PROJECT REPORT

INSTRUCTOR:MAA'M TAYYEBA
Submitted by:SAWAIRA MASOOD
BS-CS 1ST semester
Roll no:34
PROJECT TITLE:ATM INTERFACE

INTRODUCTION

- This project is a simulation of an **ATM (Automated Teller Machine) System**, designed to provide users with an interactive and secure banking experience. The program allows users to perform essential banking functions such as creating accounts, checking balances, depositing and withdrawing money, transferring funds, and managing their ATM PIN.
- Unlike a standard ATM interface, this program incorporates **enhanced security features**, including a **two-step verification system** for PIN recovery and automatic account locking after multiple incorrect PIN attempts. It also supports multi-user operations, allowing secure and seamless fund transfers between two accounts.
- With a focus on user-friendly design and robust input validation, the program ensures that all transactions are conducted securely and transparently. Additionally, features like transaction history and low-balance warnings improve the overall usability and practicality of the system.
- This project serves as a comprehensive demonstration of programming fundamentals, such as **structs**, **control flow**, **modularity**, and **file-less data management**, while incorporating practical elements of real-world banking systems. It is an excellent tool for learning about secure coding practices and enhancing problem-solving skills in programming.

➢ TECHNOLOGY STACK

- **Programming Language**: C++
- **Development Environment**: Any C++ IDE or compiler (e.g., Code::Blocks, Visual Studio, GCC).

➢ WORKFLOW

1. **Initialization**:
    - Users are prompted to create accounts before performing any other operations.
2. **Authentication**:
    - PIN-based authentication ensures that only authorized users can access account features.
3. **Transaction Execution**:
    - Based on user input, the appropriate function is executed, with real-time validation and updates to account details.

FEATURES

1. **Create Account**:
    - Users can create accounts for themselves and another person by entering details like name, PIN, two-step verification code, and initial balance.
2. **Deposit Money**:
    - Users can securely deposit money into their account by entering their PIN.
3. **Withdraw Money**:
    - Users can withdraw money securely by entering their PIN and specifying the amount.
4. **Check Balance**:
    - Users can view their current account balance. A warning is issued for low balances.

Here are the **additional features** i have added to program beyond the basic ATM functionalities:

---

1. TWO-STEP VERIFICATION FOR PIN RECOVERY

- A **two-step verification mechanism** allows users to reset their ATM PIN securely.
- During account creation, users set a **4-digit verification code**, which is required for PIN recovery.
- This feature enhances the security of the program, providing a safety net for forgotten PINs.

2. ACCOUNT LOCKING FOR SECURITY

- After **three consecutive incorrect PIN attempts**, the account is automatically locked.
- This prevents unauthorized access and enhances the security of the account.

3. LOW BALANCE WARNING

- When users check their balance, the program provides a **low-balance warning** if the balance falls below a certain threshold (e.g., less than 100 units).
- This feature ensures that users are notified to maintain sufficient funds.

4. ENHANCED TRANSACTION HISTORY

- The transaction history now includes detailed records for:
    - Deposits.
    - Withdrawals.
    - Fund transfers (with the recipient or sender's name mentioned).
- This provides users with a clearer understanding of their account activity.

- Extensive **input validation** ensures:
    - Deposits are greater than zero.
    - Withdrawals are within the available balance.
    - Transfer amounts are valid and within the sender's account balance.

6. MULTI-USER SUPPORT

- The program supports the creation and management of **two accounts**:
    - A primary user account.
    - A secondary user account (for fund transfers).
- This adds a multi-user dynamic for fund transfer operations.

