COMPUTER PROGRAMING LAB

CP LAB PROJECT

ATM MACHINE

SUBMITTED BY

–ZAIRISH FATIMA (01-135241-052)
–DURDANA BILAL (01-135241-014)

CLASS – BS (IT) 1A

DATE OF SUBMISSION–6/10/2024

SUBMITTED TO–SIR ABRAR AHMED

Department of Computer Sciences

BAHRIA UNIVERSITY, ISLAMABAD

TABLE OF CONTENT

REPORT:

INTRODUCTION:

This project presents a code for a basic Automated Teller Machine (ATM) simulation. It emulates the functionalities of a real-world ATM, enabling users to perform typical banking transactions such as withdrawals, deposits, fund transfers, balance inquiries, and PIN changes. The system requires users to input a secret PIN number for authentication, akin to the security measures of a physical ATM. Users interact with the system through a menu displayed on the screen, selecting their desired actions.

The ATM simulation project is designed to focus on simulating the user interface and transaction management of a real ATM while ensuring data security through file handling. The integration of colored output enhances user experience by providing clear visual cues for different types of messages, including errors, confirmations, and informational prompts.

OBJECTIVES:

The objectives of this project include:

1. **Transaction Management:** Implementing fundamental banking operations such as withdrawals, deposits, fund transfers, and balance inquiries with a user-friendly interface.
2. **Data Security:** Ensuring secure handling of sensitive user information, such as PINs, and accurate transaction processing.
3. **User Interface:** Providing a simple yet interactive text-based interface for users to interact with the ATM system seamlessly.
4. **Data Persistence:** Employing file handling techniques to maintain user balance and PIN data between sessions, ensuring continuity.
5. **User Guidance:** Using color-coded output to guide users through the transaction process effectively, highlighting errors, confirmations, and important information.

SCOPE:

The scope of this ATM project encompasses:

1. Core Banking Transactions: Implementation of core functionalities including withdrawals, deposits, fund transfers, and balance inquiries.

2. PIN Management: Providing functionality for users to securely change their PIN after validating their current PIN.
3. Data Storage: Utilizing file handling to store and retrieve balance and PIN data securely.
4. User Authentication: Basic PIN-based authentication mechanism to access ATM functionalities.
5. Console-Based Interaction: Provision of a straightforward console-based interface for interaction, mimicking a typical ATM environment.

ASSUMPTIONS:

1. **Fixed Initial Data**: The system assumes predetermined initial balance and PIN values stored in the account_data.txt file or defaults to specific values if the file is absent.
2. **Positive Amounts:** All deposit, withdrawal, and transfer amounts entered by the user are assumed to be positive and valid.
3. **Correct File Operations**: It is assumed that file operations for saving and loading data will execute without failure, provided the file exists and is accessible.
4. **Console Input**: The system expects user interaction through the console interface, assuming correct inputs for each transaction.

LIMITATIONS:

1. **Initial Balance and Default PIN:** In the absence of a data file, the system sets an initial balance of 50,000 units and a default PIN of 1926, assuming a new user setup.
2. **Single User Account:** The system supports only one user account and does not accommodate multiple users or accounts.
3. **Correct Data Types:** Users are expected to enter the correct data types for each input, such as numeric values for the PIN and transaction amounts.
4. **Transaction Processing:** The system processes transactions sequentially, without handling concurrent transactions.

LITERATURE REVIEW:

In developing this ATM transaction system, I referred to several resources to gain a better understanding of the necessary concepts and programming techniques. The primary sources of help were YouTube channels specializing in programming tutorials:

- ***Code with Harry**:* This channel provided comprehensive tutorials on C++ programming and basic error handling.
- ***Apna College***: This channel offered tutorials on various programming concepts, including object-oriented programming, which is crucial for managing different aspects of the ATM system.

- **W3Schools***:* The W3Schools website served as a valuable resource for detailed explanations and examples of C++ syntax, functions, and best practices. It provided foundational knowledge on file handling and basic data operations, enabling a more structured and error-free codebase.

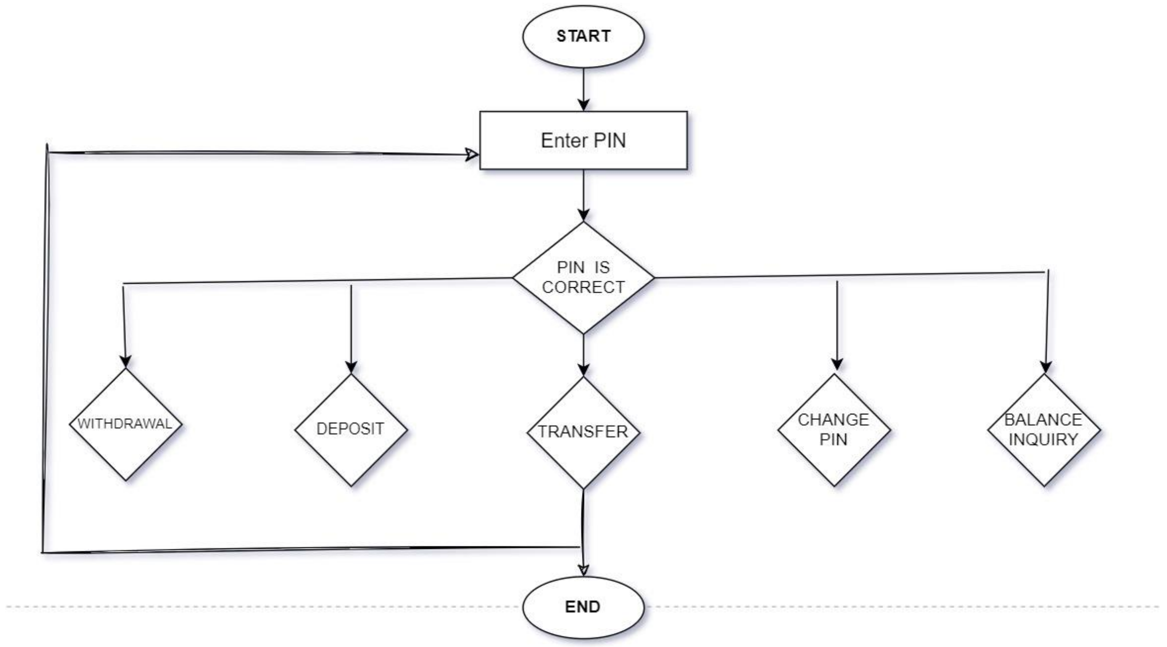
COMPARISON TABLE:

