Computer Programming

CSL-113

CP Assignment 4

M Hasan Zaheer
01-134242-080

BSCS-1C

Submitted to: Arshad Farhad

Department of Computer Science
BAHRIA UNIVERSITY, ISLAMABAD

Project Report:

ATM System Simulation

Project Objective:

The primary objective of this project is to simulate an Automated Teller Machine (ATM) system using C++. The ATM system is designed to handle basic banking operations such as balance inquiry, cash withdrawal, deposit, fund transfer, and PIN management. The system also includes features like verifying the user’s PIN, ensuring correct withdrawal and deposit amounts, and updating the user's balance after every transaction.

Design Overview:

The system was designed with a focus on clarity, simplicity, and the simulation of real-world ATM functionalities. Below is an outline of the design:

Global Variables:

- `totalcash`: Represents the total cash available in the ATM.
- `balance`: The account balance of the user, stored locally.
- `totalcashwithdraw`: The maximum amount a user can withdraw based on their card type.
- `maxdeposit`: The maximum amount a user can deposit based on their card type.
- `currentpin`: Stores the user's current PIN.

Functions:

- **loadPin()**: Loads the user's PIN from a file (pin.txt) if available, otherwise defaults to 1234.
- **savePin()**: Saves the current PIN to a file (pin.txt).

- **loadBalance()**: Loads the user’s account balance from a file (balance.txt), or defaults to 80,000 if no balance file exists.
- **saveBalance()**: Saves the updated balance to a file (balance.txt).
- **validatepin()**: Ensures the correct PIN is entered by the user, allowing up to 3 attempts.
- **cardselection()**: Prompts the user to select their card type (Silver, Gold, or Platinum), each with different withdrawal and deposit limits.
- **displaymenu()**: Displays the main menu of the ATM system, offering options for various transactions.
- **balanceinquiry()**: Displays the user's current balance.
- **cashwithdraw()**: Allows the user to withdraw money from their account, ensuring it doesn’t exceed their balance or the ATM's withdrawal limits.
- **cashdeposit()**: Allows the user to deposit money into their account, ensuring the amount doesn’t exceed the deposit limits.
- **fundstransfer()**: Allows the user to transfer funds to another account, ensuring sufficient balance is available.
- **changepin()**: Allows the user to change their PIN after validating their current PIN.

Control Flow:

- The program starts by welcoming the user and asking for their PIN, validating it, and loading their balance and card type.
- After that, a menu is displayed for the user to select one of the available banking operations (balance inquiry, withdrawal, deposit, etc.).
- The system allows the user to perform multiple operations until they choose to exit.

File Handling:

- The user’s PIN and balance are stored and loaded from text files (pin.txt and balance.txt) for persistence across program runs.
- This mimics the functionality of real-world ATMs, where data is typically stored in a secure database or file system.

Implementation Details:

The implementation was carried out using C++ programming language with a focus on the following key aspects:

- **File Handling**: ifstream and ofstream were used to read from and write to files for storing and retrieving user information (PIN and balance).
- **Control Structures**: Conditional statements (if-else, switch) and loops (for, while, do-while) were employed to validate inputs and control the flow of the program.
- **Data Validation**: Ensured that the user's inputs (PIN, withdrawal amount, deposit amount, etc.) were within the acceptable range, and handled edge cases like negative numbers or amounts exceeding available balance or limits.
- **String Handling**: String comparison and user input validation (e.g., ensuring "yes" or "no" responses) were handled using standard input/output operations.

Challenges Faced:

Handling User Input Validation:

- One of the key challenges was ensuring that the user could only input valid data at every stage (e.g., correct PIN, valid amounts for withdrawals, deposits, etc.).
- Edge cases such as negative values, exceeding the ATM withdrawal limits, and mismatched PIN entries were handled using loops and conditional checks.

File Handling Errors:

- Initially, issues arose with reading and writing to files, particularly when the files were not available or corrupted.
- Proper error handling was implemented to ensure that the program doesn't crash or behave unpredictably if the files are not found.

ATM Simulation Logic:

- Simulating the real-world behavior of an ATM, especially ensuring that the withdrawal and deposit amounts were within the allowed limits, was tricky.
- Extra care was needed to prevent the user from withdrawing more money than the ATM had available or depositing more than the specified maximum limit.

