ATM Management System

1. Introduction:

The proposed ATM Management System is a software solution designed to replicate essential functionalities of an Automated Teller Machine (ATM). This system aims to facilitate banking operations such as account creation, secure login authentication, balance inquiries, fund withdrawals, deposits, and PIN management. Developed in C++, the system heavily employs file handling techniques to manage and store user account information securely.

2. Objectives

Account Creation:

- ✓ Users initiate the account creation process by providing personal information: name, age, desired account number, chosen PIN, and an initial deposit.
- ✓ The system, through the `createAccount()` function, securely stores this information in a structured format within the "AccountsList.txt" file, ensuring organized data storage for future transactions.

Login System:

- ✓ Authentication is a pivotal aspect of the system. The `login()` function prompts users to enter their account number and PIN.
- ✓ It validates the entered credentials against the stored data in the "AccountsList.txt" file, enabling or denying access to the ATM services based on successful authentication.

Account Operations:

The system offers a range of operations mirroring those of a physical ATM:

- ❖ **View Balance:** Provides users with their current account balance without initiating any transactions.

- **Withdraw Cash:** Allows users to specify the amount they wish to withdraw, deducting the chosen sum from their account balance.

- **Deposit Funds:** Facilitates additional fund deposits into user accounts, incrementing the account balance.

- **Change PIN:** Enables users to update their PIN for enhanced security measures.

File Handling:

The system extensively uses file I/O operations to manage user account data efficiently. The **"AccountsList.txt"** file serves as a database, structuring user information for seamless retrieval and modification

3. Functional Segments:

a. Account Creation:

User input prompts via the command-line interface are handled by the createAccount() function. This segment securely collects and formats user data for storage**.**

Purpose:

- The "Account Creation" functionality enables individuals to establish new accounts within the ATM Management System. It facilitates the collection and storage of crucial user information to create a new account for future use.

Implementation:

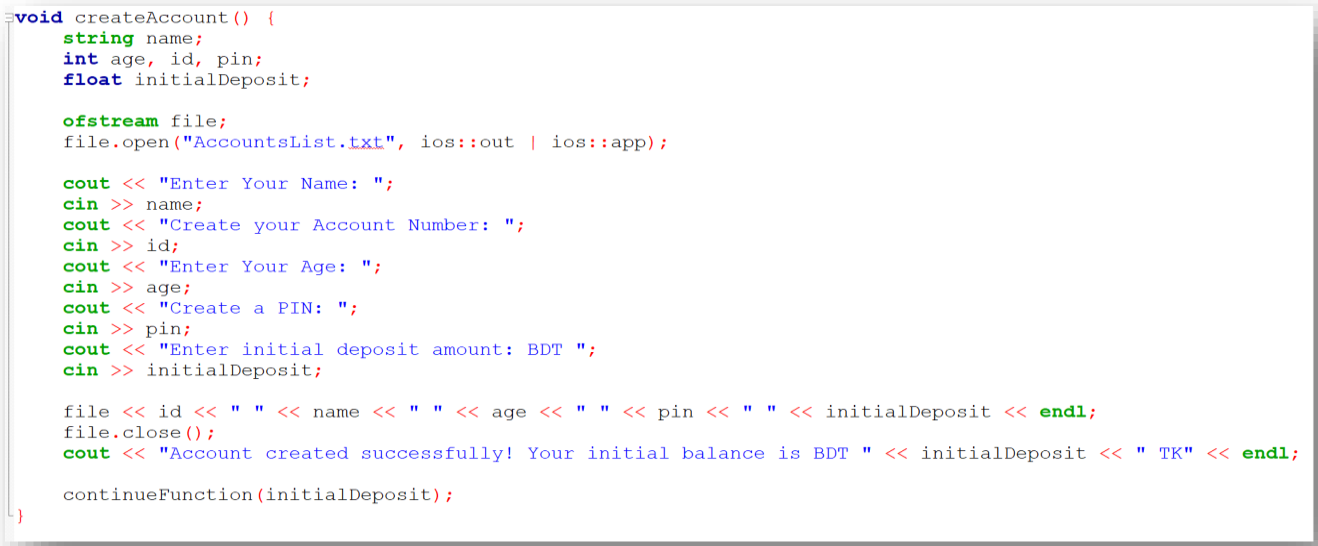
- The "Account Creation" option is typically available within the main menu of the system, inviting users to initiate the process of establishing a new account. Within the ATM system's interface, users are guided step-by-step to input required personal details for the account creation process.

Interaction Flow:

- **User Input:** Users enter their personal information as prompted by the system, usually including name, age, desired account number, chosen PIN, and an initial deposit amount.
- **Data Validation:** The system may perform basic validations on the entered information, such as format checks or ensuring uniqueness of the account number.

- **Storage:** Upon successful input and validation, the system securely stores the provided user information into the designated file (e.g., "AccountsList.txt").
- **Confirmation Message:** After successful account creation, the system provides a confirmation message indicating the successful establishment of the account.

Sample Code For Account Creation.



```
void createAccount() {
    string name;
    int age, id, pin;
    float initialDeposit;

    ofstream file;
    file.open("AccountsList.txt", ios::out | ios::app);

    cout << "Enter Your Name: ";
    cin >> name;
    cout << "Create your Account Number: ";
    cin >> id;
    cout << "Enter Your Age: ";
    cin >> age;
    cout << "Create a PIN: ";
    cin >> pin;
    cout << "Enter initial deposit amount: BDT ";
    cin >> initialDeposit;

    file << id << " " << name << " " << age << " " << pin << " " << initialDeposit << endl;
    file.close();
    cout << "Account created successfully! Your initial balance is BDT " << initialDeposit << " TK" << endl;

    continueFunction(initialDeposit);
}
```

b. Login System:

The `login()` function manages the authentication process by cross-verifying user-entered credentials with stored data, allowing or denying access to the ATM menu accordingly.

