DEPARTMENT OF UNIVERSITY INSTITUTE OF COMPUTING CHANDIGARH UNIVERSITY

Project

Title : Bank Account Management System

Submitted by :

Name : Uday Rana

UID : 24BCA10397

Section : 24BCA - 3(A)

Submitted to :

Name : Ms. Joyti Rani

Subject: Object Oriented Programming

Sub.code: 24CAH-201

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Ms. Jyoti Rani, my respected teacher, for her constant guidance, support, and encouragement throughout the completion of my project titled “Bank Account Management System using C++”.

Her valuable suggestions and motivation helped me to understand the concepts clearly and complete this project successfully.

I would also like to thank my parents, friends, and the entire faculty of Chandigarh University for their cooperation and inspiration.

Finally, I am thankful to all those who directly or indirectly helped me in making this project a success.

Uday Rana
Student, BCA
Chandigarh University

Introduction

The Bank Account Management System is a simple console-based mini project developed in C++ using the concept of Object-Oriented Programming (OOP) and basic file-less data management.
The main purpose of this project is to simulate basic banking operations such as account creation, deposit, withdrawal, balance inquiry, and account modification.
It helps understand how OOP concepts like data encapsulation, functions, structures, and modularity can be used to develop real-world applications.

Objective

- The primary objectives of this project are:
- To implement basic banking operations in C++.
- To understand and apply OOP concepts.
- To design a user-friendly menu-driven system.
- To differentiate between Admin and User operations.
- To demonstrate how data can be stored and accessed using in-memory structures.

System Overview

The system provides two modes of operation:

1. **Admin Mode** – allows the administrator to create, modify, or delete user accounts.
2. **User Mode** – allows a user to deposit or withdraw money and check account details.

Data is stored temporarily in memory using a vector of structures, and the program resets once closed (no permanent storage).

Functionalities

Admin Functions:

- **Create New Account:** Admin can register a new account by entering details such as account number, holder name, father's name, and initial amount.
- **Display All Accounts:** Displays the list of all existing accounts.
- **Modify Account:** Allows the admin to update name or balance.
- **Delete Account:** Removes a user account from the system.

User Functions:

- **Deposit Money:** Add a specified amount to the account balance.
- **Withdraw Money:** Deduct a specified amount (if balance is sufficient).
- **Balance Inquiry:** Display account holder details and current balance.

Features

- Menu-driven console interface.
- Admin/User login system.
- Error handling for invalid input.
- Secure access to admin functions.
- Clear separation of user and admin roles.
- Dynamic in-memory storage using std::vector.

Tools & Technologies

Component	**Description**
Language used	C++
Concepts Applied	Structures, Functions, Loops, Conditional Statements
Compiler/IDE	VS Code
Operating	Windows

System Flow

Main Menu

1. Admin Login
2. User Login
3. Exit

- **Admin** → Access to Create, Modify, Delete, and View Accounts
- **User** → Access to Deposit, Withdraw, and Check Balance

Advantages

- Improves understanding of data management in C++.
- Demonstrates modular and structured programming.
- Easy to expand for file-based or database-based systems.
- Practical implementation of user authentication and access control.

Limitations

- Data is not stored permanently (resets after exit).
- Does not include file handling or database.
- No graphical user interface (text-based only).

Future Enhancements

- Add file handling to store account data permanently.
- Introduce interest calculation and transaction history.
- Implement password protection per account.
- Upgrade to a GUI-based system using frameworks like Qt or Tkinter (in Python).

Code

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <cstdlib>
Using namespace std;

Class Account {
    Int accNo;
    String name;
    String type;
    Double balance;

Public:
    Void createAccount();
    Void showAccount() const;
    Void modify();
    Void deposit(double);
    Void withdraw(double);
    Void report() const;
    Int getAccNo() const { return accNo; }
    Double getBalance() const { return balance; }
    String getType() const { return type; }
};

// ------------------ Account Methods ------------------

Void Account::createAccount() {
    Cout << "\nEnter Account Number: ";
    Cin >> accNo;
    Cin.ignore();
    Cout << "Enter Account Holder Name: ";
    Getline(cin, name);
    Cout << "Enter Account Type (S = Savings / C = Current): ";
    Cin >> type;
    Type = (type == "S" || type == "s") ? "Savings" : "Current";
    Cout << "Enter Initial Deposit (>=500 for Savings, >=1000 for Current):
";
    Cin >> balance;
    Cout << "\nAccount Created Successfully!\n";
}

