CONTENTS

INTRODUCTION

The *Bank Management System* is a command-line application designed to simplify and streamline essential banking operations. It allows the user to perform tasks such as account creation, money deposit/withdrawal, fund transfer, and account modifications. All transactions are logged, and access to the system is protected by a secure PIN.

The software is developed using C++ and focuses on practical implementation of file handling and object-oriented programming concepts. It's especially tailored for internal use, where bank employees can manage customers' accounts more efficiently and securely.

❖ Importance in Real Life :

Banks handle thousands of customer accounts daily, and manually recording each transaction is prone to errors and inefficiencies. A computerized system like this helps reduce those errors, speeds up processing, and offers instant access to account information.

This program can be particularly useful for **bank employees**, enabling them to perform day-to-day operations like deposits, withdrawals, or updating customer details in a quick and organized manner. It reduces dependency on paperwork and helps maintain digital records, which can be retrieved or modified easily when needed. While this project is a mini-model, it mimics real-world banking systems and shows how basic operations can be automated to make everyday banking more secure and efficient.

❖ Motivation :

The motivation behind developing this project came from observing how frequently people rely on banks (whether for savings, business transactions, or digital payments) and how important security and reliability are in banking systems. As a computer science student, I wanted to build something that reflects real-world use, while also helping me apply technical concepts in a meaningful way. Working on this system gave me an opportunity to apply the C++ concepts I've learned in a real-world context and understand how data management, user access control, and digital record-keeping work together in software development.

GENESIS

This project originated from the idea of simulating a basic yet practical banking system that reflects the core functionality of real banks. The goal was to provide a user-friendly interface that can perform essential operations, ensure data integrity, and maintain accurate transaction history.

Using C++ allowed the integration of **object-oriented programming (OOP)** with **file handling**, making it suitable for both learning programming and employees in banking environments who require digital assistance in managing customer data.

OBJECTIVES

The objectives of this Bank Management System project include:

- **To implement real-life banking functionalities using C++.**
  This includes transactions like deposits, withdrawals, and transfers with proper checks and balances.

- **To build a secure system through PIN-based access.**
  Admin access is granted only after verifying a predefined 6-digit PIN, ensuring privacy and control.

- **To provide a secure and efficient way to manage bank accounts digitally.**
  Users should be able to perform tasks like opening, modifying, and deleting accounts with ease.

- **To provide a tool to bank employees**.
  So that they can easily manage customer records without needing large, complex software systems.

- **To enhance understanding of file handling in C++.**
  All account data is stored and retrieved using binary file operations.

- **To log each transaction for accountability and reference.**
  Every operation performed is recorded in a transaction history file.

METHODOLOGY

The development process of the Bank Management System involved several steps:

1. System Design

- **Class-based Design:**
  A single class ***Account*** is used to manage all account-related operations. This follows the OOP principle of encapsulation.
- **Function Separation:**
  Each feature (like deposit, withdraw, transfer, modify) is implemented as a separate function for modularity and reusability.

2. Data Handling

- **File Management:**
  All account records are stored in a binary file (BankAccounts.txt). Transaction history is stored in a text file (TransactionHistory.txt).
- **Input/Output Streams:**
  The system uses ifstream, ofstream, and fstream to read and write data efficiently.

3. Security and Validation

- **Admin PIN Verification:**
  A 6-digit PIN is required to access the system. Multiple incorrect attempts lead to access denial.
- **Account Validations:**
  The system checks whether an account exists before performing transactions.

4. User Interface

- **Menu-based Interface:**
  The main menu allows users to select from options like Create Account, Deposit, Withdraw, Modify, etc.
- **Input Prompts & Feedback:**
  Users are guided with clear instructions and receive confirmation messages after each operation.