Purpose:

➢ The "Login" functionality verifies user identity by cross-checking entered credentials with stored data, allowing access to the ATM system's features.

Implementation:

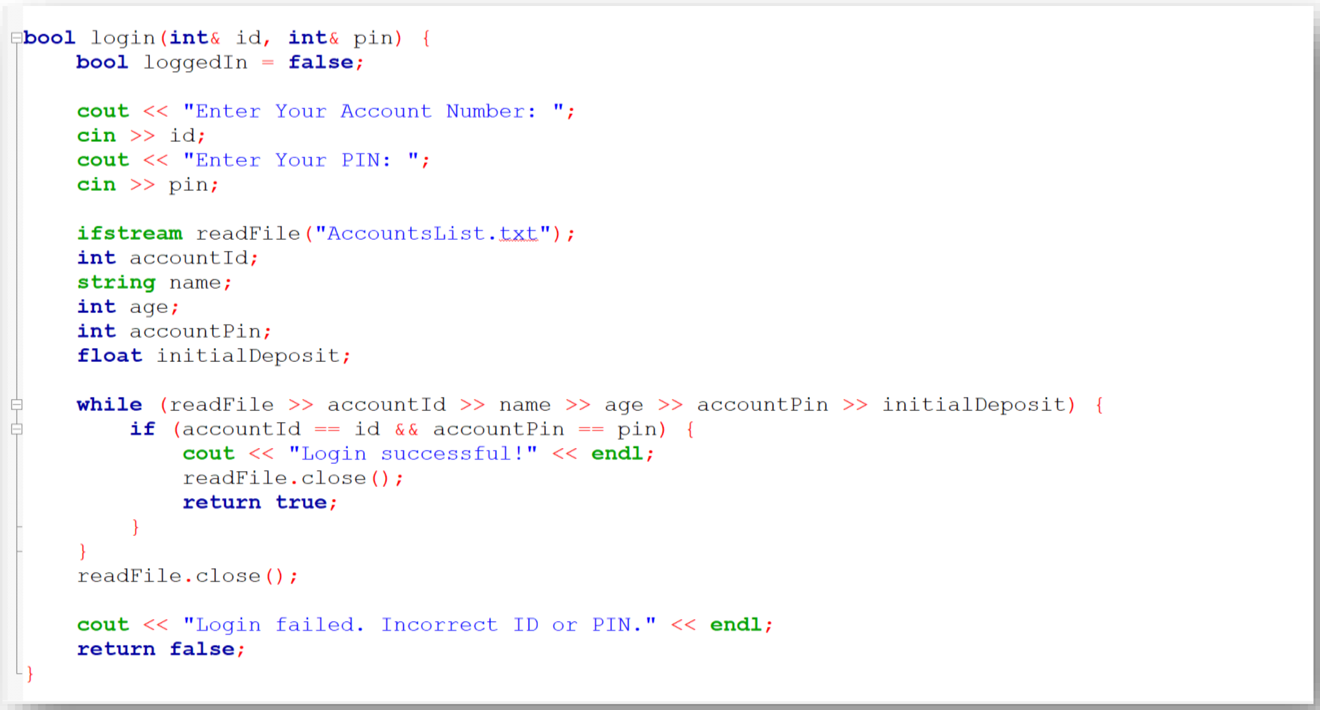
➢ The "Login" option is available within the main menu after the system launches or after account creation.Within the ATM system's interface, users are prompted to input their account number and associated PIN to authenticate themselves.

Interaction Flow:

- **User Input:** Users enter their account number and PIN as requested by the system.
- **Credential Validation:** The system compares the entered account number and PIN against the stored data in the "AccountsList.txt" file using the login() function.

- **Authentication:** If the entered credentials match the stored data, the system grants access to the ATM Menu, allowing the user to perform transactions.
- **Access Denied:** In cases of mismatched or incorrect credentials, the system denies access and prompts the user to re-enter the correct information.

Sample Code For Login



```
bool login(int& id, int& pin) {
    bool loggedIn = false;

    cout << "Enter Your Account Number: ";
    cin >> id;
    cout << "Enter Your PIN: ";
    cin >> pin;

    ifstream readFile("AccountsList.txt");
    int accountId;
    string name;
    int age;
    int accountPin;
    float initialDeposit;

    while (readFile >> accountId >> name >> age >> accountPin >> initialDeposit) {
        if (accountId == id && accountPin == pin) {
            cout << "Login successful!" << endl;
            readFile.close();
            return true;
        }
    }
    readFile.close();

    cout << "Login failed. Incorrect ID or PIN." << endl;
    return false;
}
```

c. ATM Menu:

Once logged in, users are presented with a comprehensive menu of options, each invoking a specific function:

✓ **View Balance:** Displays the current account balance.
Purpose:
➢ The "View Balance" functionality within the ATM Management System allows users to check the current balance of their account without initiating any transactions. It provides a quick and convenient way for account holders to inquire about their available funds.

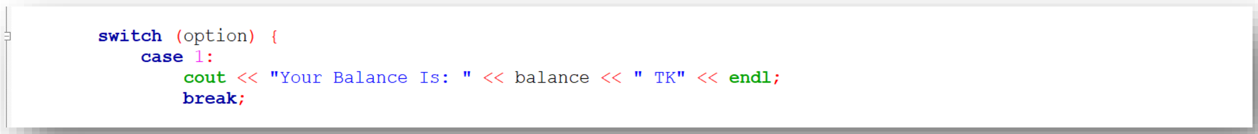
Implementation:

➢ This functionality is part of the ATM Menu presented to users after successful login. Within the atmMenu() function, the system offers the option to view the account balance, typically represented by an option number (e.g., option 1).Upon selecting the "View Balance" option, the system displays the current account balance associated with the logged-in account.