Void Account::showAccount() const {
    Cout << "\nAccount No: " << accNo;
    Cout << "\nName: " << name;
    Cout << "\nType: " << type;
    Cout << "\nBalance: " << balance << endl;
}

Void Account::modify() {
    Cout << "\nModify Account Holder Name: ";
    Cin.ignore();
    Getline(cin, name);
    Cout << "Modify Type (S/C): ";
    Cin >> type;
    Type = (type == "S" || type == "s") ? "Savings" : "Current";
    Cout << "Modify Balance: ";
    Cin >> balance;
```

```
}

Void Account::deposit(double amt) {
    Balance += amt;
}

Void Account::withdraw(double amt) {
    Balance -= amt;
}

Void Account::report() const {
    Cout << setw(10) << accNo
         << setw(20) << name
         << setw(10) << type
         << setw(10) << fixed << setprecision(2) << balance << endl;
}

// ------------------ Function Prototypes ------------------

Void writeAccount();
Void displayAccount(int);
Void modifyAccount(int);
Void deleteAccount(int);
Void displayAll();
Void depositWithdraw(int, int);
Bool adminLogin();
Bool userLogin();

// ------------------ Main Function ------------------
```

```
Int main() {
   Char choice;
   Int num;

   While (true) {
      System("cls"); // clear screen for Windows
      Cout  <<  "\n\n***  BANK  ACCOUNT  MANAGEMENT  SYSTEM
***\n";
      Cout << "\n1. Admin Login";
      Cout << "\n2. User Login";
      Cout << "\n3. Exit";
      Cout << "\n\nSelect option: ";
      Cin >> choice;

      If (choice == '1') {
         If (adminLogin()) {
            Char ch;
            Do {
               System("cls");
               Cout << "\n*** ADMIN MENU ***\n";
               Cout << "\n1. Create New Account";
               Cout << "\n2. Display All Accounts";
               Cout << "\n3. Delete an Account";
               Cout << "\n4. Modify an Account";
               Cout << "\n5. Back to Main Menu";
               Cout << "\n\nEnter choice: ";
               Cin >> ch;
               Switch (ch) {
                  Case '1': writeAccount(); break;
                  Case '2': displayAll(); break;
                  Case ‘3’:
                     Cout << “Enter Account No: “; cin >> num;
                     deleteAccount(num);
                     break;
                  case ‘4’:
                     cout << “Enter Account No: “; cin >> num;
                     modifyAccount(num);
                     break;
                  case ‘5’: break;
                  default: cout << “\nInvalid Option!\n”;
               }
               Cin.ignore();
               Cin.get();
            } while (ch != ‘5’);
         }
      }
      Else if (choice == ‘2’) {
         If (userLogin()) {
            Char ch;
            Do {
               System(“cls”);
               Cout << “\n*** USER MENU ***\n”;
               Cout << “\n1. Deposit Money”;
               Cout << “\n2. Withdraw Money”;
               Cout << “\n3. Balance Inquiry”;
               Cout << “\n4. Back to Main Menu”;
               Cout << “\n\nEnter choice: “;
               Cin >> ch;
               Switch (ch) {
                  Case ‘1’:
         }
                     Cout << “Enter Account No: “; cin >> num;
                     depositWithdraw(num, 1);
                     break;
                  case ‘2’:
                     cout << “Enter Account No: “; cin >> num;
                     depositWithdraw(num, 2);
                     break;
                  case ‘3’:
                     cout << “Enter Account No: “; cin >> num;
                     displayAccount(num);
                     break;
                  case ‘4’: break;
                  default: cout << “\nInvalid Option!\n”;
               }
               Cin.ignore();
               Cin.get();
            } while (ch != ‘4’);
         }
      }
      Else if (choice == ‘3’) {
         Cout << “\nThank you for using the system!\n”;
         Break;
      }
      Else {
         Cout << “\nInvalid Option!\n”;
      }
         }

         Return 0;
       }
```

```
// ------------------ Function Definitions ------------------

Void writeAccount() {
   Account ac;
   Ofstream outFile("accounts.dat", ios::binary | ios::app);
   Ac.createAccount();
   outFile.write(reinterpret_cast<char*>(&ac), sizeof(Account));
   outFile.close();
}