Aspect	What I Learned	What I Implemented
File handling	Reading from and writing to files in C++	Save/load balance and PIN data
Error handling	Basic techniques for input validation	Checked for valid inputs during transactions
Modular Programming	Importance of using functions	Divided code into functions
Windows.h library	Utilizing Windows.h for colored console output	Applied colored text using Windows.h library
User Interface Design	Design principles for user-friendly interfaces	Implemented clear and concise menu prompts
Data Persistence	Techniques for data storage and retrieval	Saved and loaded balance and PIN data from file
User Authentication	Methods for user verification and security	Required PIN entry for accessing account
Transaction Security	Ensuring secure transaction processing	Validated transaction inputs to prevent errors
Program Flow Control	Structuring program flow for efficient operation	Implemented a loop structure for user interaction
Error Reporting	Handling and reporting errors effectively	Displayed error messages for invalid transactions
Input Validation	Validating user inputs to ensure data integrity	Checked for non-negative transaction amounts
Data Encapsulation	Encapsulating data within structures for organization	Used a struct to store account details
Function Prototyping	Declaring function prototypes for clarity and order	Prototyped functions at the beginning of the code

METHODOLOGY:

The methodology employed in developing the ATM simulation project included:

Flowchart:



CODE OVERVIEW:

The provided code consists of a single C++ program that implements the ATM system simulation. The program uses a structure Account to store the user's account balance and PIN. The program has several functions to perform different transactions, including withdrawal, deposit, fundtransfer, changePIN, saveData, and loadData.

Functions:

1. **withdrawal(Account& account)**: This function allows the user to withdraw a specified amount from their account. It checks if the amount is valid (greater than 0 and less than or equal to the account balance) and updates the account balance accordingly.
2. **deposit(Account& account)**: This function allows the user to deposit a specified amount into their account. It checks if the amount is valid (greater than 0) and updates the account balance accordingly.
3. **fundtransfer(Account& account)**: This function allows the user to transfer a specified amount from their account to another account. It checks if the amount is valid (greater than 0 and less than or equal to the account balance) and updates the account balance accordingly.
4. **changePIN(Account& account)**: This function allows the user to change their PIN. It checks if the old PIN is correct and updates the PIN accordingly.
5. **saveData(const Account& account)**: This function saves the account balance and PIN to a file named "account_data.txt".

6. **loadData(Account& account)**: This function loads the account balance and PIN from the file "account_data.txt" and initializes the **Account** structure.

Main Function:

The main function is the entry point of the program. It initializes the Account structure and loads the account balance and PIN from the file using the loadData function. The program then enters a loop where it prompts the user to enter their PIN. If the PIN is correct, the program displays a menu of options for the user to select. Based on the user's selection, the program calls the corresponding function to perform the transaction. After each transaction, the program saves the updated account balance and PIN to the file using the saveData function.

Console Color:

The program uses the windows.h library to set the console text color using the SetConsoleTextAttribute function. This is used to display different colors for different messages, such as green for successful transactions and red for errors.

Testing:

The program was tested with various inputs to ensure that it works correctly. The following tests were performed:

- **Withdrawal:** Tested with valid and invalid amounts.
- **Deposit:** Tested with valid and invalid amounts.
- **Fund Transfer:** Tested with valid and invalid amounts.
- **PIN Change:** Tested with correct and incorrect old PINs.
- **Balance Inquiry:** Tested to ensure that the correct balance is displayed.

CONCLUSION:

In conclusion, the ATM simulation project in C++ successfully achieves its objectives of providing a basic yet functional emulation of a real-world ATM system. By implementing core banking transactions, secure data handling, a user-friendly interface, and color-coded output for improved user guidance, the system offers users a seamless banking experience. However, certain limitations and assumptions should be considered for further enhancements and refinement of the system.

References:

[1] Code with Harry. (n.d.). YouTube channel specializing in C++ programming tutorials. Retrieved from https://www.youtube.com/channel/UCeVMnSShP_Iviwkknt83cww

[2] Apna College. (n.d.). YouTube channel offering tutorials on various programming concepts, including object-oriented programming. Retrieved from https://www.youtube.com/channel/UCBwmMxybNva6P_5VmxjzwqA

[3] W3Schools. (n.d.). Online resource for detailed explanations and examples of C++ syntax, functions, and best practices. Retrieved from

https://www.w3schools.com/cpp/