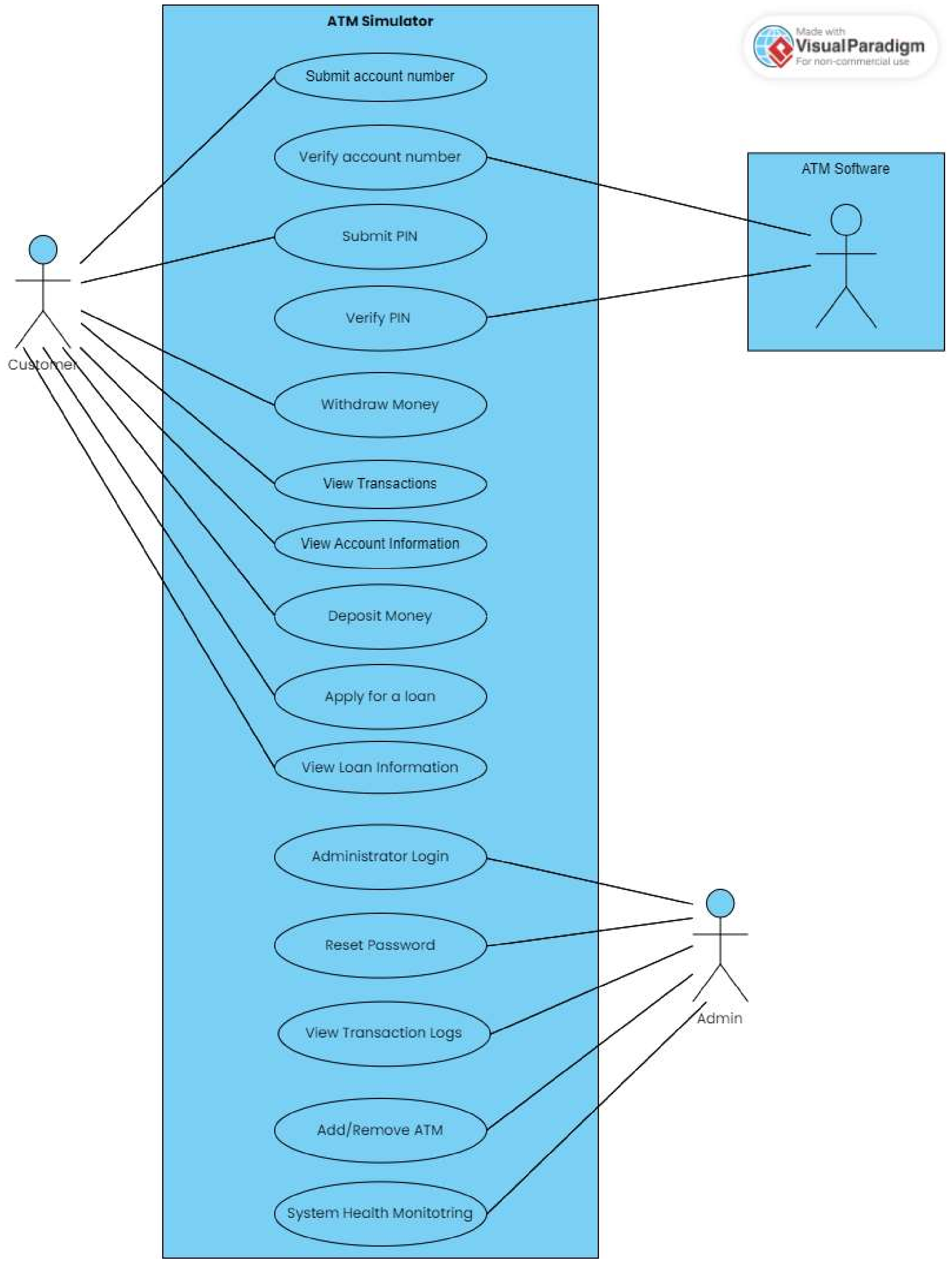
## *Architectural Design*



## *Class diagram*



## *Use Case Diagram*



# UE21CS341A - Software Engineering
# Test Plan Documentation

*ATM Simulator*

Team #: __T_____

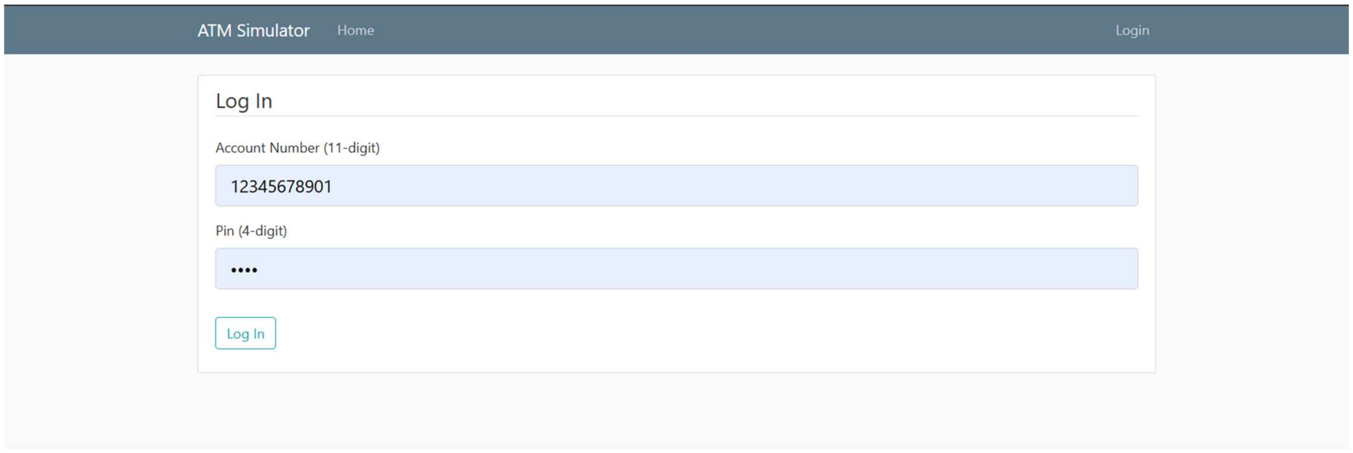
| | | | |
|---|---|---|---|
| PES1UG21CS652 | Swaraj MK | PES1UG21CS662 | Tanmay Praveen Udupa |
| PES1UG21CS618 | Srikrishna B | PES1UG21CS659 | T P Shriambikesh |

| Test Case ID | Name of Module | Test Case Description | Pre-conditions | Test steps | Test Data | Expected Result | Actual Results | Test Result |
|---|---|---|---|---|---|---|---|---|
| LOGIN_001 | Login | Valid Login with correct account number and pin | User should be on login screen | 1) Enter a valid account number 2)Enter a valid pin number | 12345678901 | User should be logged in and presented with the main menu | User is logged in and presented with the main menu | Pass |
| LOGIN_002 | Login | Invalid login with incorrect account number and pin | User should be on login screen | 1) Enter a invalid account number | 98765432101 | User should be presented with an error message indicating that the login details are incorrect | User is presented with an error message indicating that the login details are incorrect | Pass |
| LOGIN_003 | Login | Account number exceeding 11 digits | User should be on login screen | 1) Enter an account number that exceeds 11 digits | 1.23457E+11 | User should be presented with an error message | User is presented with an error message | Pass |
| LOGIN_004 | Login | Account number less than 11 digits | User should be on login screen | 1) Enter an account number that is less than 11 digits | 12345679 | User should be presented with an error message | User is presented with an error message | Pass |
| WITH_001 | Withdrawal | Successful withdrawal with sufficient funds | User should be logged in and have sufficient funds in their accounts | 1)Navigate to the withdraw tab 2) enter valid withdrawal amount | 100 | User should be presented with a successful message and should be redirected to main menu | User is presented with a successful message and should be redirected to main menu | Pass |