Void displayAccount(int n) {
   Account ac;
   Bool found = false;
   Ifstream inFile("accounts.dat", ios::binary);
   If (!inFile) {
      Cout << "File could not be opened!";
      Return;
   }
   While (inFile.read(reinterpret_cast<char*>(&ac), sizeof(Account))) {
      If (ac.getAccNo() == n) {
         Ac.showAccount();
         Found = true;
      }
   }
   inFile.close();
   if (!found)
      cout << "\nAccount not found!\n";
}
```

```
Void modifyAccount(int n) {
   Bool found = false;
   Account ac;
   Fstream file(“accounts.dat”, ios::binary | ios::in | ios::out);
   If (!file) {
      Cout << “File could not be opened!”;
      Return;
   }
   While (!file.eof() && found == false) {
      Int pos = file.tellg();
      File.read(reinterpret_cast<char*>(&ac), sizeof(Account));
      If (ac.getAccNo() == n) {
         Ac.showAccount();
         Cout << “\nEnter new details:\n”;
         Ac.modify();
         File.seekp(pos);
         File.write(reinterpret_cast<char*>(&ac), sizeof(Account));
         Cout << “\nRecord Updated!\n”;
         Found = true;
      }
   }
   File.close();
   If (!found)
      Cout << “\nRecord Not Found!\n”;
}

Void deleteAccount(int n) {
   Account ac;
   Ifstream inFile(“accounts.dat”, ios::binary);
   Ofstream outFile(“temp.dat”, ios::binary);
   If (!inFile) {
      Cout << “File could not be opened!”;
      Return;
   }
   While (inFile.read(reinterpret_cast<char*>(&ac), sizeof(Account))) {
      If (ac.getAccNo() != n)
         outFile.write(reinterpret_cast<char*>(&ac), sizeof(Account));
   }
   inFile.close();
   outFile.close();
   remove(“accounts.dat”);
   rename(“temp.dat”, “accounts.dat”);
   cout << “\nRecord Deleted Successfully!\n”;
}

Void displayAll() {
   Account ac;
   Ifstream inFile(“accounts.dat”, ios::binary);
   If (!inFile) {
      Cout << “File could not be opened!”;
      Return;
   }
   Cout << “\n\nACCOUNT HOLDER LIST\n”;
   Cout                                                              <<
“======================================================\n
”;
   Cout << setw(10) << “A/c No” << setw(20) << “Name” << setw(10) <<
“Type” << setw(10) << “Balance\n”;
   Cout                                                              <<
“======================================================\n
```

```
";
    While (inFile.read(reinterpret_cast<char*>(&ac), sizeof(Account))) {
        Ac.report();
    }
    inFile.close();
}

Void depositWithdraw(int n, int option) {
    Double amt;
    Bool found = false;
    Account ac;
    Fstream file("accounts.dat", ios::binary | ios::in | ios::out);
    If (!file) {
        Cout << "File could not be opened!";
        Return;
    }
    While (!file.eof() && found == false) {
        Int pos = file.tellg();
        File.read(reinterpret_cast<char*>(&ac), sizeof(Account));
        If (ac.getAccNo() == n) {
            Ac.showAccount();
            If (option == 1) {
                Cout << "\nEnter amount to deposit: ";
                Cin >> amt;
                Ac.deposit(amt);
            }
            If (option == 2) {
                Cout << "\nEnter amount to withdraw: ";
                Cin >> amt;
                Double bal = ac.getBalance() – amt;
                If ((bal < 500 && ac.getType() == “Savings”) || (bal < 1000 &&
ac.getType() == “Current”))
                Cout << “\nInsufficient Balance!\n”;
                Else
                Ac.withdraw(amt);
            }
            File.seekp(pos);
            File.write(reinterpret_cast<char*>(&ac), sizeof(Account));
            Cout << “\nTransaction Successful!\n”;
            Found = true;
        }
    }
    File.close();
    If (!found)
        Cout << “\nRecord Not Found!\n”;
}

// ------------------ Login Systems ------------------

Bool adminLogin() {
    String user, pass;
    Cout << “\nEnter Admin Username: “;
    Cin >> user;
    Cout << “Enter Admin Password: “;
    Cin >> pass;
    If (user == “admin” && pass == “1234”) {
        Cout << “\nLogin Successful!\n”;
        Return true;
    } else {
        Cout << “\nInvalid Credentials!\n”;
```