Multi-step PIN Change Process:

- The PIN change logic required careful validation to ensure that the old PIN was correct, and that the new PIN was confirmed properly.
- Ensuring that all conditions were met (e.g., new PIN matching confirmation) was critical to avoid unexpected behavior.

Future Improvements and Enhancements:

Security Enhancements:

- Implementing encryption for the PIN and balance data would significantly improve the security of the system. Right now, data is stored in plain text, which could be vulnerable to attacks.

Additional Transaction Options:

- Adding additional functionalities such as viewing transaction history, setting up recurring payments, or allowing users to transfer funds between different accounts could make the system more comprehensive.

Error Recovery:

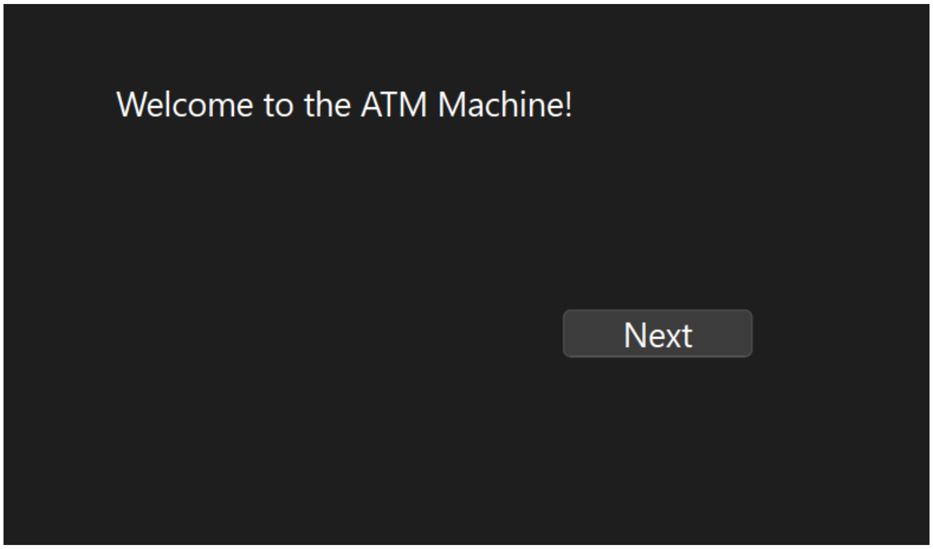
- Implementing a more robust error recovery mechanism (e.g., automatic retries for failed file handling or network issues) would enhance system resilience.

Network Connectivity:

- A future version of this system could connect to a remote server to fetch real-time information or enable account access from multiple ATMs.

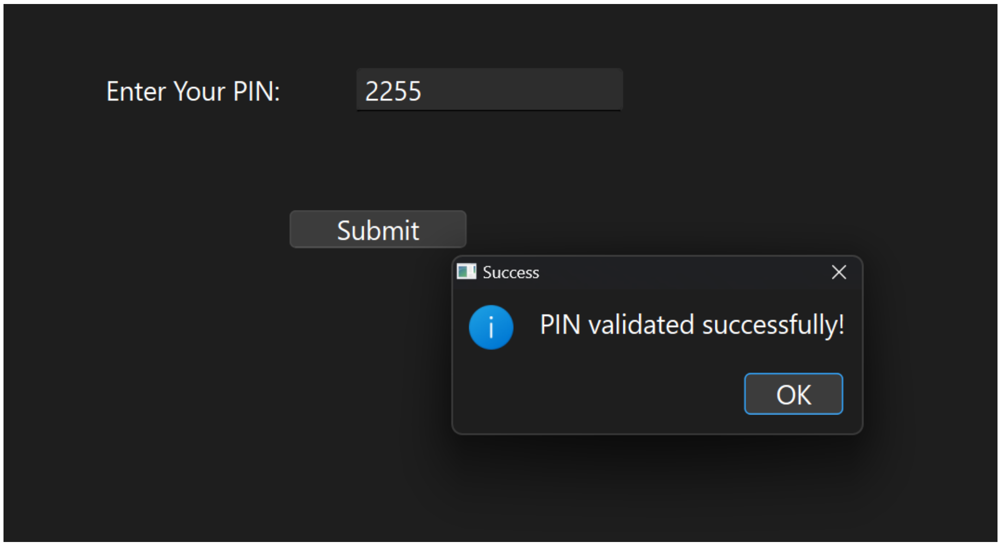
Screenshot Of Final Project:

Welcome Screen:



Welcome to the ATM Machine!

