Laboratory Report

On

Bank Account Management System

(OOP Lab)

Submitted by:

Archit Sarkar (1930172)

Aayushi Singh (1930166)

Pranay Kumar (1930177)

Pritom Mukherjee (1930181)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

B.Tech Programme in Electronics and Computer Science Engineering

School of Electronics Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Bhubaneswar, India

April 2021

CONTENT :

In the present system, all the banking work is done manually and it is also difficult to find the account information of account holders. In this bank management system, we will automate all the banking processes. This "**Bank Account Management System"** allows users to add new customer accounts, delete accounts and users can also modify existing user account information. Also, using this system users can search any individual account in a few seconds.

PROBLEM DEFINITION & PROPOSED SOLUTION :

The main aim of this project is to develop a system for the Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems. The Bank Account Management System keeps the day by day tally record as a complete banking system. It can keep the information of accounts, account opening form, balance and search the transaction details. The main objective of the system is to automate all the banking processes with improved performance and realize the vision of paperless banking. Using this bank management system any information can be easily searched. Users can view all the details of the customer. The proposed system provides faster data access, data entry and retrieval. The users can manage a large number of customer details with ease.

FUNCTIONS & APPLICATION USERS :

The Bank Account Management System is an application for maintaining a person's account in the banking sector. The user of this system is mainly employees working in banks. This system enables the user to maintain the information of the customer efficiently.

The system provides the access to the employee to :-

- ❖ Add record
- ❖ Show record
- ❖ Modify record
- ❖ Delete record

Goals and Objectives:-

- ❑ The primary aim of this **"Bank Account Management System"** is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for the banking sector.
- ❑ Our motto is to develop a software program for managing the entire bank process related to Administration of customer's accounts and to keep each and every track about their various transaction processes efficiently.

- It helps in protection of the customer's account, money and his privacy.

CLASSES, DATA MEMBERS & MEMBER FUNCTIONS :

NAME OF THE CLASS :-	1. Account_Intro		2. Account_Data	
DATA -MEMBER :	-		account_number[20] :- firstName[10]:- lastName[10] :- total_Balance :-	char char char char
MEMBER FUNCTION :-	instruction():-	void	read_data() :- show_data() :-	void void

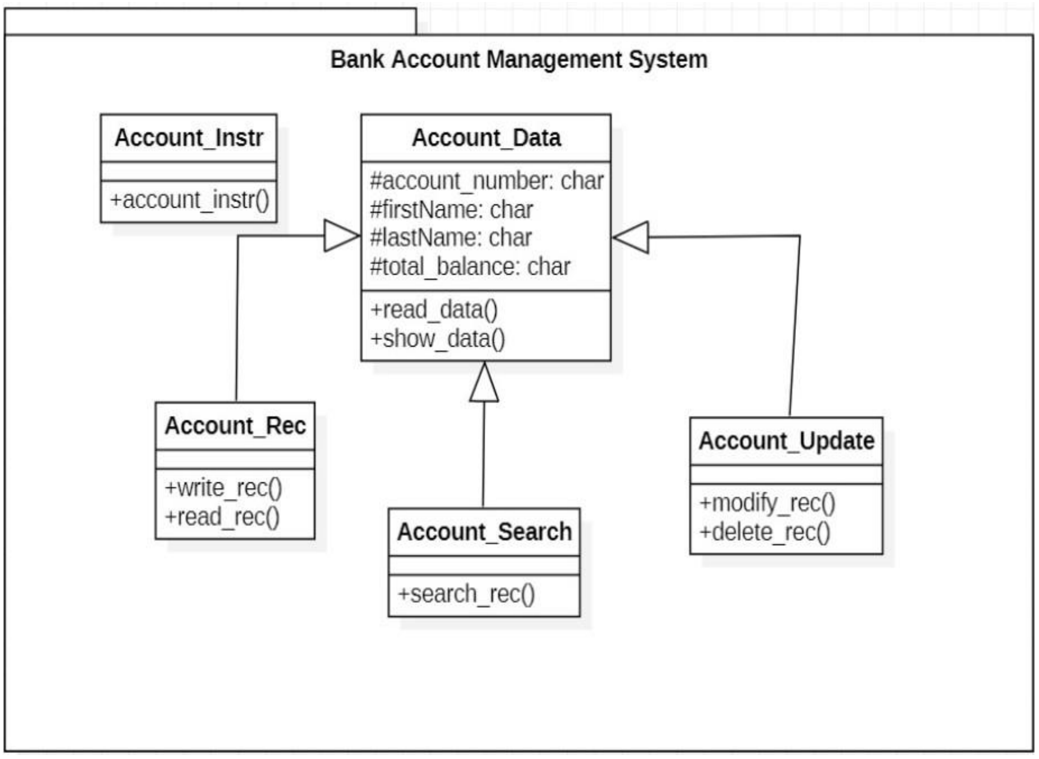
NAME OF THE CLASS :-	3. Account_Rec		4. Account_Search	
DATA-MEMBER :-	-		-	
MEMBER FUNCTION :-	write_rec() :- read_rec() :-	void void	search_rec() :-	void