Interaction Flow:

- **User Selection:** The user inputs the option corresponding to the "View Balance" operation in the ATM Menu.
- **System Response:** The system accesses the account information stored in the "AccountsList.txt" file.
- **Display Balance:** The system retrieves the current balance linked to the user's account and displays it on the interface.
- **No Transaction:** This operation does not alter the account balance or initiate any deductions or deposits. It solely serves to provide information to the user.

Sample Code for View Balance



```
switch (option) {
    case 1:
        cout << "Your Balance Is: " << balance << " TK" << endl;
        break;
```

✓ **Withdraw Cash**: Facilitates cash withdrawal by deducting the specified amount from the account balance.

Purpose:

➢ The "Withdraw Cash" functionality allows account holders to withdraw funds from their account using the ATM system. It enables users to access their available balance and request a specific amount of cash.

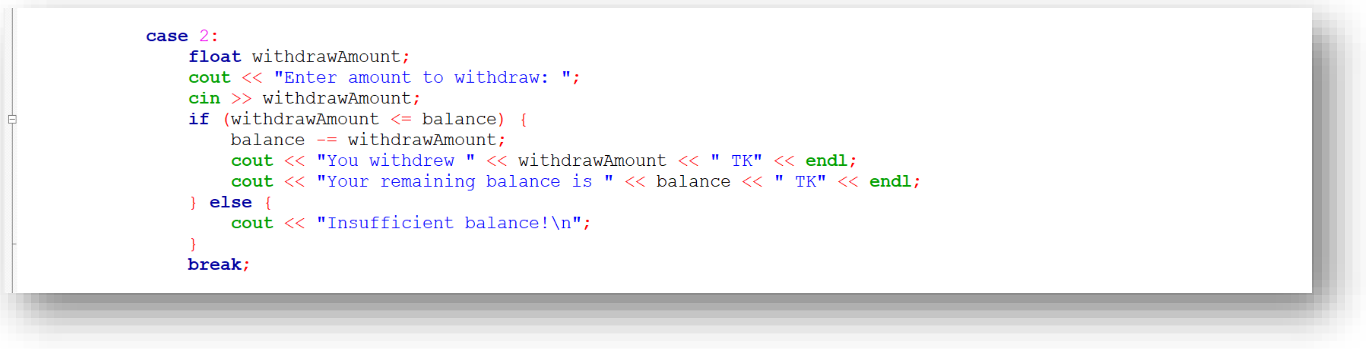
Implementation:

➢ This functionality is part of the ATM Menu presented to users after successful login.Within the atmMenu() function, users are prompted to select the option for cash withdrawal (usually represented by an option number, e.g., option 2).Upon selecting the "Withdraw Cash" option, the system initiates the process of deducting the specified amount from the user's account balance.

Interaction Flow:

- **User Selection:** The user inputs the option corresponding to the "Withdraw Cash" operation in the ATM Menu.
- **Withdrawal Amount:** The system prompts the user to enter the desired withdrawal amount.
- **Balance Check:** The system verifies if the user's account balance is sufficient to cover the requested withdrawal amount.
- **Deduct Balance:** If the balance is adequate, the system deducts the specified amount from the account balance.
- **Dispense Cash:** Simulated in the system, this step represents the dispensing of cash in a physical ATM.
- **Display Remaining Balance:** After the withdrawal, the system displays the remaining account balance to the user.

<u>**Sample Code for Withdraw**</u>



```
        case 2:
            float withdrawAmount;
            cout << "Enter amount to withdraw: ";
            cin >> withdrawAmount;
            if (withdrawAmount <= balance) {
                balance -= withdrawAmount;
                cout << "You withdrew " << withdrawAmount << " TK" << endl;
                cout << "Your remaining balance is " << balance << " TK" << endl;
            } else {
                cout << "Insufficient balance!\n";
            }
            break;
```

✓ **Deposit Funds:**Allows users to input additional funds, increasing their account balance.

Purpose:

➢ The "Deposit Funds" functionality allows account holders to add funds to their account using the ATM system. It provides users with a convenient method to increase their account balance by depositing cash.