CODE IN C++ :-

```
//BANK MANAGEMENT SYSTEM
//A program by HIMANI SHARMA Roll no. 17  MCA 2nd semester

#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

#define ADMIN_PIN "123456"

const int SIZE=31;
```

// Definition of class:-

```
class Account
{
   int acc_no;
   char name[SIZE];
   double balance;
public:
   Account()
   {
      acc_no = 0;
      balance = 0.0;
      name[0] ='\0';
   }

   void getaccountnumber(int accountnumber)
   { acc_no = accountnumber; }
   int putaccountnumber() const
   { return acc_no; }

   void getname(const char* accName)
   { strcpy(name,accName);}
         const char* putname() const
   { return name; }

   void getbalance(double bal)
   { balance = bal; }
   double putbalance() const
   { return balance; }

   void createaccount();
   bool deposit(int,double);
   bool withdraw(int,double);
   void transfer();
   void modify();
   void deleteaccount();
```

```
    void displayaccount();
    void displayAllaccount();
    void viewTransactionHistory();
};
```

// Function to create an account:-

```
void Account :: createaccount()
{
    cout << "Enter Account Number: ";
    cin >> acc_no;

    bool accountFound = false;
    fstream file("BankAccounts.txt", ios::binary | ios::in | ios::out);

    Account acct1;
            while (file.read((char*) &acct1, sizeof(acct1)))
            {
     if (acct1.putaccountnumber() == acc_no)
                    {
                            accountFound = true;
                            break;
                    }
            }
            file.close();

            if(accountFound)
            {
                    cout << "Account already exists !!"<<endl;
     return;
            }
            else
            {
                    cin.ignore();
    cout << "Enter Account Holder Name (max "<< sizeof(name)-1 <<" characters): ";
    cin.getline(name, sizeof(name));

    cout << "Enter Initial Balance: ";
    cin >> balance;

    ofstream outFile("BankAccounts.txt", ios::app | ios::binary);
    if (!outFile)
                    {
            cout << "Error opening file!" << endl;
            return;
    }

    outFile.write((char*) this, sizeof(*this));
    outFile.close();

    cout << "Account created successfully!" << endl;
```

```
                    ofstream log("TransactionHistory.txt", ios::app);
                    log << "\nCreation => A new account created with Acc No.  : " <<
acc_no  <<endl << endl;
                    log.close();
                }
    }
```

// Function to deposit money:-

```
    bool Account :: deposit(int inputacc,double deposit)
        {
        bool accountFound = false;

        fstream file("BankAccounts.txt", ios::in | ios::out | ios::binary);
        if (!file)
                {
            cout << "Error opening file!" << endl;
            return false;
        }

        Account acct1;
        while (file.read((char*) &acct1, sizeof(acct1)))
                {
            if (acct1.putaccountnumber() == inputacc)
                        {
                acct1.getbalance(acct1.putbalance() + deposit);

                file.seekp(-sizeof(acct1), ios::cur);
                file.write((char*) &acct1, sizeof(acct1));
                accountFound = true;
                break;
            }
        }

        file.close();
        if (!accountFound)
                {
            cout << "Account not found!" << endl;
        }

        return accountFound;
    }
```

// Function to withdraw money:-

```
    bool Account :: withdraw(int inputacc,double withdrawal)
        {
        bool accountFound = false;
        bool sufbal =true;

        fstream file("BankAccounts.txt", ios::in | ios::out | ios::binary);
        if (!file)
```

```
            {
        cout << "Error opening file!" << endl;
        return false;
    }

    Account acct1;
    while (file.read((char*) &acct1, sizeof(acct1)))
            {
        if (acct1.putaccountnumber() == inputacc)
                    {
            if (acct1.putbalance() >= withdrawal)
                        {
                acct1.getbalance(acct1.putbalance() - withdrawal);

                file.seekp(-sizeof(acct1), ios::cur);
                file.write((char*) (&acct1), sizeof(acct1));
                accountFound = true;
                break;
            }
                        else
                        {
                cout << "Insufficient balance!" << endl;
                accountFound = true;
                sufbal=false;
                break;
            }
        }
    }

             file.close();
    if (!accountFound)
            {
       cout << "Account not found!" << endl;
    }
    return (accountFound && sufbal);
}
```