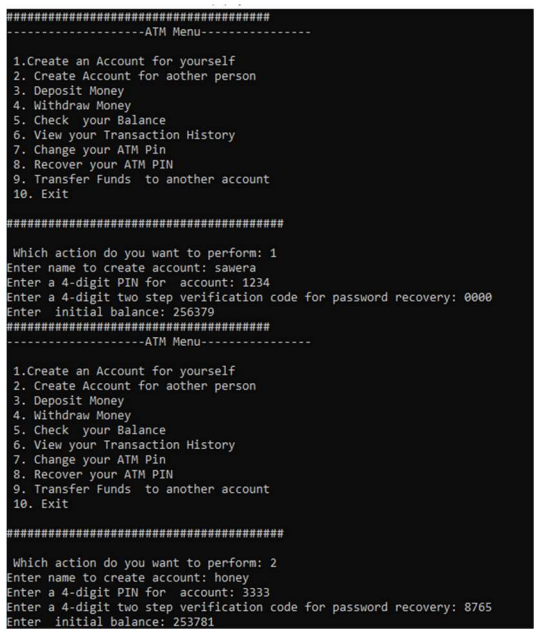
7. SECURE FUND TRANSFER

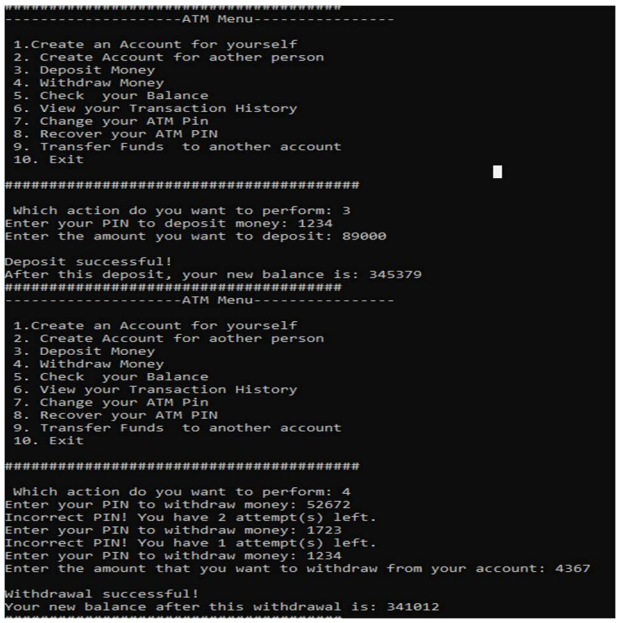
- Added functionality to transfer funds between two accounts with:
    - Validation for sufficient balance in the sender's account.
    - Real-time updates to both accounts' transaction history and balances.

8 FAILED ATTEMPT WARNINGS

- During PIN validation for deposits, withdrawals, and transfers:
    - Users receive a warning showing the number of remaining attempts.
    - If all attempts are exhausted, a **lockout message** is displayed.