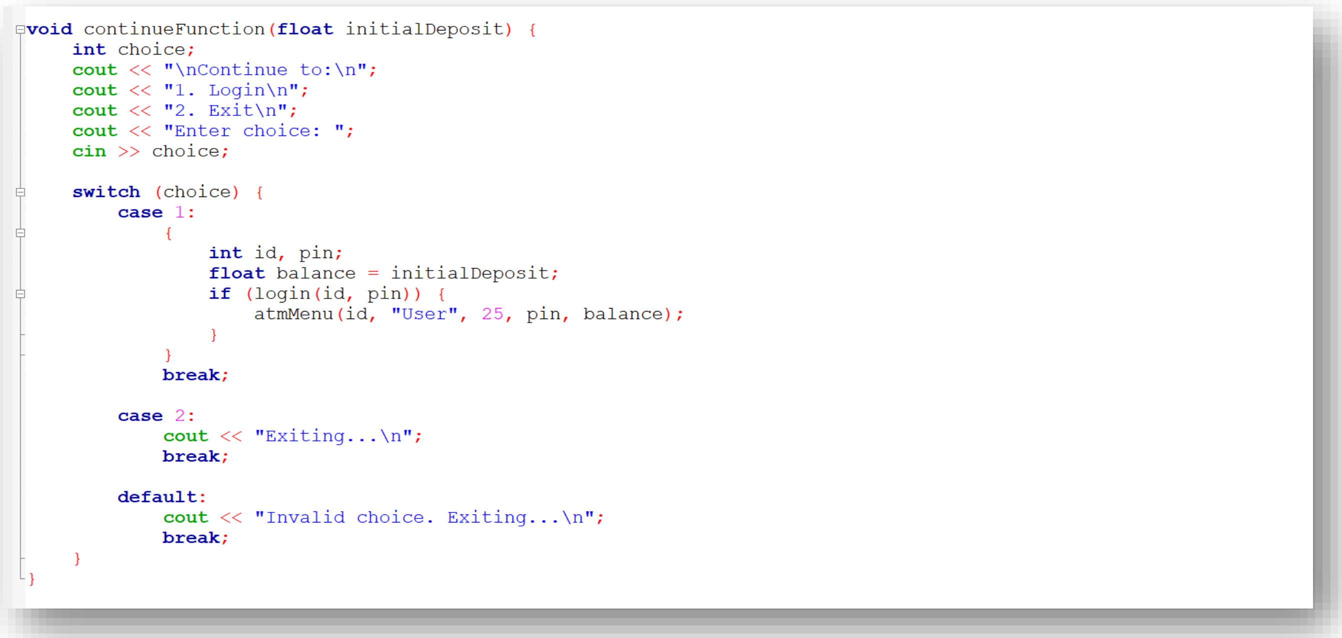
Implementation:

➢ Similar to other functionalities, the "Deposit Funds" option is part of the ATM Menu presented to users after successful login.Within the atmMenu() function, users are prompted to select the option for depositing funds (usually represented by an option number, e.g., option 3).Upon selecting the "Deposit Funds" option, the system initiates the process of adding the specified amount to the user's account balance.

Interaction Flow:

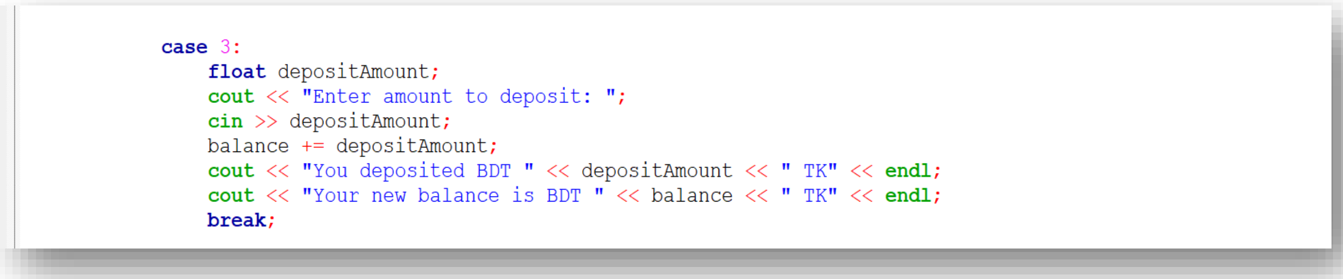
- **User Selection**: The user inputs the option corresponding to the "Deposit Funds" operation in the ATM Menu.
- **Deposit Amount:** The system prompts the user to input the desired deposit amount.
- **Update Balance**: The system adds the specified deposit amount to the user's account balance.
- **Confirmation Message:** Upon successful deposit, the system displays a confirmation message acknowledging the completion of the transaction and the updated balance.

Sample Code for Deposit



```
void continueFunction(float initialDeposit) {
    int choice;
    cout << "\nContinue to:\n";
    cout << "1. Login\n";
    cout << "2. Exit\n";
    cout << "Enter choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            {
                int id, pin;
                float balance = initialDeposit;
                if (login(id, pin)) {
                    atmMenu(id, "User", 25, pin, balance);
                }
            }
            break;

        case 2:
            cout << "Exiting...\n";
            break;

        default:
            cout << "Invalid choice. Exiting...\n";
            break;
    }
}
```

Sample Code for Deposit



```
        case 3:
            float depositAmount;
            cout << "Enter amount to deposit: ";
            cin >> depositAmount;
            balance += depositAmount;
            cout << "You deposited BDT " << depositAmount << " TK" << endl;
            cout << "Your new balance is BDT " << balance << " TK" << endl;
            break;
```

✓ **Change PIN:** Enables users to modify their PIN for security purposes.

Purpose:

➢ The "Change PIN" functionality allows account holders to update their Personal Identification Number (PIN) associated with their account. It provides users with an option to modify their PIN for enhanced security measures.