Pin Validation screen:

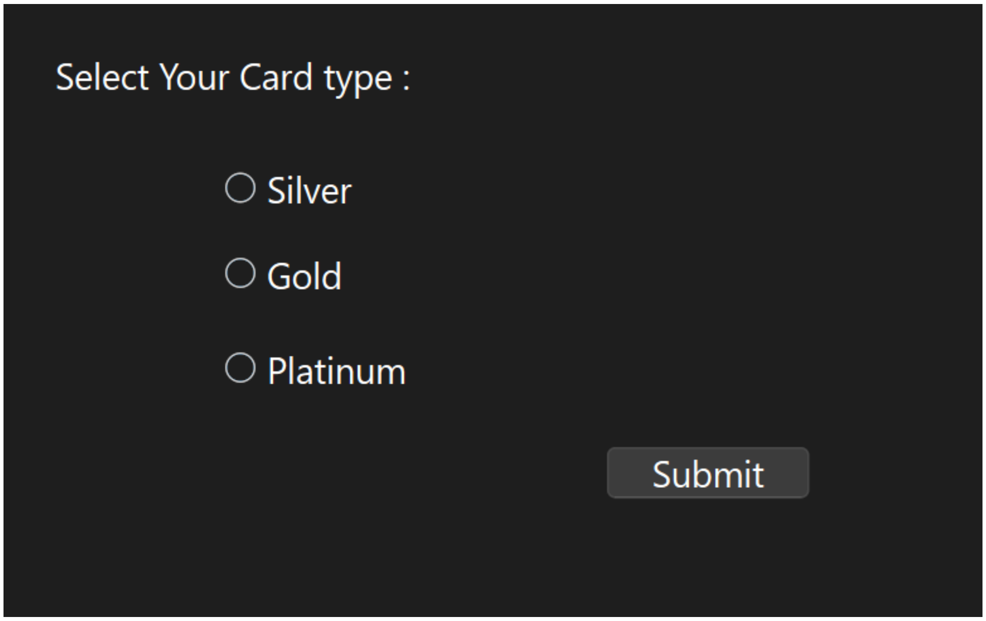


Enter Your PIN: 2255

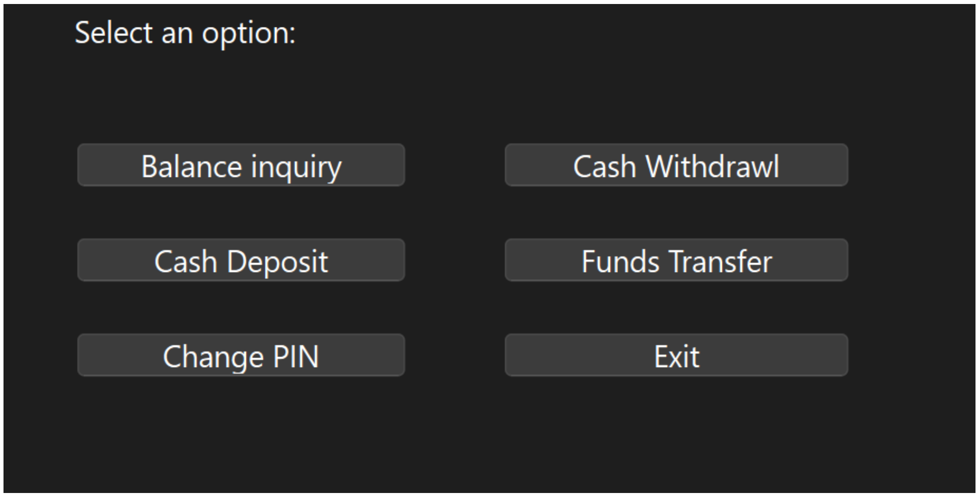
Success

PIN validated successfully!

Card selection Screen:



Main menu Screen:


Select an option:
Balance inquiry
Cash Withdrawl
Cash Deposit
Funds Transfer
Change PIN
Exit