NAME OF THE CLASS :-	5. Account_Update	
DATA-MEMBER :-	-	
MEMBER FUNCTION :-	modify_rec() :- delete_rec() :-	void void

CLASS DIAGRAMS WITH RELATIONSHIPS :

The created program contains a total of five classes which are namely :
a)Account_Instr
b)Account_Data
c)Account_Rec
d)Account_Update
e)Account_Search

Class Account_Instr exists as an independent class to show the interface and details of the project. The class Account_Data holds the data variables and is the parent class for the remaining classes. This Inheritance is an example of Hierarchical inheritance.



ALGORITHMS:

1. Write Operation

- Declare class ofstream and create object fout .
- Open record.txt file in append mode . By opening in append mode the previous datas won't delete.
- Check for errors.
- Call read_data() function declared in Account_data class.
- The fout.write will write the entered data in record.txt file.
- After writing the data in the file, close the file using fout.close().

2. Read Operation

- Declare class ifstream and create object fin .
- Open record.txt file in append mode . By opening in append mode the previous datas won't delete.
- Check for errors.
- Read the datas present in the file in the output screen till the end of file [fin.eof()] is reached.
- Call show_data(). Datas are shown on screen till the end of file.
- After reading the data from the file, close the file using fin.close().

3. Search Operation

- Declare class ifstream and create object fin.
- 2.Open record.txt file to read the data.
- 3.Check for error.
- 4.Using fin.seekg(0,ios::end) we traverse the get pointer to the end
-     of the file.
- 5.Fin.tellg()/sizeof(*this) is being used to count the number of records
-     in the file.
- 6.Enter record number to be searched(n).
- 7.Use seekg((n-1)*sizeof(*this)) to find the particular record.
- 8.Read the data from the file using read().

4. Modify Operation

- Declare class fstream and create the object iofile.
- Open Record.txt file to read the data in ate mode.
- Check for errors.
- iofile.seekg(0,ios::end) is being used to traverse the get pointer to the end of the file.

- iofile.tellg()/sizeof(*this) has been used to count the number of records in the file.
- The record number to be edited (n) is being entered.
- Use iofile.seekg((n-1)*sizeof(*this)) to find the particular record.
- Read the data from the file using read().
- show_data() is used to display the data.
- iofile.close() is used to close the file.
- Again, Record.txt file is opened in ate mode.
- Use iofile.seekp((n-1)*sizeof(*this)) to find the particular record.
- read_data() is again used to enter the data.
- iofile.write((char*)this,sizeof(*this)) is used to modify/update data in the file.

5. Delete Operation

- Declare class ifstream and create object fin.
- Open record.txt file using fin. open in append mode.
- Check for errors.
- fin.seekg(0, ios::end) is being used to traverse the get pointer to the end of the file.
- fin.tellg()/sizeof(*this) is being used to count the number of records in the file. Number of records is initialized to variable count.
- Enter the record number.
- Declare class fstream and create object tmpfile.
- Open Tmpfile.txt file using tmpfile.open.
- Copy all the data present in the record.txt file in the Tmpfile.txt file leaving the one that we intend to delete.
- Close both record.txt and Tmpfile.txt file.
- Remove record.txt file and rename Tmpfile.txt file as record.txt file.

FILES REQUIRED WITH ATTRIBUTES :

Name : Record
Identifier : .txt
Type of file : Text file
Location of file : Generated at runtime and is present in the same folder as the program.
Size : Can be calculated once execution is complete.
Protection : Provided as a public file (can be viewed by all).