//Function to transfer money:-

```
 void Account :: transfer()
{
      int acc1;
      int acc2;
    double transfer;
    bool account1Found=false;
    bool account2Found=false;

    cout << "FROM Account Number: ";
    cin >> acc1;
    cout << "TO Account Number: ";
    cin >> acc2;
    cout << "Enter Amount to be transferred : ";
```

```
        cin >> transfer;

        fstream file("BankAccounts.txt", ios::in | ios::out | ios::binary);
        if (!file)
                {
            cout << "Error opening file!" << endl;
            return;
        }

        Account acct1;
        while (file.read((char*) &acct1, sizeof(acct1)))
                {
            if (acct1.putaccountnumber() == acc1)
                account1Found = true;
                        if (acct1.putaccountnumber() == acc2)
                account2Found = true;
        }

        file.close();

        if(!account1Found)
        {
          cout<<"Account no. "<<acc1<<" not found \nTransfer Failed !!";
          return;
                }
                if(!account2Found)
        {
          cout<<"Account no. "<<acc2<<" not found \nTransfer Failed !!";
          return;
                }

        Account acct;
        acct.withdraw(acc1,transfer);
        acct.deposit(acc2,transfer);
        cout<<"Transfer successful !\n";

        ofstream log("TransactionHistory.txt", ios::app);
                log << "Transfer => Amount : " << transfer << " transferred From Acc No. : "
<< acc1 <<" To Acc No. : "<<acc2 << endl;
                log.close();
          }
```

// **Function to modify an account:-**

```
          void Account :: modify()
          {
                int inputacc;
        bool accountFound = false;
        bool modify=false;

        cout << "Enter Account Number: ";
        cin >> inputacc;
```

```
        fstream file("BankAccounts.txt", ios::binary | ios::in | ios::out);
        if (!file)
                {
           cout << "Error opening file!" << endl;
           return;
        }

        Account acct1;
        while (file.read((char*) &acct1, sizeof(acct1)))
                {
           if (acct1.putaccountnumber() == inputacc)
                        {
                                int choice,newaccno;
                                char newname[SIZE];
                                double newbalance;
              cout << "Choose what you want to modify : 1.Name  2.Balance 3.Account
No.\n";
              cin>>choice;
              switch(choice)
              {
                   case 1:cout<<"Current name = "<<acct1.name<<endl;
                              cin.ignore();
                                                cout<<"Enter new Name :-";
                              cin.getline(newname,sizeof(newname));
                              strcpy(acct1.name,newname);
                              file.seekp(-sizeof(acct1), ios::cur);
                     file.write((char*) (&acct1), sizeof(acct1));
                     cout<<"Name modified successfullly!";
                     modify=true;
                              break;
                   case 2:cout<<"Current balance = "<<acct1.balance<<endl;
                                                cout<<"Enter new Balance :-";
                              cin>>newbalance;
                              acct1.balance=newbalance;
                              file.seekp(-sizeof(acct1), ios::cur);
                     file.write((char*) (&acct1), sizeof(acct1));
                     cout<<"Balance modified successfullly!";
                     modify=true;
                              break;
                   case 3:cout<<"Current Account no. = "<<acct1.acc_no<<endl;
                                                cout<<"Enter new Account no. :-";
                              cin>>newaccno;
                              acct1.acc_no=newaccno;
                              file.seekp(-sizeof(acct1), ios::cur);
                     file.write((char*) (&acct1), sizeof(acct1));
                     cout<<"Account No. modified successfullly!";
                     modify=true;
                              break;
                   default:
                            cout<<"Incorrect choice.Choose 1, 2 or 3.";
```

```
                                }
            accountFound = true;
            break;
          }
       }

       file.close();

       if (!accountFound)
                {
          cout << "Account not found!" << endl;
       }
       if(modify)
       {
         ofstream log("TransactionHistory.txt", ios::app);
                        log << "\nModification => Acc No. : " << acc_no <<" is modified."
<<endl << endl;
                        log.close();
                }

         }
```

// Function to delete an account:-

```
 void Account::deleteaccount()
 {
        int inputacc;
        bool accountDeleted=false;
        cout << "Enter Account Number to delete: ";
        cin >> inputacc;

        char confirm;
        cout << "Are you sure you want to delete this account? (y/n): ";
        cin >> confirm;
        if (confirm != 'y' && confirm != 'Y') return;

        ifstream inFile("BankAccounts.txt", ios::binary);
        ofstream outFile("Temp.txt", ios::binary);

        Account acct1;
        while (inFile.read((char*)&acct1, sizeof(acct1)))
        {
        if (acct1.putaccountnumber() != inputacc)
        {
                outFile.write((char*)&acct1, sizeof(acct1));
        }
        else
        {
                accountDeleted = true;
        }
                }
```

```
        inFile.close();
        outFile.close();

        remove("BankAccounts.txt");
        rename("Temp.txt", "BankAccounts.txt");

                if (accountDeleted)
                {
        cout << "Account deleted successfully!\n";

        ofstream log("TransactionHistory.txt", ios::app);
                        log << "\nDeletion => Acc No. : " << acc_no <<" is deleted." <<endl <<
endl;
                        log.close();
                }
        else
        cout << "Account not found !!\n";
        }
```