| WITH_002 | Withdrawal | Alphanumeric Entry in withdrawal | User should be logged in and should be on deposit page | 1) Enter a alphanumeric value in the text box | 100E | The application should show a error message stating that the entry is invalid | The application should show a error message stating that the entry is invalid | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| WITH_003 | Withdrawal | Unsuccessful withdrawal with insufficient funds | User should be logged in and have sufficient funds in their accounts | 1)Navigate to the withdraw tab 2) enter an invalid withdrawal amount | 10000 | User should be presented with an error message indicating that withdrawal amount is greater than balance | User is presented with an error message indicating that withdrawal amount is greater than balance | Pass |
| DEP_001 | Deposit | Successful deposit | User should be logged in | 1) Navigate to the deposit tab 2) Enter a valid deposit amount | 1000 | User should be able to view his updated balance in the view account details tab | User is able to view his updated balance in the view account details tab | Pass |
| DEP_002 | Deposit | Alphanumeric Entry in deposit form | User should be logged in and should be on deposit page | 1) Enter a alphanumeric value in the text box | 100E | The application should show a error message stating that the entry is invalid | The application should show a error message stating that the entry is invalid | Pass |
| BAL_001 | Balance check | Successful balance check | User should be logged in and should be on main menu | 1) Navigate to the View Account details tab 2) Check the available balance | N/A | User should be able to view his current balance | User is able to view his current balance | Pass |
| TRANSAC_001 | Transaction history | Check transaction history successful | User should be logged in and should be on main menu | 1) Navigate to the "View Transactions" tab and check the transactions history | N/A | User should be able to view all his transaction history | User is able to view all his transaction history | Pass |