```
        Return false;
    }
}

Bool userLogin() {
    String user, pass;
    Cout << "\nEnter User ID: ";
    Cin >> user;
    Cout << "Enter Password: ";
    Cin >> pass;
    If (user == "user" && pass == "0000") {
        Cout << "\nLogin Successful!\n";
        Return true;
    } else {
        Cout << "\nInvalid Credentials!\n";
        Return false;
    }
}
```

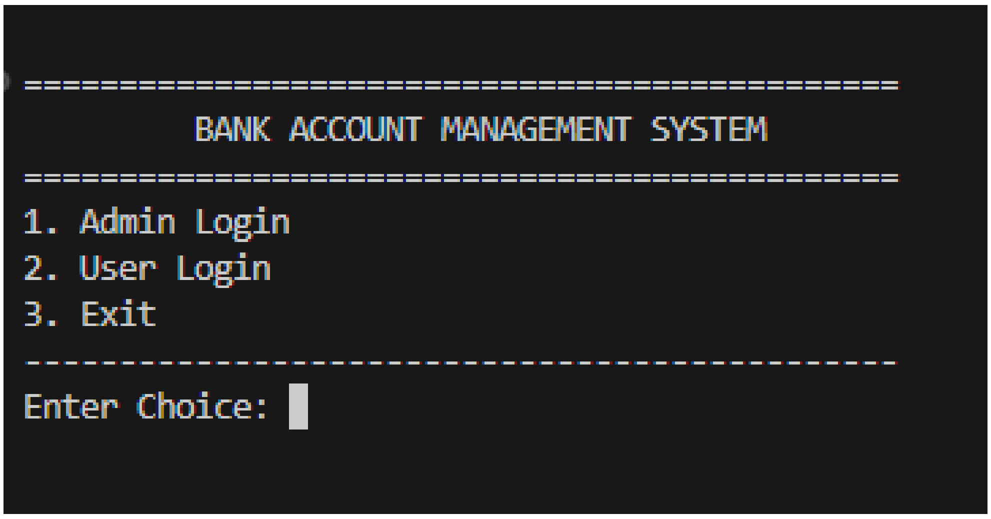
Conclusion

This project demonstrates how fundamental programming concepts can be applied to solve real-life problems such as managing bank accounts.

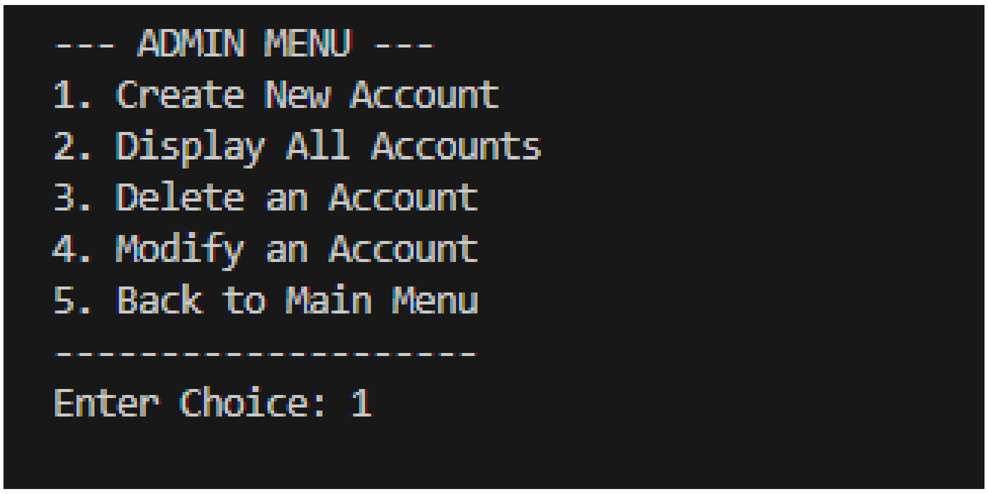
It provides a practical understanding of object-oriented design, modular programming, and user interaction in C++.

The project can serve as a foundation for developing more advanced financial or database systems.

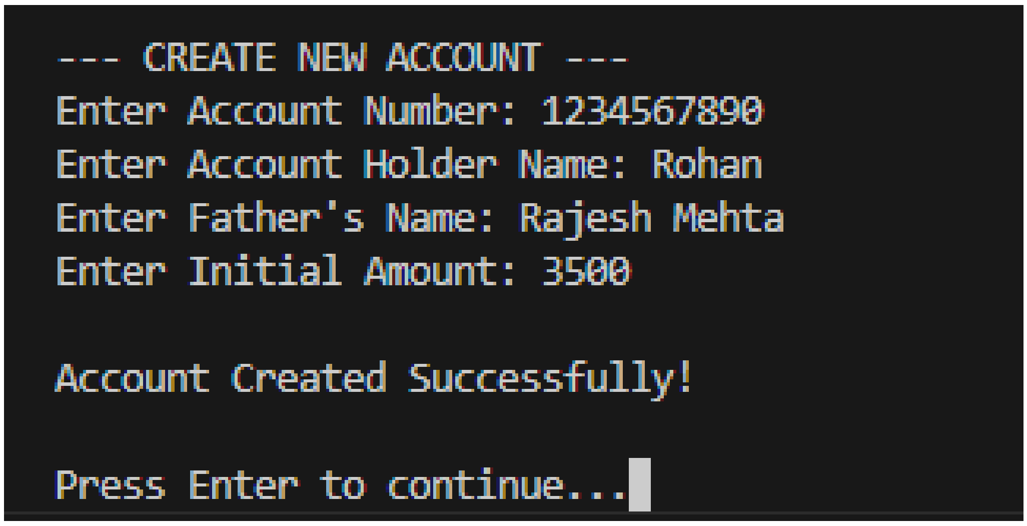
Output Screens



