ATM Simulation Program...!

By: Jitender Kumar

Date: 21/04/2025

Course: BCA

Collage: JUNCR

Introduction:

This project is an ATM simulation system designed using C programming. The system allows users to perform essential banking operations such as checking the balance, withdrawing money, depositing money, and viewing a mini statement. It is a basic representation of how an ATM system functions in real life, but it provides an interactive learning experience for users.

The program uses functions to modularize the operations, ensuring better code management and readability of code. It also supports a dual-language interface (English and Hinglish) for a better user experience.

Features & Functionality:

The ATM simulation program offers the following features:

1. Language Selection
2. User Registration and Login
3. Balance Checking
4. Withdraw Money
5. Deposit Money
6. Mini Statement
7. Error Handling
8. Session Management

Code Walkthrough:

The program is divided into several functions to manage different operations. Some of the key functions are:

1. `checkBalance()`: This function is used to display the current balance of the user. Depending on the chosen language, the message is displayed in either English or Hinglish.

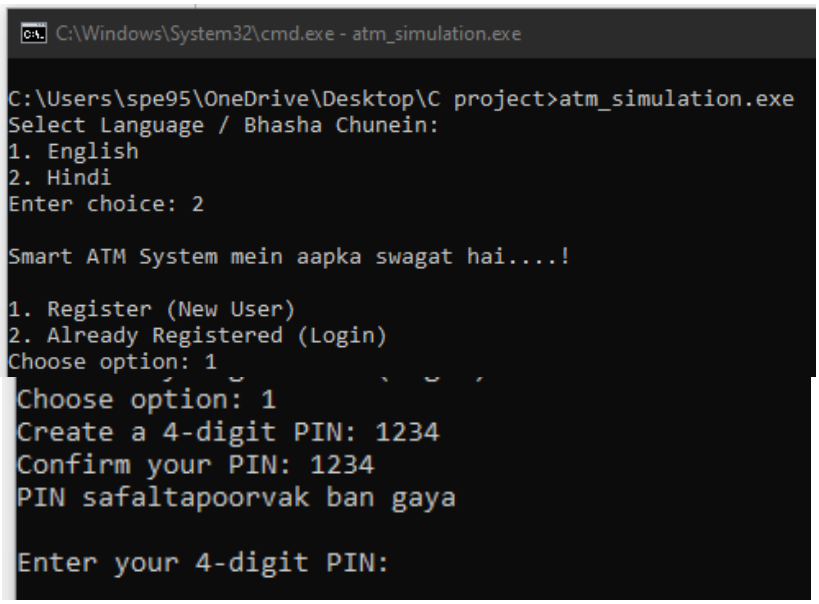
2. `withdrawMoney()`: This function allows the user to withdraw money. It checks if the withdrawal amount is valid (i.e., it doesn't exceed the balance or is a non-positive value). If the withdrawal is successful, the balance is updated.

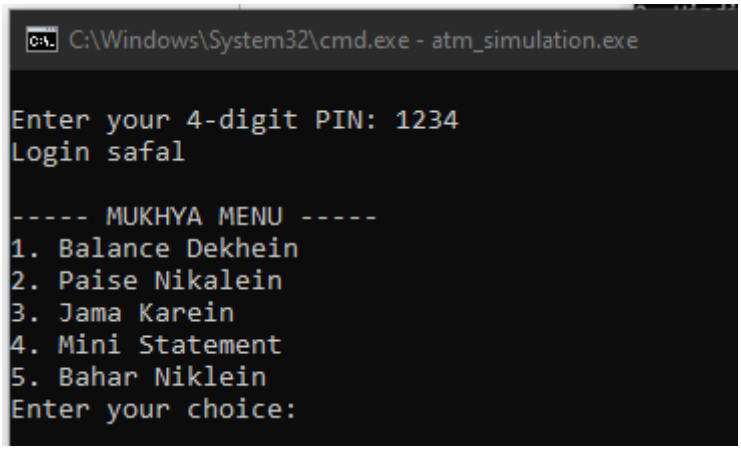
3. `depositMoney()`: Similar to withdrawal, this function allows the user to deposit a specified amount to their account.

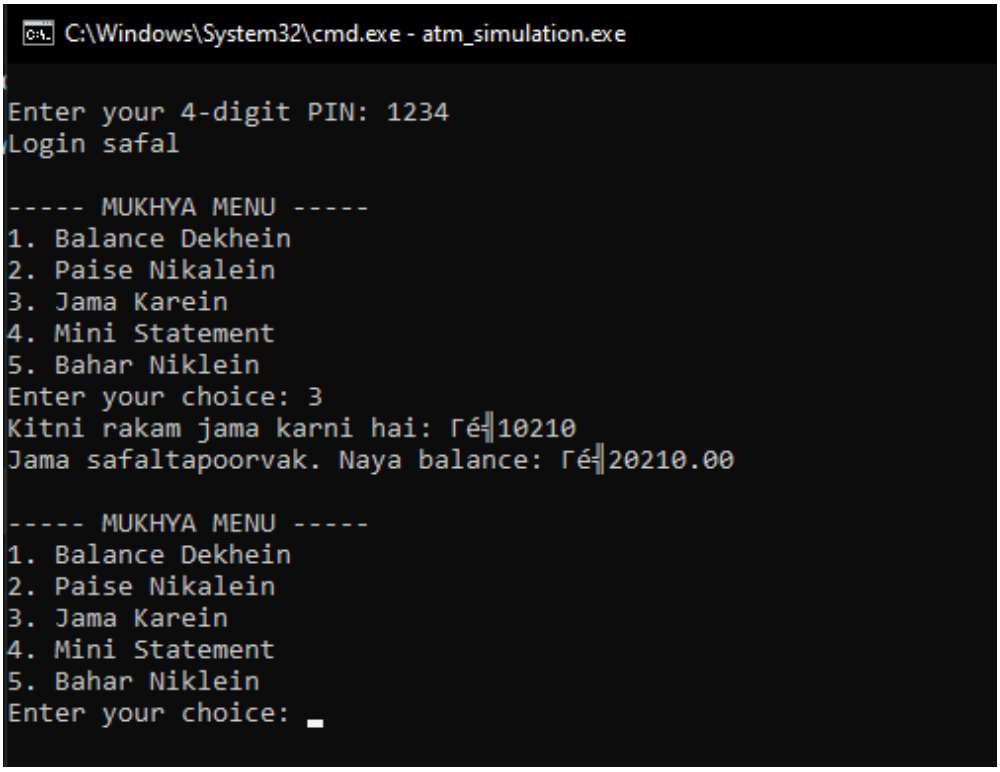
4. `addTransaction()`: This function stores the transaction details (e.g., deposit or withdrawal) in an array, which can be accessed in the mini statement.

5. `showMenu()`: This function presents the user with various options (e.g., check balance, withdraw money) and prompts for input.

Some Outputs:







Challenges & Solutions:

- Challenge 1: Handling user input validation for various options (such as checking for valid withdrawal amounts).

- Solution: I used a combination of `if` statements to validate inputs, ensuring that the user enters appropriate values (e.g., non-negative amounts).

- Challenge 2: Keeping track of transactions and showing the last 3 transactions in a mini statement.

- Solution: I used an array to store the transaction details and implemented a shifting mechanism to always keep the last 3 transactions visible.

Conclusion:

This ATM simulation system is a simple yet effective way to demonstrate the basic operations of an ATM. It helps in understanding how a real-world ATM system functions, including the handling of user registration, transactions, and errors. The dual-language interface makes it accessible to a wider audience. With the use of functions and modular coding, the system remains efficient and maintainable.