**Software Requirements Specification**

**ATM System**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the needs for a simple Automated Teller Machine (ATM) system, which allows users to understand basic banking transactions. This project aims to enhance programming skills by implementing functionality of C or C++.

**1.2 Scope**

The ATM system operates in a console-based environment. Users will interact through a menu to perform basic transactions, such as:

1. Entering a valid PIN to access the bank account.
2. Viewing their account balance.
3. Depositing money into their account.
4. Withdrawing money from their account.
5. Exiting the system.

The system is designed to focus on the implementation of core ATM functionalities.

**1.3 Definitions and Abbreviations**

* **ATM**: Automated Teller Machine, a system that allows users to perform banking transactions.
* **PIN**: Personal Identification Number, used for user authentication in ATM systems.
* **Console-based**: An application that runs in a terminal or command-line interface.

**2. Overall Description**

**2.1 Product Perspective**

The ATM system is a simulated version of a real ATM machine. It does not involve real-time communication with banking networks or hardware like card readers and cash dispensers. Instead, the system:

* Uses variables to simulate user account details.
* Operates entirely in a command-line interface.
* Provides a menu-driven interface for interaction.

This is a basic run through of the application of programming constructs like loops, conditionals and functions.

**2.2 Product Functions**

The system offers the following core functionalities:

1. **Balance Inquiry**: Users can check their account balance, which is initialized at the start of the program.
2. **Cash Deposit**: Users can add money to their account balance by entering the deposit amount.
3. **Cash Withdrawal**: Users can withdraw money from their account balance, provided sufficient funds are available and the amount does not exceed the daily or monthly withdrawal limit.
4. **Exit**: Users can exit the system which terminates the program.

**2.3 User Characteristics**

The system is intended for users who:

* Have basic knowledge of using an ATM system.
* Can understand text-based instructions in English.
* Find it convenient to navigate a menu in a console-based interface.

**2.4 Assumptions and Dependencies**

* Users start with a predefined account balance.
* Users do not attempt to deposit or withdraw negative amounts.

**3. Specific Requirements**

**3.1 Functional Requirements**

1. **Menu Display**:
   * The system shall display a menu with the following options:

1. Enter a valid pin

2. Check Balance

3. Deposit Money

4. Withdraw Money

5. Exit

1. **Balance Inquiry**:
   * The system shall display the user’s current balance.
2. **Cash Deposit**:
   * The system shall prompt the user to enter the deposit amount.
   * The amount entered shall be added to the balance.
3. **Cash Withdrawal**:
   * The system shall prompt the user to enter the withdrawal amount.
   * If the entered amount exceeds the balance, the system shall display an error message.
   * If the balance is sufficient, the entered amount shall be deducted from the balance.
4. **Exit**:

The system shall end the program when the user selects the "Exit" option.

**3.2 Non-Functional Requirements**

1. **Performance**:
   * The system shall process the transactions and inputs of the user within 1 second.
2. **Usability**:
   * The system shall display clear instructions and error messages for invalid inputs.
3. **Reliability**:
   * The system shall handle invalid inputs by re-prompting the user without crashing.
4. **Portability**:
   * The program shall run on any C or C++ compiler including GCC and Turbo C++.

**4. Design Constraints**

* The system will use only basic in-memory data handling.
* If a database is integrated in the future, the required setup will include simple SQL commands.

**5. External Interface requirements**

**5.1 User Interface**

The user will interact with the system through a text-based menu. The system will provide clear instructions for every action and will guide the user through each step of the transaction.

**Error Handling:**

* If an invalid PIN or menu option is entered, the system will display an appropriate error message and prompt the user again.

**5.2 Hardware Interface**

The ATM system will be run on any standard desktop that has a C or C++ compiler. In the basic version, it does not interface with hardware devices (such as card reader, cash dispenser, etc.) and it is entirely software-based.

**5.3 Software Interface**

The system can interact with databases in the future. If an SQL database is used, it will interface with a simple database engine like SQLite or MySQL. This future interface will allow the system to:

* Authenticate users via stored PINs.
* Update the balance of user after withdrawals and deposits.

**6.0 Technologies used**

-Operating system - Windows 11/10

-Platform – VS code

-Coding Language - C

**7. Performance Requirements**

The ATM should have an availability of 99.5% with sufficient downtime for maintenance purpose.

An individual transaction should be processed within 5 seconds of input provided by the user.

**8. Security Enhancements**

Strong user authentication methods like Multi-Factor Authentication (MFA) or biometric features like fingerprint scanning can give a great boost to system security.

**9. Cross-Platform Compatibility**

The system could be made compatible with platforms like Linux or MacOS by using various tools for cross-platform development. This could ensure the smooth running of system on various operating systems.