## Test Screenshots

### TEST CASE: LOGIN_001



Log In

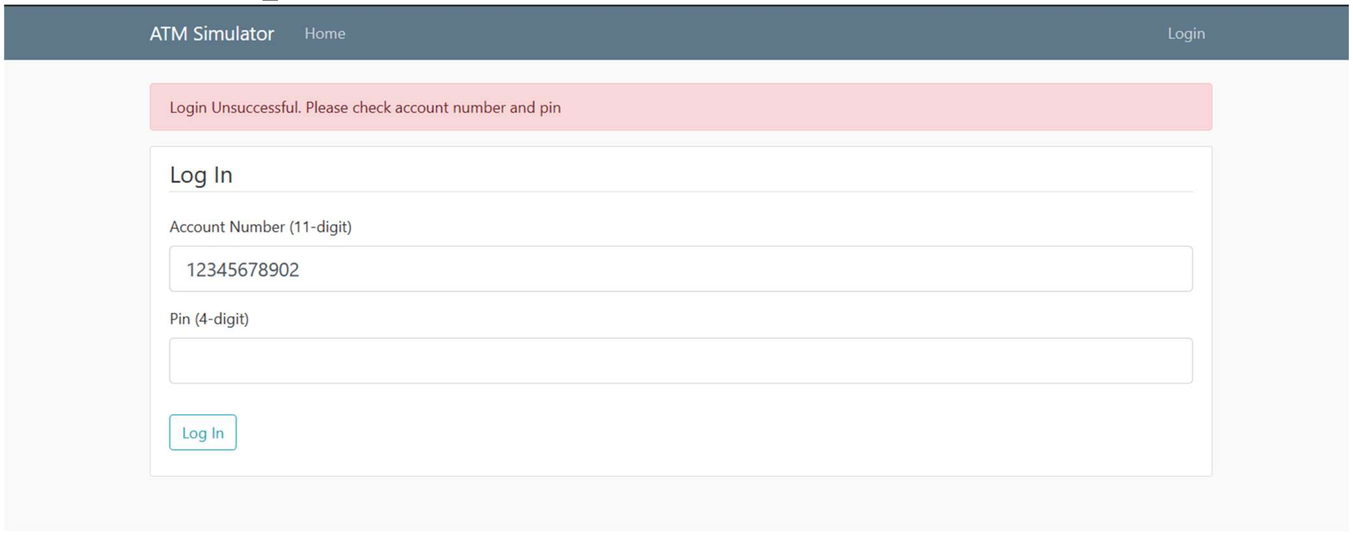
Account Number (11-digit)

12345678901

Pin (4-digit)

