**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 will allow users to get through basic banking transactions. This project is primarily for educational purposes and aims to enhance programming skills by implementing functionality in C or C++.

**1.2 Scope**

The ATM system is an application that operates in a console-based environment. Users will interact with the system 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 project does not include real-time interaction with a database or physical hardware, and all data is stored temporarily during program execution. The system is designed to be simple, focusing on the logical 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.

**1.4 Overview**

This document outlines the detailed requirements for the ATM system, covering functional and non-functional aspects. It also provides a high-level overview of the system’s structure and behaviour, ensuring clarity for implementation and testing.

**2. Overall Description**

**2.1 Product Perspective**

The ATM system is a standalone, 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 basic simulation is designed for learning and practice, focusing on 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.
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.
* Are comfortable navigating a menu in a console interface.

**2.4 Constraints**

* The system does not support persistent data storage; all data is lost when the program exits.
* The program assumes users will provide numeric inputs for menu options and transaction amounts.
* Only one user account is simulated during runtime.

**2.5 Assumptions and Dependencies**

* Users start with a predefined account balance (e.g., ₹10,000).
* 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

* + The menu shall reappear after completing a transaction unless the user selects the "Exit" option.

1. **Balance Inquiry**:
   * The system shall display the user’s current balance.
2. **Cash Deposit**:
   * The system shall prompt the user to enter the amount to deposit.
   * The entered amount shall be added to the balance.
3. **Cash Withdrawal**:
   * The system shall prompt the user to enter the amount to withdraw.
   * 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 terminate the program when the user selects the "Exit" option.

**3.2 Non-Functional Requirements**

1. **Performance**:
   * The system shall process user inputs and transactions 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 in the basic version.
* If a database is integrated in the future, the required setup will include simple SQL commands to manage user data.

**5.External Interface requirements**

**5.1 User Interface**

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

**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 or laptop with a C or C++ compiler. In the basic version, it does not interface with hardware devices (such as a physical 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 user balances after deposits and withdrawals.

**6.0 Technologies used**

-Operating system - Windows 11/10

-Platform – VS code

-Coding Language - C