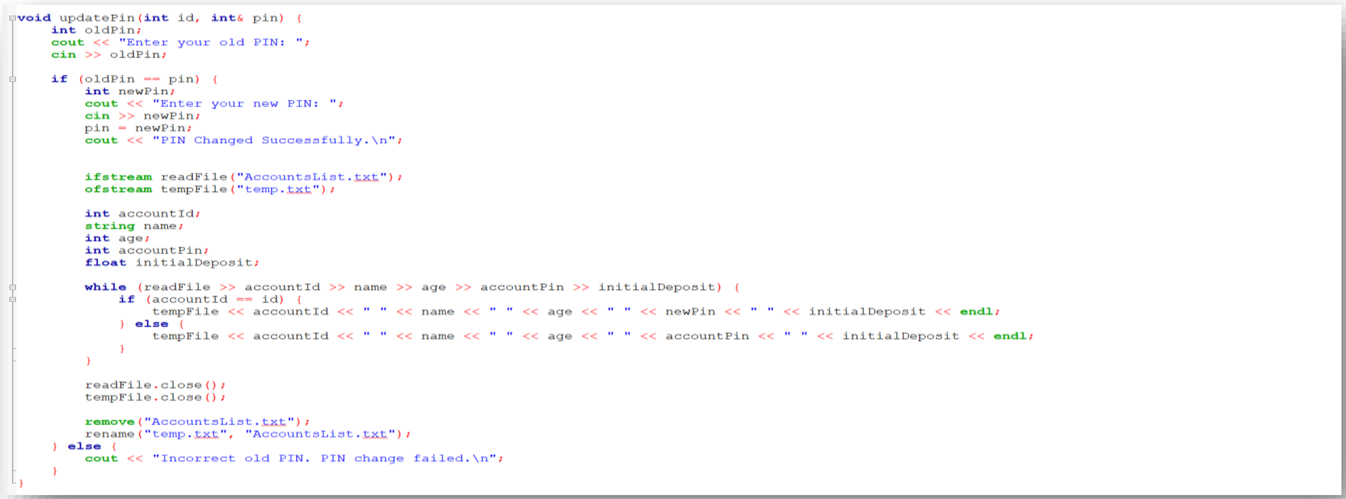
Implementation:

➢ The "Change PIN" option is available within the ATM Menu presented to users after successful login.Within the atmMenu() function, users are prompted to select the option for changing their PIN (usually represented by an option number, e.g., option 4).Upon selecting the "Change PIN" option, the system initiates the process to update the user's existing PIN with a new one.

Interaction Flow:

- **User Selection:** The user inputs the option corresponding to the "Change PIN" operation in the ATM Menu.
- **Old PIN Verification:** The system prompts the user to enter their old/current PIN for validation.
- **New PIN Entry:** Upon successful verification, the user inputs the desired new PIN.
- **Confirmation Message:** The system confirms the successful PIN change and updates the PIN associated with the user's account.

Sample Code for Change Pin



```cpp
void updatePin(int id, int& pin) {
    int oldPin;
    cout << "Enter your old PIN: ";
    cin >> oldPin;

    if (oldPin == pin) {
        int newPin;
        cout << "Enter your new PIN: ";
        cin >> newPin;
        pin = newPin;
        cout << "PIN Changed Successfully.\n";


        ifstream readFile("AccountsList.txt");
        ofstream tempFile("temp.txt");

        int accountId;
        string name;
        int age;
        int accountPin;
        float initialDeposit;

        while (readFile >> accountId >> name >> age >> accountPin >> initialDeposit) {
            if (accountId == id) {
                tempFile << accountId << " " << name << " " << age << " " << newPin << " " << initialDeposit << endl;
            } else {
                tempFile << accountId << " " << name << " " << age << " " << accountPin << " " << initialDeposit << endl;
            }
        }

        readFile.close();
        tempFile.close();

        remove("AccountsList.txt");
        rename("temp.txt", "AccountsList.txt");
    } else {
        cout << "Incorrect old PIN. PIN change failed.\n";
    }
}
```

- ✓ **Exit:** Allows users to exit the ATM service and return to the main menu.

  Purpose:

  - ➢ The "Exit" functionality serves to gracefully conclude the user's session within the ATM Management System. It enables users to log out or exit the system, returning to the main menu or ending their ATM session securely.

  Implementation:

  - ➢ The "Exit" option is available within the ATM Menu presented to users after successful login or within the main menu.Within the user interface (UI) of the ATM system, the "Exit" option is typically represented as a menu choice or button.Upon selecting the "Exit" option, the system executes the procedure to conclude the user's session and return control to the main menu or terminate the program, depending on the system's design.

  Interaction Flow:

  - **User Selection:** The user inputs the option corresponding to the "Exit" operation in the ATM Menu or main menu.
  - **System Response:** Upon selecting the "Exit" option, the system triggers the necessary steps to conclude the current session.
  - **Session Termination:** The system performs actions such as logging the user out, clearing temporary data, or terminating the program, depending on the intended functionality.

d. Update PIN:

The `updatePin()` function validates the old PIN before enabling users to set a new PIN. Upon successful PIN change, it updates the new PIN in the **"AccountsList.txt"** file, maintaining data consistency and security.

4. Improvements & Considerations:

Error Handling:

- ✓ Implementation of robust error handling mechanisms to manage unexpected inputs, file-related issues, and potential system errors.
- ✓ Exception handling techniques can prevent program crashes and ensure smooth user interactions.

Enhanced Security Measures:

- ✓ Consideration of encryption techniques to safeguard sensitive user data stored in the "AccountsList.txt" file.
- ✓ Encryption adds an additional layer of security against unauthorized access or data breaches.

User Experience Enhancement:

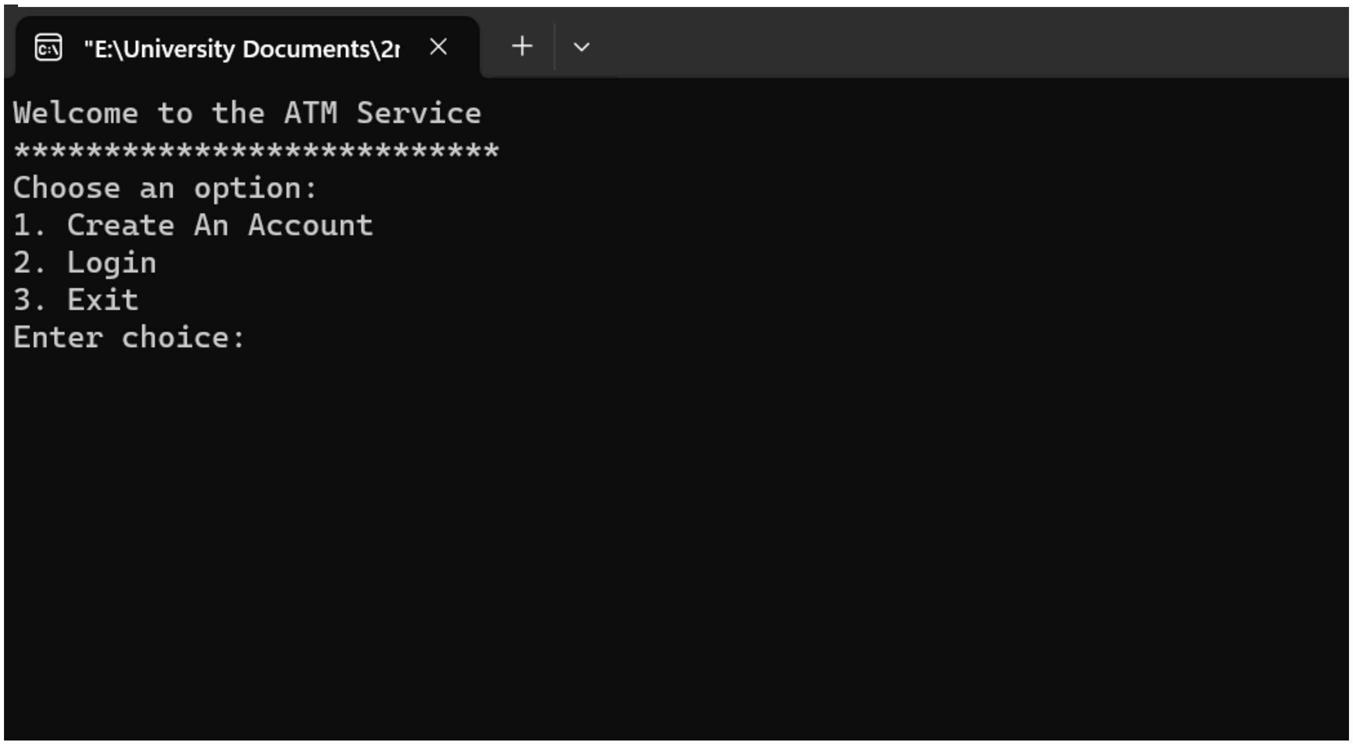
Expansion of functionalities to offer more user-friendly experiences, such as enabling balance inquiries without login requirements, providing transaction history, or supporting multiple accounts per user.

Code Refactoring & Optimization:

- ✓ Codebase optimization through refactoring and modularization to enhance maintainability, readability, and scalability for future enhancements or modifications.

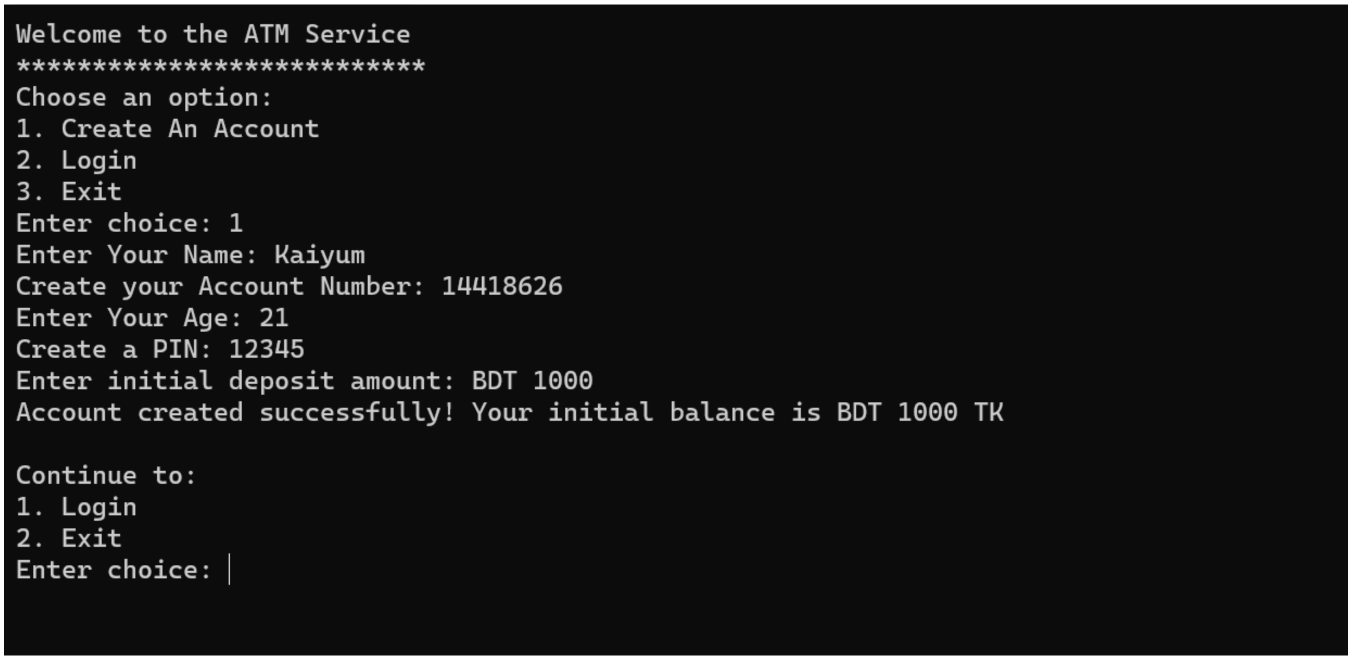
OutPut of the Project:

ATM Management Interface



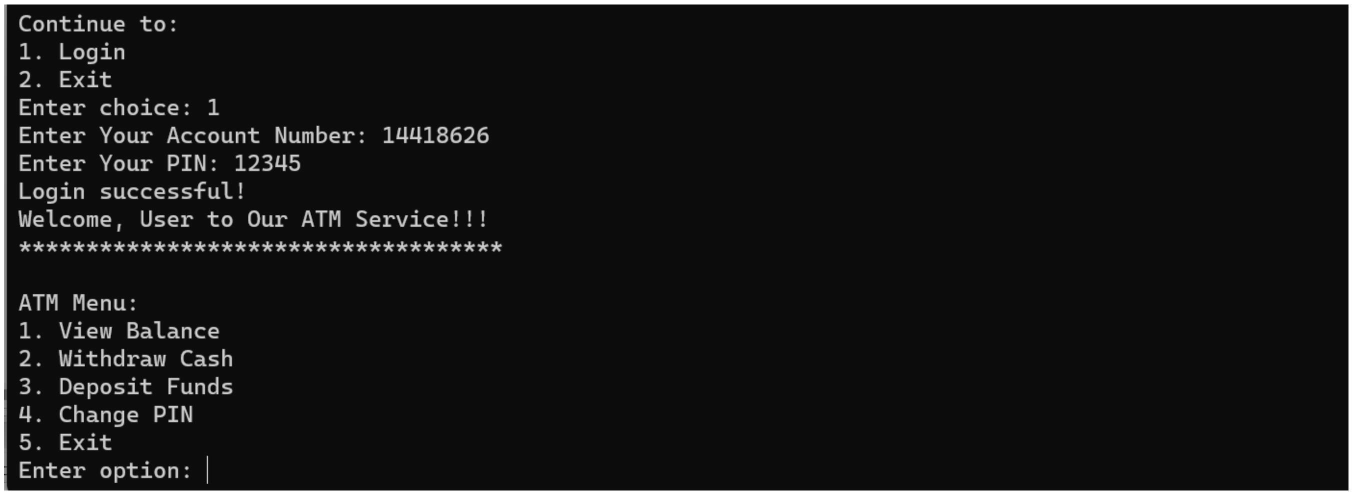
```
Welcome to the ATM Service
***************************
Choose an option:
1. Create An Account
2. Login
3. Exit
Enter choice:
```

Account Creation Interface



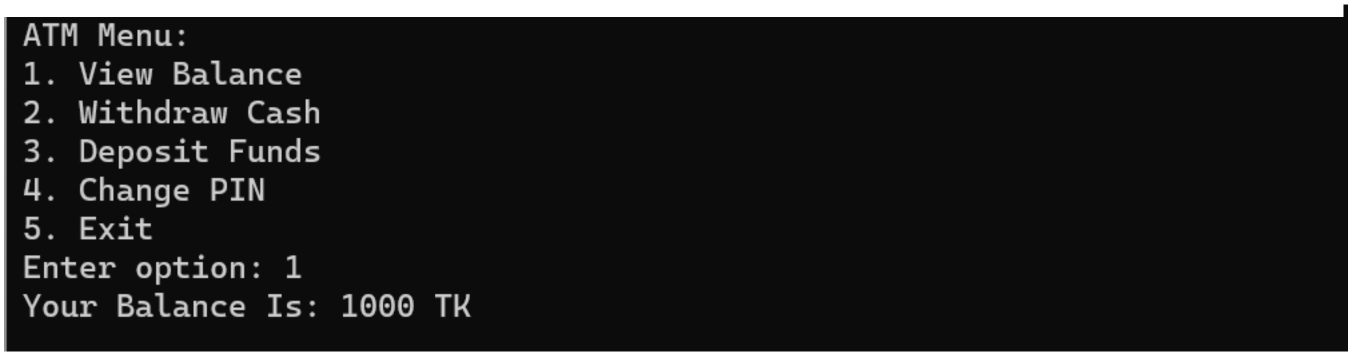
```
Welcome to the ATM Service
***************************
Choose an option:
1. Create An Account
2. Login
3. Exit
Enter choice: 1
Enter Your Name: Kaiyum
Create your Account Number: 14418626
Enter Your Age: 21
Create a PIN: 12345
Enter initial deposit amount: BDT 1000
Account created successfully! Your initial balance is BDT 1000 TK

Continue to:
1. Login
2. Exit
Enter choice:
```

Login Interface



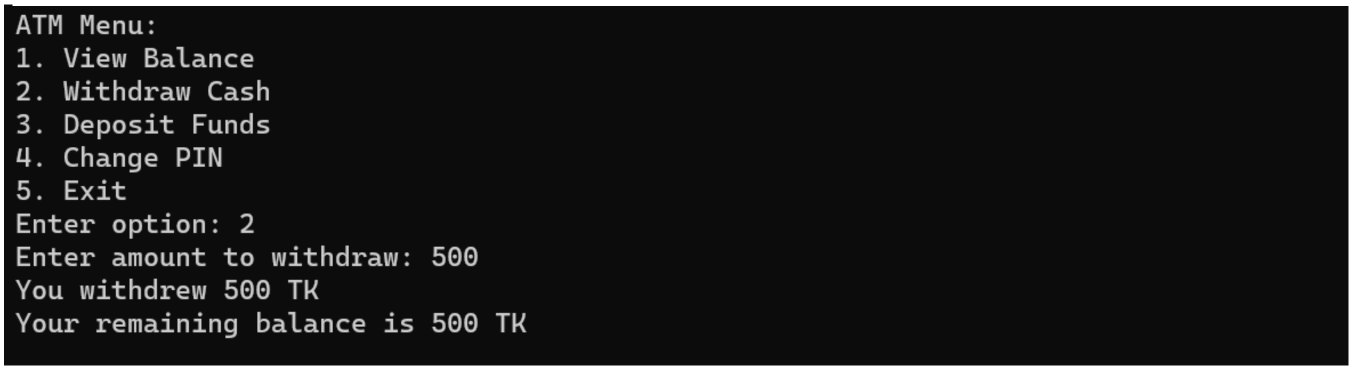
```
Continue to:
1. Login
2. Exit
Enter choice: 1
Enter Your Account Number: 14418626
Enter Your PIN: 12345
Login successful!
Welcome, User to Our ATM Service!!!
***********************************

ATM Menu:
1. View Balance
2. Withdraw Cash
3. Deposit Funds
4. Change PIN
5. Exit
Enter option:
```