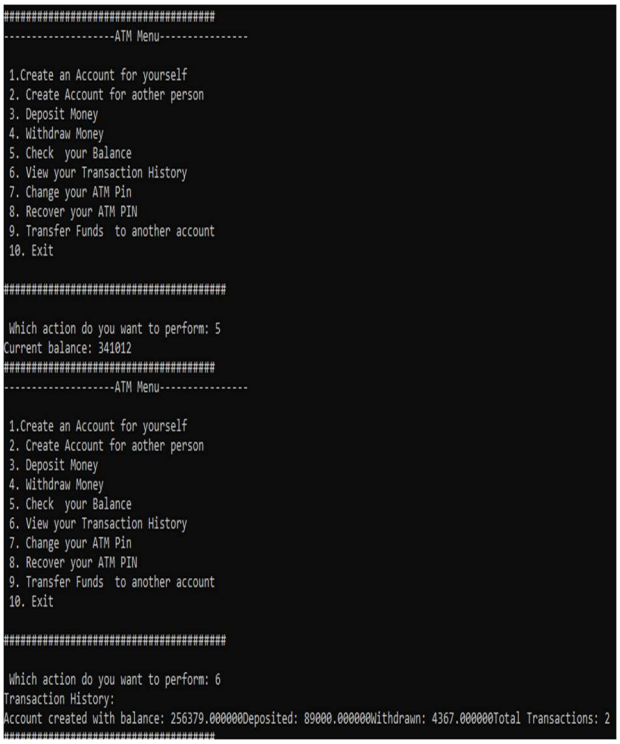
OUTPUT

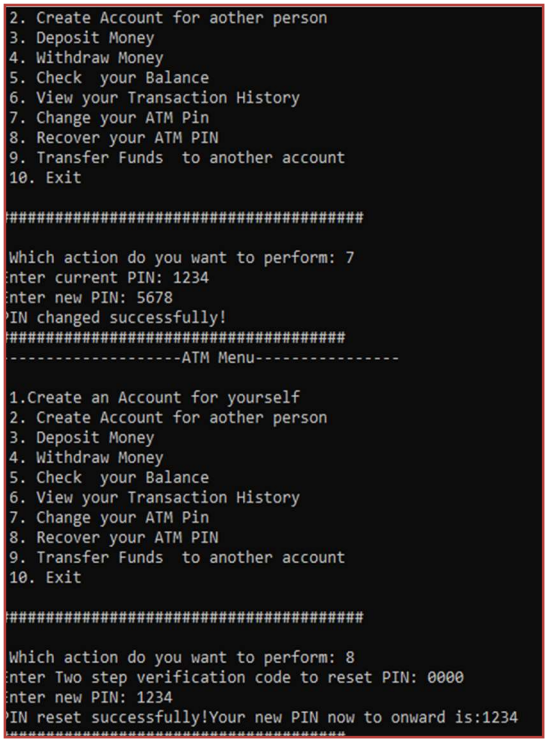


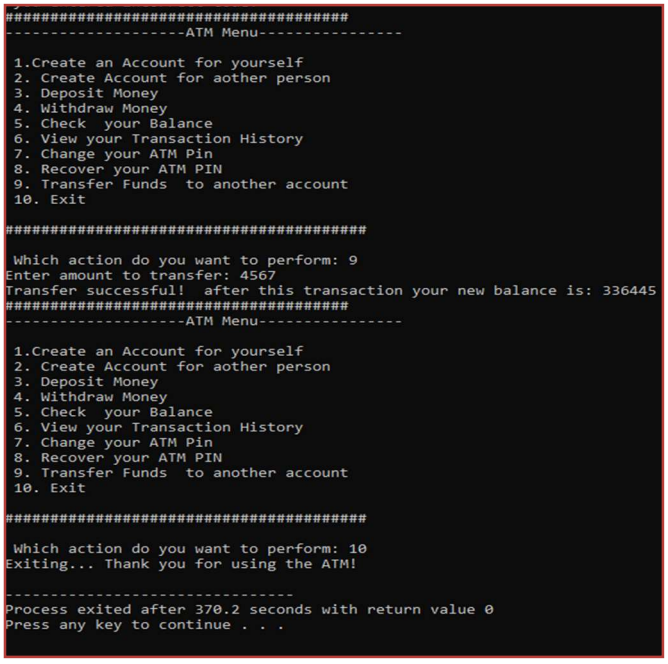


- In the **first picture**, the operations **Create an Account for yourself** and **Create Account for another person** are performed, and their actions are displayed.

- In the **second picture**, the operations **Deposit Money** and **Withdraw Money** are executed, with their corresponding actions shown.







- In the **first picture**, the operations Deposit Money and Withdraw Money and their actions are displayed.

- In the **second picture**, the operations Check your Balance and View your Transaction History are executed, with their corresponding actions shown.

- In the **third picture**, the last two operations are executed, with their corresponding actions shown.