// **Function to display an account:-**

```
   void Account :: displayaccount()
        {
      int inputacc;
      bool accountFound = false;

      cout << "Enter Account Number: ";
      cin >> inputacc;

      ifstream inFile("BankAccounts.txt", ios::binary);
      if (!inFile)
                {
       cout << "Error opening file!" << endl;
       return;
      }

      Account acct1;
      while (inFile.read((char*) &acct1, sizeof(acct1)))
                {
       if (acct1.putaccountnumber() == inputacc)
                        {
          cout << "\nAccount Details:\n";
          cout << "Account Number: " << acct1.putaccountnumber() << endl;
          cout << "Account Holder: " << acct1.putname() << endl;
          cout << "Balance: " << acct1.putbalance() << endl;
          accountFound = true;
          break;
       }
      }

      inFile.close();
```

```
    if (!accountFound)
            {
        cout << "Account not found!" << endl;
    }
}
```

// Function to display all accounts :-

```
void Account :: displayAllaccount()
     {
    ifstream inFile("BankAccounts.txt", ios::binary);
    if (!inFile)
            {
        cout << "Error opening file!" << endl;
        return;
    }

    Account acct1;
    cout<<"All accounts details :- \n";
    while (inFile.read((char*) &acct1, sizeof(acct1)))
            {
          cout << "Acct No. "<< acct1.putaccountnumber() << " => "<< acct1.putname()
<< " [ Balance : ";
          cout << acct1.putbalance() << " ]"<<endl;
    }

    inFile.close();
}
```

// Function to view transaction history:-

```
void Account::viewTransactionHistory()
      {
      ifstream file("TransactionHistory.txt");
      if (!file)
      {
      cout << "No transaction history available." << endl;
      return;
      }

      string line;
      cout << "\nTransaction History :- \n";

            while (getline(file, line))
      {
      cout << line << endl;
      }

      file.close();
      }
```

//Function to verify PIN:-

```
bool verifyPIN()
{
string input;
int attempts = 4;

while (attempts != 0)
{
system("cls");
    cout << "\n\n";
          cout << "========================================\n";
          cout << "          BANK MANAGEMENT LOGIN        \n";
cout << "========================================\n\n";
cout << "      Enter 6-digit PIN to continue     \n";
cout << "      Attempts left: " << attempts << "\n\n";
cout << "========================================\n";
cout << "Enter PIN: ";
cin >> input;

if (input == ADMIN_PIN)
{
          cout << "\nAccess Granted. Welcome Admin!\n";
          system("pause");
          system("cls");
          return true;
}
else
{
          if(attempts>1)
          {
                                cout << "\nIncorrect PIN! Please try again.\n";
                    system("pause");
          }
          attempts--;
 }
}

cout << "\nToo many incorrect attempts. Access denied !!\n";
return false;
}
```

// Main function :-

```
int main()
{
    Account user;
    int choice;

          if (!verifyPIN())
          {
```