CODE IMPLEMENTATION :

```
#include<iostream>

#include<fstream>

#include<stdlib.h>

#include<stdio.h>

using namespace std;


class Account_Intro

{

        public:

      Account_Intro()
                {

                        cout<<endl<<"\t BANK ACCOUNT";

                cout<<endl<<endl<<"\t MANAGEMENT";

                cout<<endl<<endl<<"\t SYSTEM";

                cout<<endl<<endl<<"PROJECT MADE BY :- "<<endl<<"Archit

Sarkar"<<endl<<"Aayushi Singh";

                cout<<endl<<"Pranay Kumar"<<endl<<"Pritom Mukherjee"<<endl;

                cout<<endl<<endl<<"COLLEGE : KIIT

University"<<endl<<endl<<"STREAM : Electronics and Computer Science

Engineering";

                cin.get();

                }

                 void instruction()

                 {

                 cout<<endl<<"Select one option below:"<<endl;

        cout<<"\n\t1-->Add Record to file";

        cout<<"\n\t2-->Show Record from file";

        cout<<"\n\t3-->Search Record from file";
```

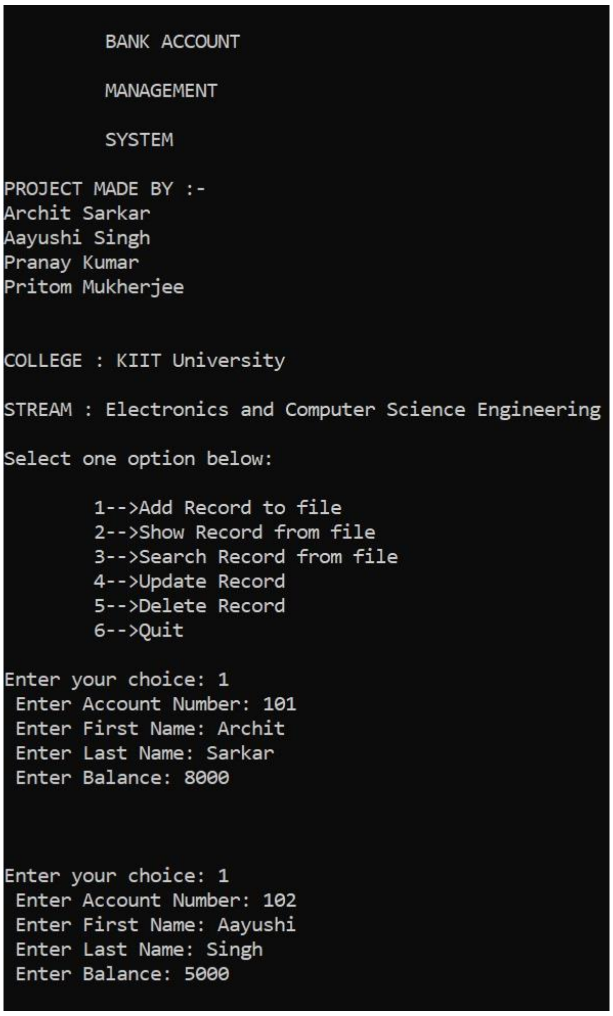
```
            cout<<"\n\t4-->Update Record";
            cout<<"\n\t5-->Delete  Record";
            cout<<"\n\t6-->Quit";
                        }
};

class Account_Data
{
protected:
            char account_number[20];
    char firstName[10];
    char lastName[10];
    float total_Balance;
public:
            void read_data()
            {
            cout<<" Enter Account Number: ";
            cin>>account_number;
            cout<<" Enter First Name: ";
            cin>>firstName;
            cout<<" Enter Last Name: ";
            cin>>lastName;
            cout<<" Enter Balance: ";
            cin>>total_Balance;
            cout<<endl;
            }
```

```
        void show_data()
    {
        cout<<"Account Number: "<<account_number<<endl;
        cout<<"First Name: "<<firstName<<endl;
        cout<<"Last Name: "<<lastName<<endl;
        cout<<"Current Balance: Rs. "<<total_Balance<<endl;
        cout<<"------------------------------ "<<endl;
    }
};

class Account_Rec : public Account_Data
{
public:
    void write_rec()
        {
        ofstream fout;
        fout.open("Record.txt",ios::app);
        if(!fout)
        {
                cout<<"Error"<<endl;
                }
                read_data();
                fout.write((char *)this, sizeof(*this));
        fout.close();
        }
        void read_rec()
        {
        ifstream fin;
        fin.open("Record.txt", ios::app);
        if(!fin)
        {
        cout<<" Error in Opening! File Not Found!!"<<endl;
        return;
        }
        cout<<endl<<"**Data from file**"<<endl;

        while(!fin.eof())
        {
                if(fin.read((char*)(this), sizeof(*this))>0)
        {
                show_data();
        }
        }
        fin.close();
        }
};

class Account_Search : public Account_Data
{
public:
        void search_rec()
    {
        int n,count;
        ifstream fin;
```