••••

### TEST CASE: LOGIN_002



Login Unsuccessful. Please check account number and pin

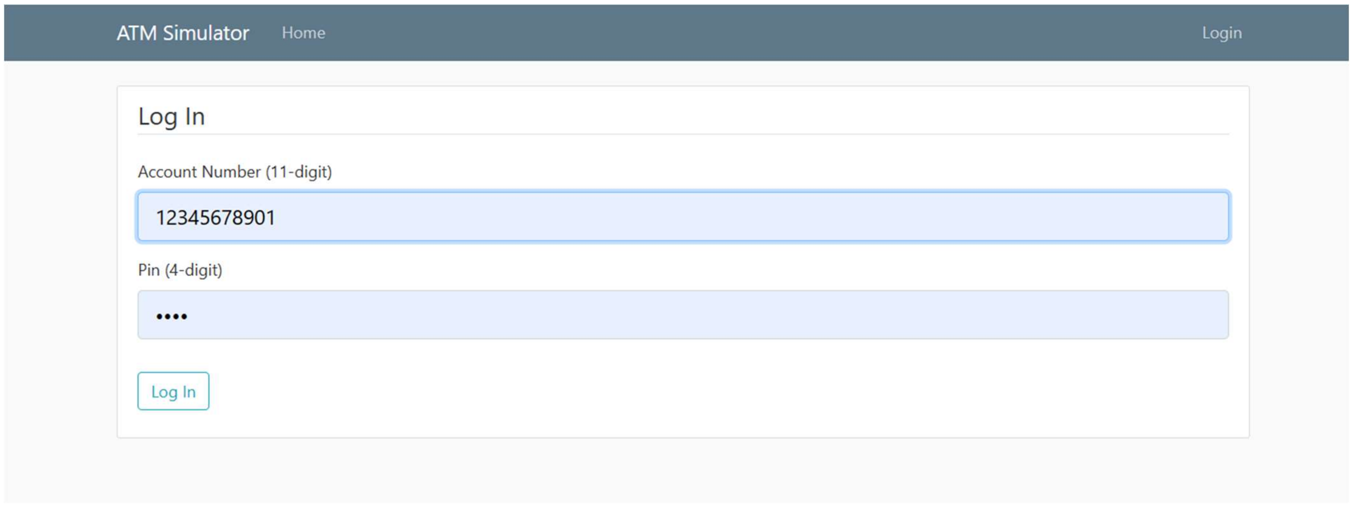
Log In

Account Number (11-digit)

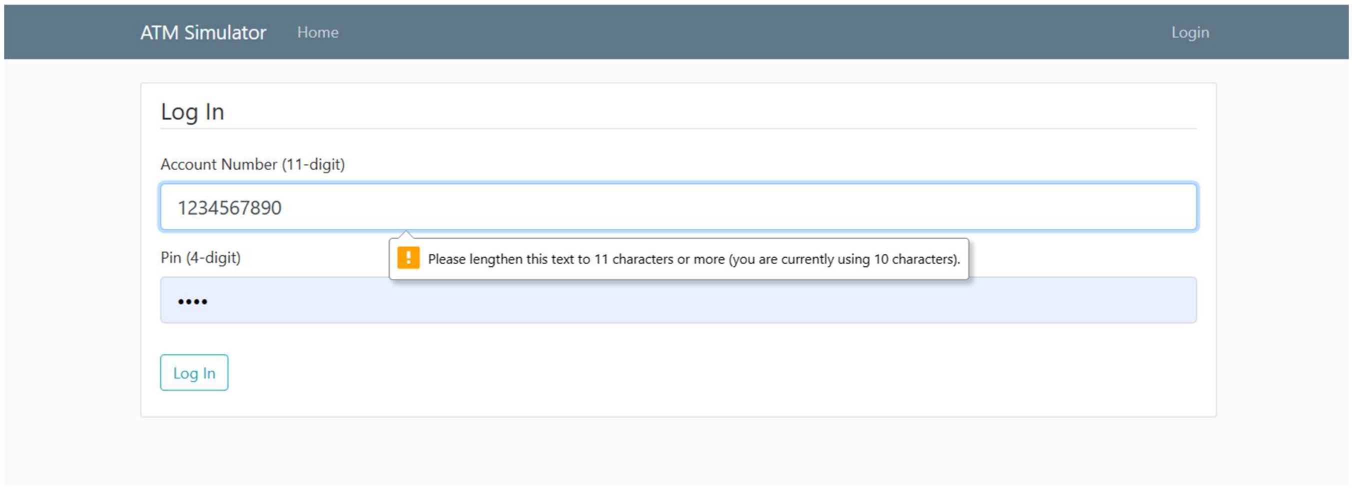
12345678902

Pin (4-digit)