SOURCE CODE

```cpp
#include <iostream>
#include <string>
using namespace std;

struct Account {
    string name;
    int pin;
    double balance;
    string history;
    int countTransaction;
    int NumOfAttempts;
    int  TwoStepVerification;
};
void createAccount(Account &acc) {
    cout << "Enter name to create account: ";
    cin >> acc.name;
    cout << "Enter a 4-digit PIN for  account: ";
    cin >> acc.pin;
    cout << "Enter a 4-digit two step verification code for password recovery: ";
    cin>>acc.TwoStepVerification;
    cout << "Enter  initial balance: ";
    cin >> acc.balance;
    acc.history = "Account created with balance: " + to_string(acc.balance);
    acc.countTransaction = 0;
    acc.NumOfAttempts = 0;
}
void deposit(Account &acc) {
    double amount;
    int pin;
    int maxAttempts = 3;
    int attempts = 0;
    while (attempts < maxAttempts) {
        cout << "Enter your PIN to deposit money: ";
        cin >> pin;
        if (pin == acc.pin) {
            cout << "Enter the amount you want to deposit: ";
            cin >> amount;
            if (amount > 0) {
                acc.balance += amount;
                acc.history += "Deposited: " + to_string(amount) ;
                acc.countTransaction++;
                cout << "\nDeposit successful!";
```

```cpp
                cout << "\nAfter this deposit, your new balance is: " << acc.balance << endl;
                return;
            } else {
                cout << "Invalid amount! Deposit must be greater than 0." << endl;
                return;
            }
        } else {
            attempts++;
            cout << "Incorrect PIN! You have " << (maxAttempts - attempts)
                 << " attempt(s) left." << endl;
        }
    }
    cout << "Maximum attempts reached! Deposit canceled for security reasons & account is locked!!!." << endl;
}
void withdraw(Account &acc) {
    double amount;
    int pin;
    int maxAttempts = 3;
    int attempts = 0;
    while (attempts < maxAttempts) {
        cout << "Enter your PIN to withdraw money: ";
        cin >> pin;
        if (pin == acc.pin) {
            cout << "Enter the amount that you want to withdraw from your account: ";
            cin >> amount;
            if (amount > 0 && amount <= acc.balance) {
                acc.balance -= amount;
                acc.history += "Withdrawn: " + to_string(amount);
                acc.countTransaction++;
                cout << "\nWithdrawal successful!";
                cout << "\nYour new balance after this withdrawal is: " << acc.balance << endl;
                return;
            } else if (amount > acc.balance) {
                cout << "Invalid amount! Your balance is insufficient." << endl;
                return;
            } else {
                cout << "Invalid amount! Withdrawal amount must be greater than 0." << endl;
                return;
            }
        } else {
            attempts++;
            cout << "Incorrect PIN! You have " << (maxAttempts - attempts)
```

```cpp
                 << " attempt(s) left." << endl;
        }
    }
    cout << "Maximum attempts reached! Withdrawal canceled for security reasons & acccount is locked n
}
void checkBalance(const Account &acc) {
    cout << "Current balance: " << acc.balance << endl;
    if (acc.balance < 100) {
        cout << "Warning: Low balance! Please upgrade your balance" << endl;
    }
}
void history(const Account &acc) {
    cout << "Transaction History:\n" << acc.history;
    cout << "Total Transactions: " << acc.countTransaction << endl;
}
void changePin(Account &acc) {
    int oldPin, newPin;
    cout << "Enter current PIN: ";
    cin >> oldPin;
    if (oldPin == acc.pin) {
        cout << "Enter new PIN: ";
        cin >> newPin;
        acc.pin = newPin;
        cout << "PIN changed successfully!" << endl;
    } else {
        acc.NumOfAttempts++;
        cout << "You enter the wrong PIN ,Please enter the correct one!" << endl;
        if (acc.NumOfAttempts >= 3) {
            cout << "Warning: Account is locked due to multiple  incorrect attempts!" << endl;
        }
    }
}
void recoverPin(Account &acc) {
    int code;
    cout << "Enter Two step verification code to reset PIN: ";
    cin>>code;
    if (code == acc.TwoStepVerification) {
        int newPin;
        cout << "Enter new PIN: ";
        cin >> newPin;
        acc.pin = newPin;
        cout << "PIN reset successfully!"<<"Your new PIN now to onward is:"<<newPin << endl;
```

```cpp
}
void transferFunds(Account &sender, Account &receiver) {
    double amount;
    cout << "Enter amount to transfer: ";
    cin >> amount;
    if (amount > 0 && amount <= sender.balance) {
        sender.balance -= amount;
        receiver.balance += amount;
        sender.history += "Transferred: " + to_string(amount) + " to " + receiver.name ;
        receiver.history += "Received: " + to_string(amount) + " from " + sender.name;
        sender.countTransaction++;
        receiver.countTransaction++;
        cout << "Transfer successful!  after this transaction your new balance is: " << sender.balance << endl;
    } else if(amount > sender.balance)  {
        cout <<  "Insufficient balance!"<< endl;
    }
    else{
        cout << "Invalid amount!"<< endl;
    }
}
int main() {
    Account acc1, acc2;
    int choice;
    bool acc1Created= false, acc2Created = false;
    do {
        cout<<"###################################";
        cout << "\n----------------------ATM Menu----------------\n";
        cout << "\n 1.Create an Account for yourself ";
        cout << "\n 2. Create Account for aother person";
        cout << "\n 3. Deposit Money ";
        cout << "\n 4. Withdraw Money ";
        cout << "\n 5. Check  your Balance ";
        cout << "\n 6. View your Transaction History ";
        cout << "\n 7. Change your ATM Pin ";
        cout << "\n 8. Recover your ATM PIN ";
        cout << "\n 9. Transfer Funds  to another account";
        cout << "\n 10. Exit";
        cout<<"\n";
        cout<<"\n###################################";
        cout<<"\n";
        cout << "\n Which action do you want to perform: ";
        cin >> choice;
```

```cpp
        switch (choice) {
            case 1:
                if (!acc1Created) {
                    createAccount(acc1);
                    acc1Created = true;
                } else {
                    cout << "You already created an acoount for yourself!!" << endl;
                }
                break;
            case 2:
                if (!acc2Created) {
                    createAccount(acc2);
                    acc2Created = true;
                } else {
                    cout << " For another  user account is already created!" << endl;
                }
                break;
            case 3:
                if (acc1Created) {
                deposit(acc1);
                }
                else {
                cout << "No such  account found! Please create it first!" << endl;
                }
                break;
            case 4:
                if (acc1Created){
                withdraw(acc1);
                }
                else {
                cout << "No matching account found! Please create it first!" << endl;
                }
                break;
            case 5:
                if (acc1Created){
                    checkBalance(acc1);
                }
                else {
                    cout << "Account not found! Please create it first!" << endl;
                }
                break;
```