1. View Balance Interface



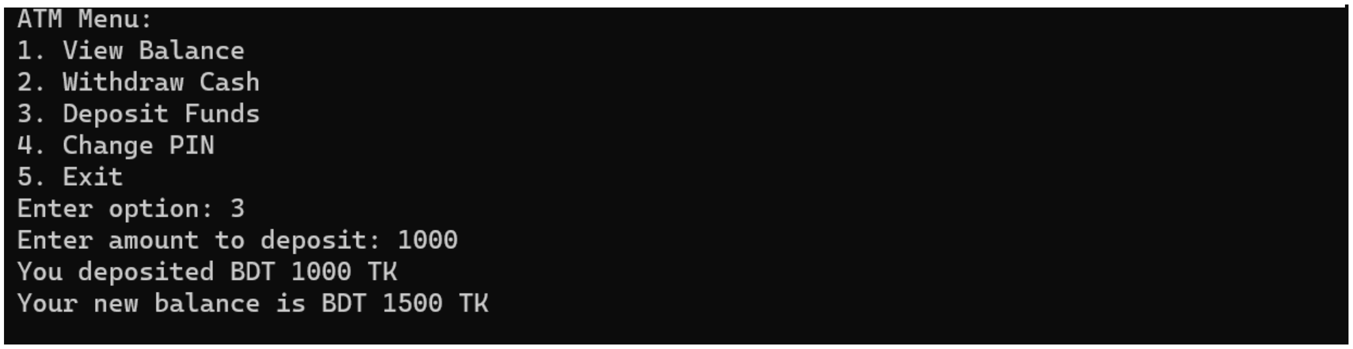
```
ATM Menu:
1. View Balance
2. Withdraw Cash
3. Deposit Funds
4. Change PIN
5. Exit
Enter option: 1
Your Balance Is: 1000 TK
```

2. Withdraw Cash Interface



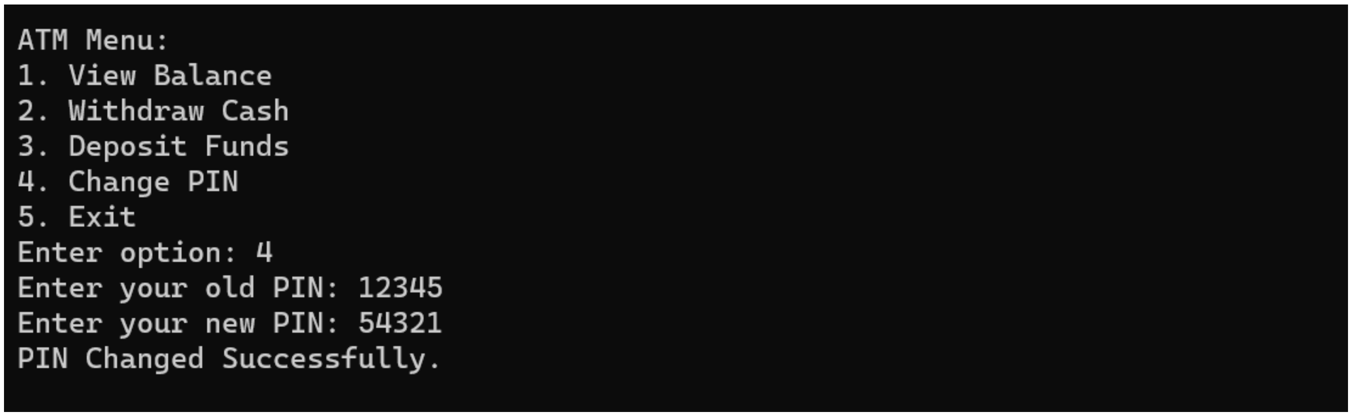
```
ATM Menu:
1. View Balance
2. Withdraw Cash
3. Deposit Funds
4. Change PIN
5. Exit
Enter option: 2
Enter amount to withdraw: 500
You withdrew 500 TK
Your remaining balance is 500 TK
```

3.Deposit Funds Interface



```
ATM Menu:
1. View Balance
2. Withdraw Cash
3. Deposit Funds
4. Change PIN
5. Exit
Enter option: 3
Enter amount to deposit: 1000
You deposited BDT 1000 TK
Your new balance is BDT 1500 TK
```

4. Change Pin Interface



```
ATM Menu:
1. View Balance
2. Withdraw Cash
3. Deposit Funds
4. Change PIN
5. Exit
Enter option: 4
Enter your old PIN: 12345
Enter your new PIN: 54321
PIN Changed Successfully.
```

5. Conclusion:

The proposed ATM Management System establishes a foundational solution for basic banking operations, delivering essential functionalities for account management and transactions. This project provides a platform for further expansion and refinement, aligning with industry standards for secure and efficient banking systems.This comprehensive proposal outlines the project's objectives, functional segments, potential improvements, and considerations for further development. Adjustments or additions can be made based on specific project requirements or desired enhancements.