### TEST CASE: LOGIN_003

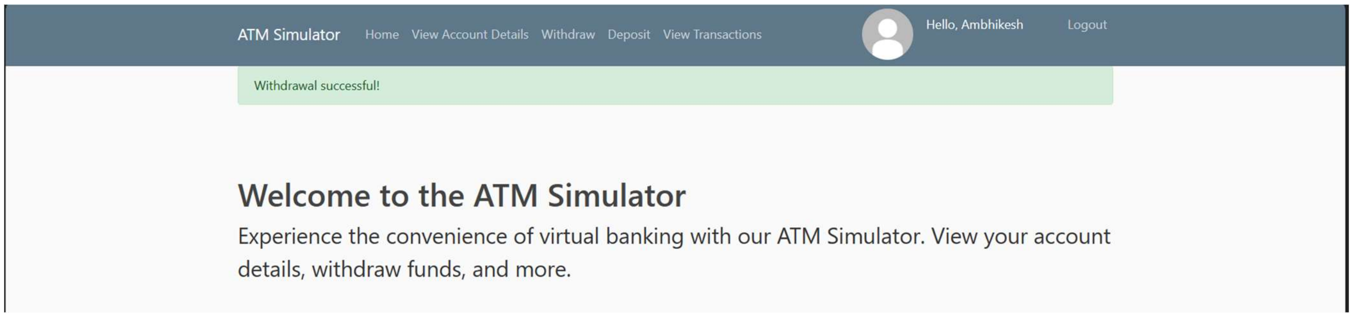


TEST CASE: LOGIN_004



Please lengthen this text to 11 characters or more (you are currently using 10 characters).

TEST CASE: WITH_001

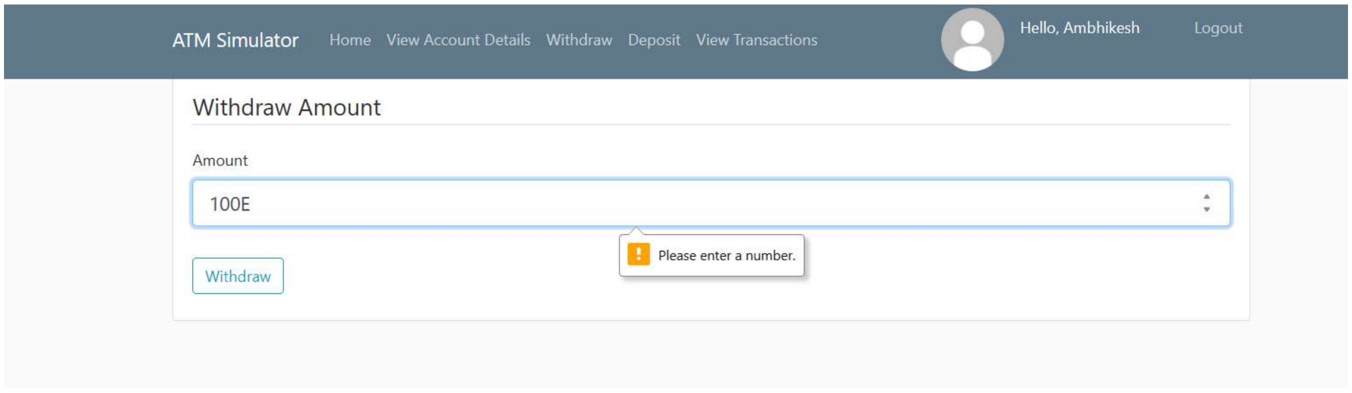


Withdrawal successful!