```
==============================================
        BANK ACCOUNT MANAGEMENT SYSTEM
==============================================
1. Admin Login
2. User Login
3. Exit
----------------------------------------------
Enter Choice: 
```



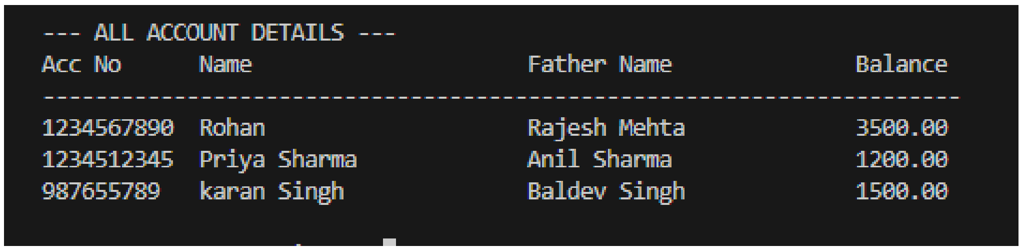
```
--- ADMIN MENU ---
1. Create New Account
2. Display All Accounts
3. Delete an Account
4. Modify an Account
5. Back to Main Menu
--------------------
Enter Choice: 1
```



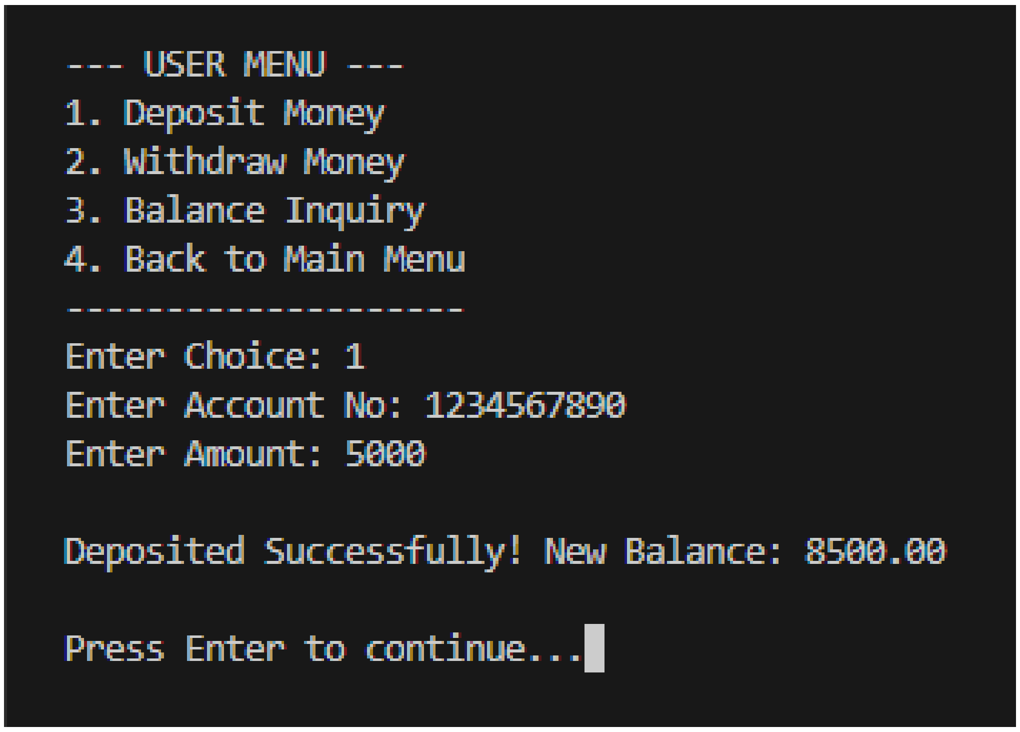
```
--- CREATE NEW ACCOUNT ---
Enter Account Number: 1234567890
Enter Account Holder Name: Rohan
Enter Father's Name: Rajesh Mehta
Enter Initial Amount: 3500

Account Created Successfully!