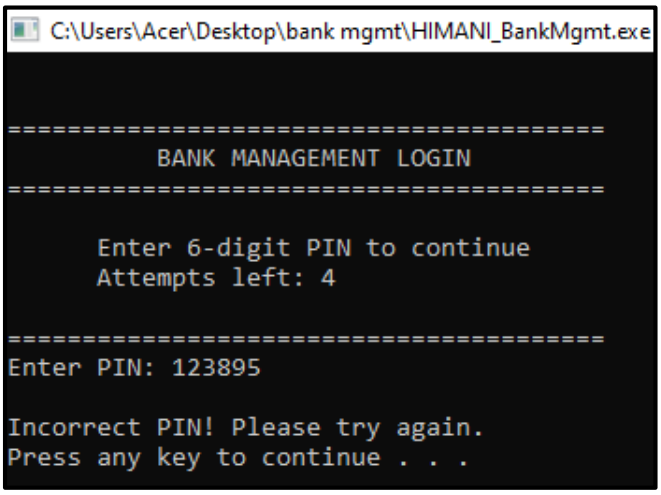
```
        cout << "\nExiting program....." << endl;
        return 0;
    }

    while (1)
    {
        cout << "\n--------------------------\n";
                cout << "Bank Management System\n";
        cout << "--------------------------\n";
        cout << "1. Create Account\n";
        cout << "2. Deposit Money\n";
        cout << "3. Withdraw Money\n";
        cout << "4. Transfer Money\n";
        cout << "5. Modify Account\n";
        cout << "6. Delete Account\n";
        cout << "7. Display Single Account Details\n";
        cout << "8. Display All Account Details\n";
        cout << "9. View Transaction History\n";
        cout << "10. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice)
        {
            case 1:
                user.createaccount();
                break;
            case 2:
                    int inputacc;
                    double deposit;
                    bool accountFound;
                    cout << "Enter Account Number: ";
                    cin >> inputacc;
                    cout << "Enter Deposit Amount: ";
                    cin >> deposit;
                accountFound=user.deposit(inputacc,deposit);
                if (accountFound)
                                {
                        cout << "Deposit successful!" << endl;

                        ofstream log("TransactionHistory.txt", ios::app);
                                log << "Deposit => Amount : " << deposit << " deposited
from Acc No. : " << inputacc << endl;
                                log.close();
                    }
                                else
                                {
                        cout << "Deposit Failed !!" << endl;
                    }
                break;
            case 3:
```

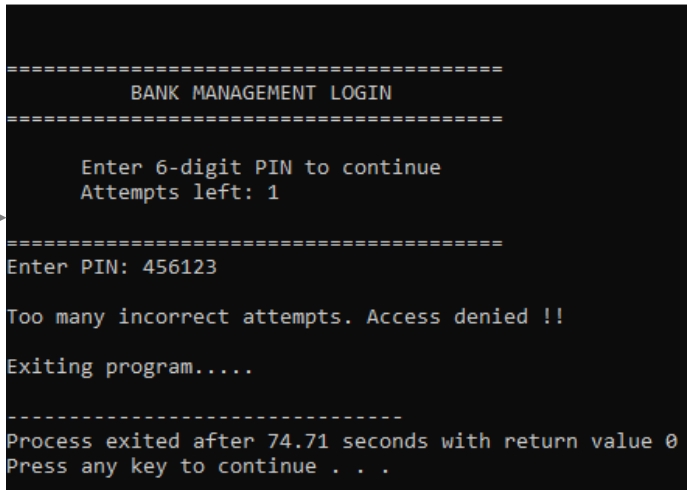
```
                double withdrawal;
                bool success;
                cout << "Enter Account Number: ";
                cin >> inputacc;
                cout << "Enter Withdrawl Amount: ";
                cin >> withdrawal;
            success=user.withdraw(inputacc,withdrawal);
             if (success)
                                {
                        cout << "Withdrawal successful!" << endl;

                        ofstream log("TransactionHistory.txt", ios::app);
                                log << "Withdraw => Amount : " << withdrawal << "
withdrawn from Acc No. : " << inputacc << endl;
                                log.close();
                }
                else
                {
                        cout << "Withdrawal failed !!" << endl;
                                }
            break;
         case 4:
            user.transfer();
            break;
         case 5:
            user.modify();
            break;
         case 6:
            user.deleteaccount();
            break;
         case 7:
            user.displayaccount();
            break;
         case 8:
                user.displayAllaccount();
            break;
         case 9:
                system("cls");
                user.viewTransactionHistory();
                break;
         case 10:
            cout << "Exiting the program...\n";
            exit(0);
         default:
            cout << "Invalid choice! Please try again.\n";
      }
   }

   return 0;
}
```