Welcome to the ATM Simulator

Experience the convenience of virtual banking with our ATM Simulator. View your account details, withdraw funds, and more.

TEST CASE: WITH_002



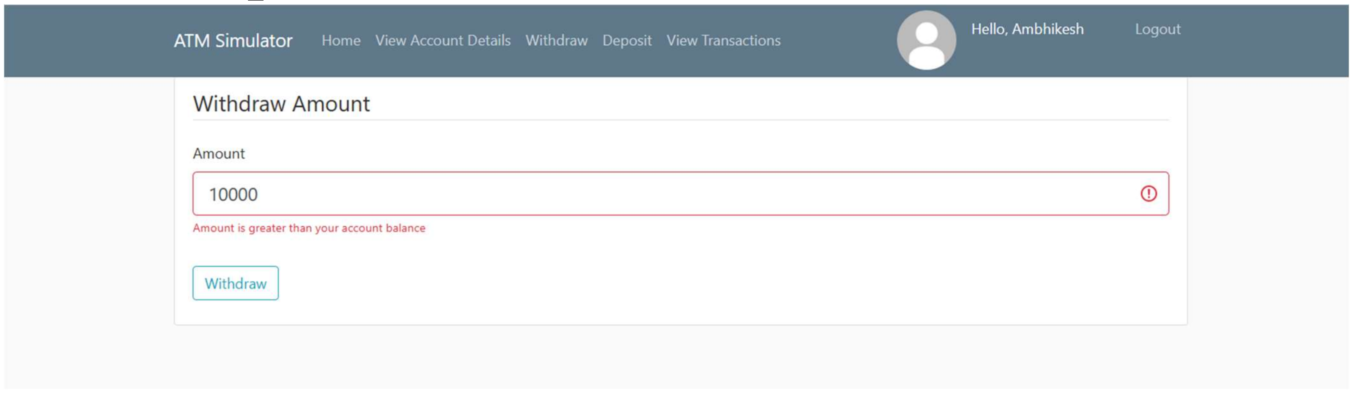
Withdraw Amount

Amount

100E

Please enter a number.