Press Enter to continue...
```



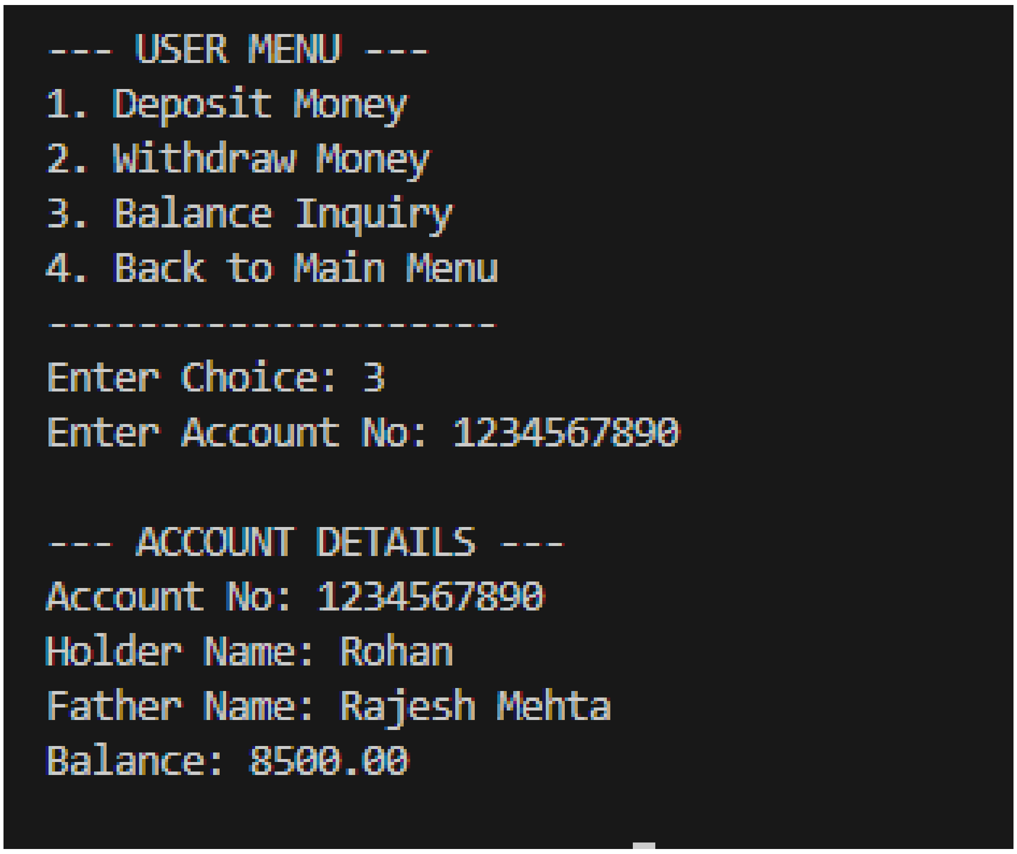
```
--- ALL ACCOUNT DETAILS ---
Acc No      Name                     Father Name              Balance
----------------------------------------------------------------------
1234567890  Rohan                    Rajesh Mehta             3500.00
1234512345  Priya Sharma             Anil Sharma              1200.00
987655789   karan Singh              Baldev Singh             1500.00
```



```
--- USER MENU ---
1. Deposit Money
2. Withdraw Money
3. Balance Inquiry
4. Back to Main Menu
--------------------
Enter Choice: 1
Enter Account No: 1234567890
Enter Amount: 5000

Deposited Successfully! New Balance: 8500.00

Press Enter to continue...
```



```
--- USER MENU ---
1. Deposit Money
2. Withdraw Money
3. Balance Inquiry
4. Back to Main Menu
--------------------
Enter Choice: 3
Enter Account No: 1234567890

--- ACCOUNT DETAILS ---
Account No: 1234567890
Holder Name: Rohan
Father Name: Rajesh Mehta
Balance: 8500.00
```