OUTPUTS :-

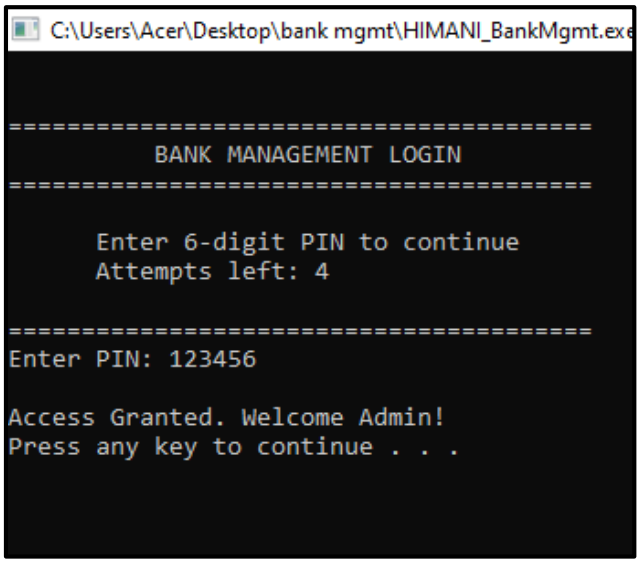
➢ Entering PIN :-

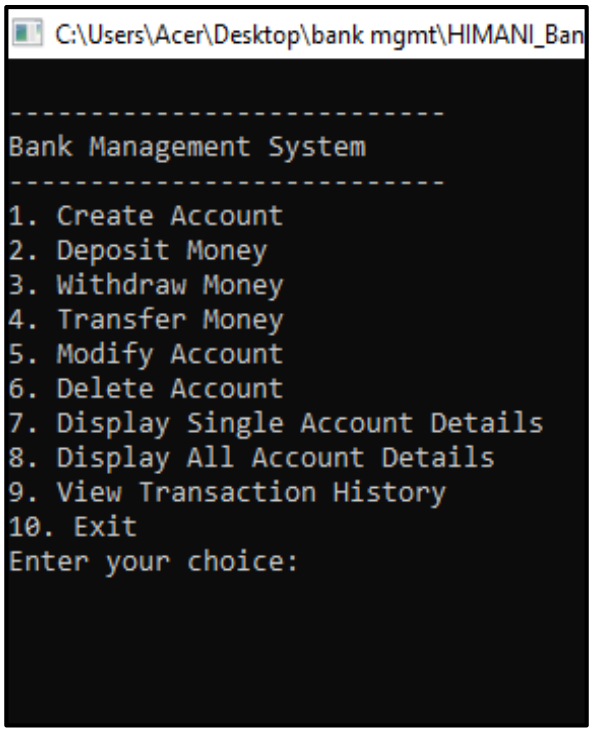
- **Incorrect pin :-**





- **Correct pin :-**





➢ Creating accounts :-

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 1
Enter Account Number: 101
Enter Account Holder Name (max 30 characters): Himani
Enter Initial Balance: 5000
Account created successfully!
```

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 1
Enter Account Number: 102
Enter Account Holder Name (max 30 characters): Amit
Enter Initial Balance: 6000
Account created successfully!
```

```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 1
Enter Account Number: 101
Account already exists !!
```

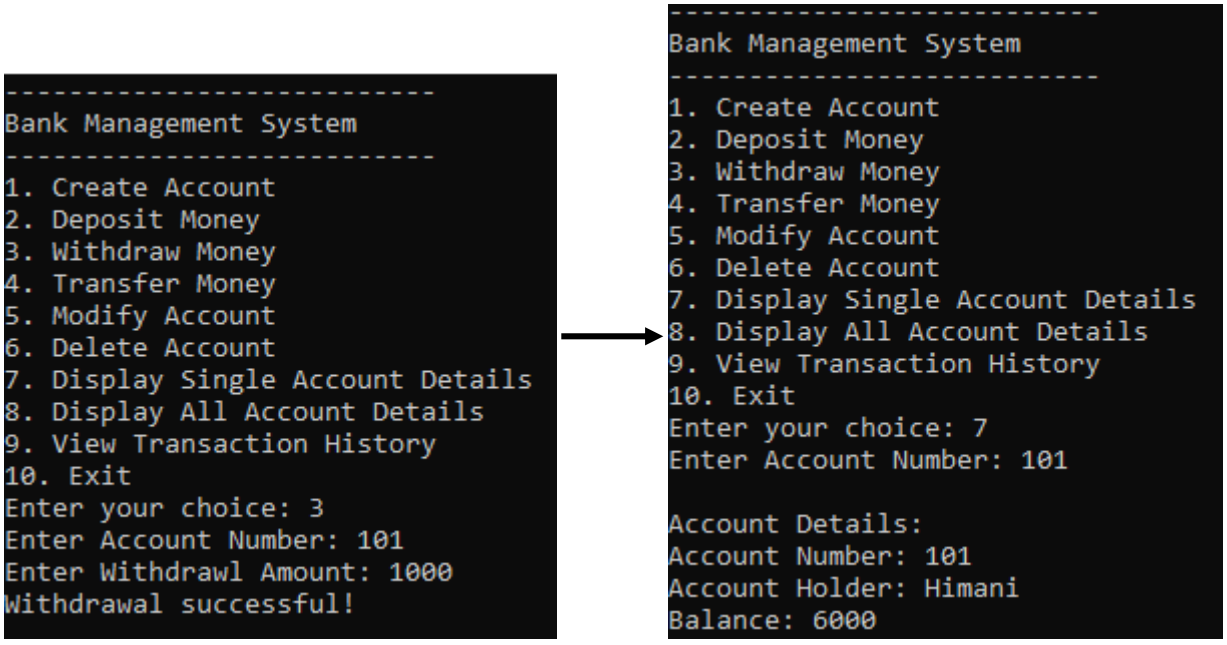
➢ Displaying account and depositing money :-