TEST CASE: WITH_003



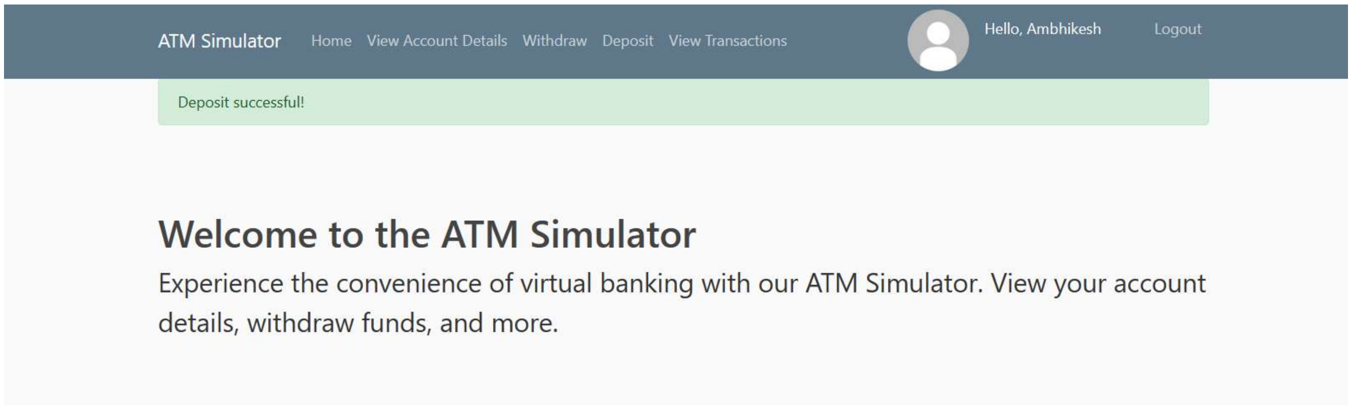
Withdraw Amount

Amount

10000

Amount is greater than your account balance

TEST CASE: DEP_001

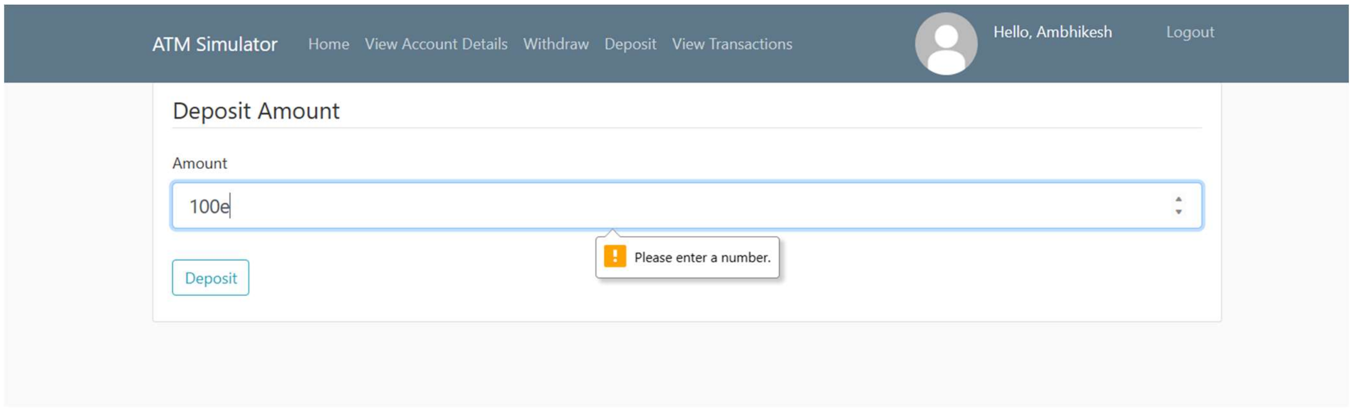


Deposit successful!

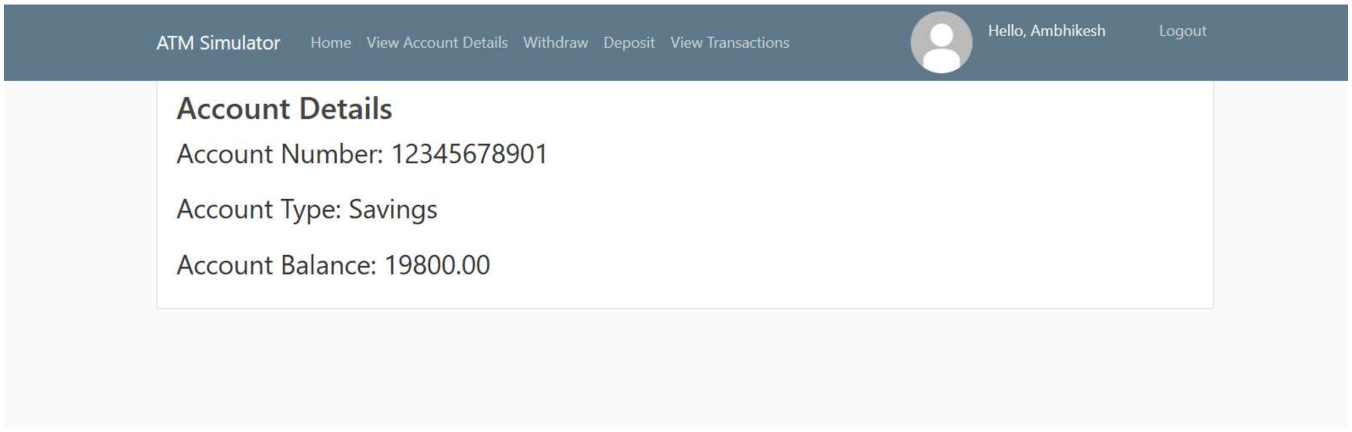
Welcome to the ATM Simulator

Experience the convenience of virtual banking with our ATM Simulator. View your account details, withdraw funds, and more.

TEST CASE: DEP_002



TEST CASE: BAL_001



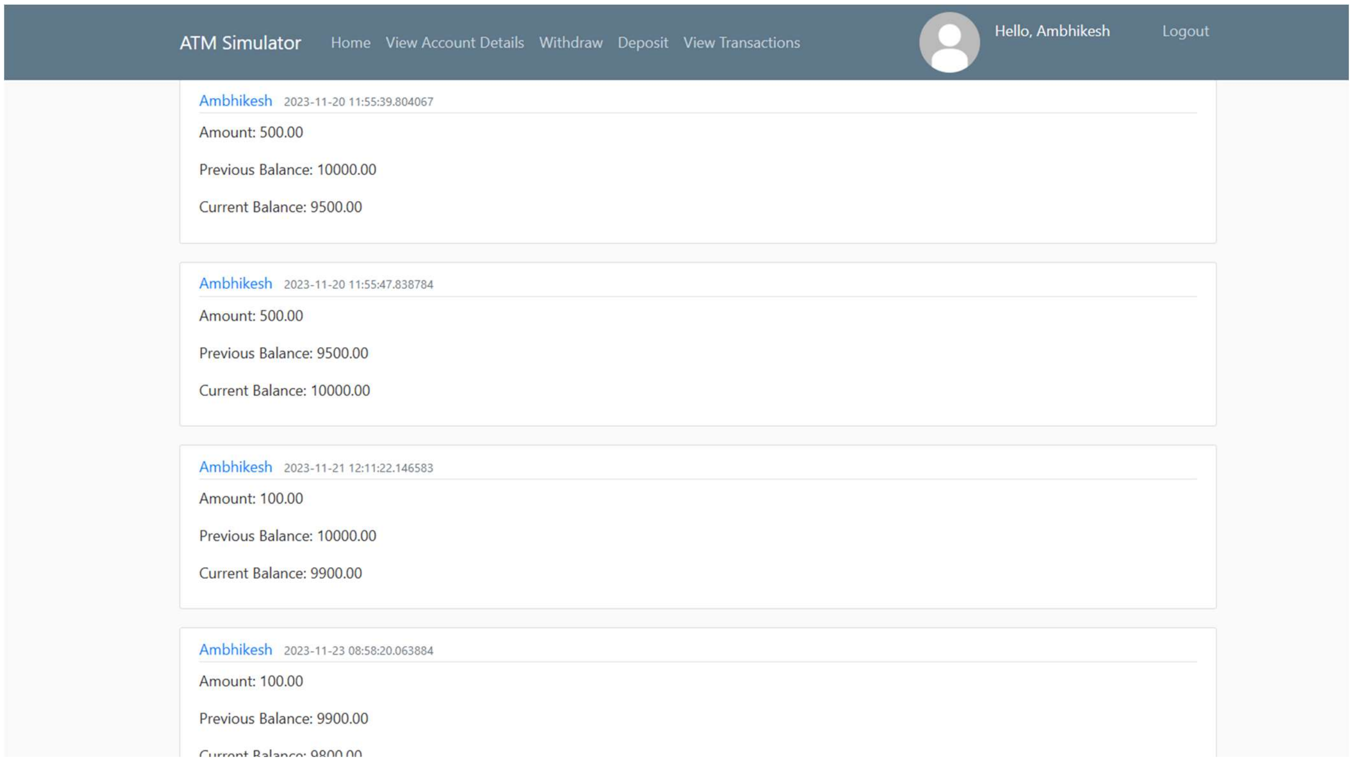
Account Details

Account Number: 12345678901

Account Type: Savings

Account Balance: 19800.00

TEST CASE: TRANSAC_001



Ambhikesh 2023-11-20 11:55:39.804067

Amount: 500.00

Previous Balance: 10000.00

Current Balance: 9500.00

Ambhikesh 2023-11-20 11:55:47.838784

Amount: 500.00

Previous Balance: 9500.00

Current Balance: 10000.00

Ambhikesh 2023-11-21 12:11:22.146583

Amount: 100.00

Previous Balance: 10000.00

Current Balance: 9900.00

Ambhikesh 2023-11-23 08:58:20.063884

Amount: 100.00

Previous Balance: 9900.00

Current Balance: 9800.00