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 7
Enter Account Number: 101

Account Details:
Account Number: 101
Account Holder: Himani
Balance: 5000
```

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 2
Enter Account Number: 101
Enter Deposit Amount: 2000
Deposit successful!
```

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 7
Enter Account Number: 101

Account Details:
Account Number: 101
Account Holder: Himani
Balance: 7000
```

➢ Withdrawing money :-



```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 3
Enter Account Number: 101
Enter Withdrawl Amount: 1000
Withdrawal successful!
```

```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 7
Enter Account Number: 101

Account Details:
Account Number: 101
Account Holder: Himani
Balance: 6000
```

➢ Modifying account :-

```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 5
Enter Account Number: 102
Choose what you want to modify : 1.Name  2.Balance 3.Account No.
1
Current name = Amit
Enter new Name :-Atul
Name modified successfullly!
```

➢ Displaying all accounts :-

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 8
All accounts details :-
Acct No. 101 => Himani [ Balance : 6000 ]
Acct No. 102 => Atul [ Balance : 6000 ]
Acct No. 103 => Akshay [ Balance : 10000 ]
```

➢ Deleting an account :-

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 6
Enter Account Number to delete: 103
Are you sure you want to delete this account? (y/n): y
Account deleted successfully!
```

```
--------------------------
Bank Management System
--------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 8
All accounts details :-
Acct No. 101 => Himani [ Balance : 6000 ]
Acct No. 102 => Atul [ Balance : 6000 ]
```

➢ Transferring Money :-

```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 4
FROM Account Number: 102
TO Account Number: 101
Enter Amount to be transferred : 4000
Transfer successful !
```

```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 8
All accounts details :-
Acct No. 101 => Himani [ Balance : 10000 ]
Acct No. 102 => Atul [ Balance : 2000 ]
```

➢ Viewing Transaction History :-

```
Transaction History :-

Creation => A new account created with Acc No.  : 101


Creation => A new account created with Acc No.  : 102


Creation => A new account created with Acc No.  : 103

Deposit => Amount : 2000 deposited from Acc No. : 101
Withdraw => Amount : 1000 withdrawn from Acc No. : 101

Modification => Acc No. : 103 is modified.


Deletion => Acc No. : 103 is deleted.


Transfer => Amount : 4000 transferred From Acc No. : 102 To Acc No. : 101
```

➢ Exiting the Program :-

```
---------------------------
Bank Management System
---------------------------
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Modify Account
6. Delete Account
7. Display Single Account Details
8. Display All Account Details
9. View Transaction History
10. Exit
Enter your choice: 10
Exiting the program...

--------------------------------
Process exited after 759.3 seconds with return value 0
Press any key to continue . . .
```

CONCLUSION

The **Bank Management System** has successfully achieved its goal of providing a simple, efficient, and secure way to manage bank accounts digitally.

Achievements of this project can be summarised as:

✓ Realistic simulation of banking operations.

✓ Hands-on experience with object oriented programming and file handling in C++.

✓ Understanding data security through access control and transaction logging.

✓ Application of Object-Oriented Programming concepts.

✓ A flexible model that can be expanded into a full-fledged banking software in the future.

This project has been a valuable learning experience. It not only helped me practice the programming skills I've learned so far but also gave me insight into real-world software challenges and how to overcome them.

By working on this project, I was able to bridge the gap between theoretical learning and practical implementation. It has also helped me appreciate the importance of software in sectors like banking, where accuracy and trust are critical.