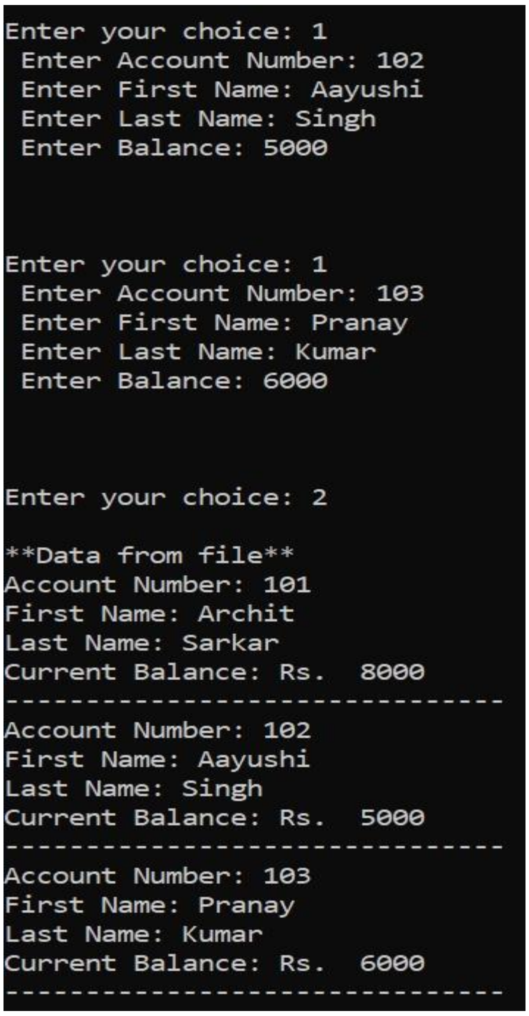
```
            fin.open("Record.txt", ios::app);
            if(!fin)
            {
            cout<<endl<<" Error in opening! File Not Found!! "<<endl;
            return;
            }
            fin.seekg(0,ios::end);
            count = fin.tellg()/sizeof(*this);
            cout<<endl<<" There are "<<count<<" records in the file"<<endl;
            cout<<" Enter Record Number to Search: ";
            cin>>n;
            fin.seekg((n-1)*sizeof(*this));
            fin.read((char *)this, sizeof(*this));
            show_data();
            }
};

class Account_Update: public Account_Data
{
public:
   void modify_rec()
   {
            int n;
            fstream iofile;
            iofile.open("Record.txt", ios::in|ios::ate);
            if(!iofile)
            {
```

```
        cout<<"\nError in opening! File Not Found!!"<<endl;
        return;
        }
        iofile.seekg(0, ios::end);
        int count = iofile.tellg()/sizeof(*this);
        cout<<"\n There are "<<count<<" records in the file"<<endl;
        cout<<"\n Enter Record Number to Modify: ";
        cin>>n;
        iofile.seekg((n-1)*sizeof(*this));
        iofile.read((char *)this, sizeof(*this));
        cout<<"Record "<<n<<" has following data"<<endl;
        show_data();
        iofile.close();
        iofile.open("Record.txt", ios::out|ios::in|ios::ate);
        iofile.seekp((n-1)*sizeof(*this));
        cout<<"\nEnter data to Modify "<<endl;
        read_data();
        iofile.write((char *)this, sizeof(*this));
        }
void delete_rec()
{
        int n,count;
        ifstream fin;
    fin.open("Record.txt", ios::app);
    if(!fin)
    {
        cout<<endl<<" Error in opening! File Not Found!! "<<endl;
        return;
        }
        fin.seekg(0, ios::end);
      count = fin.tellg()/sizeof(*this);
        cout<<"\n There are "<<count<<" records in the file"<<endl;
                cout<<"\n Enter Record Number to Delete: ";
        cin>>n;

        fstream tmpfile;
        tmpfile.open("Tmpfile.txt", ios::out|ios::app);
        fin.seekg(0);
        for(int i=0; i<count; i++)
        {
        fin.read((char *)this, sizeof(*this));
        if(i==(n-1))
                continue;
        tmpfile.write((char *)this, sizeof(*this));
        }
        fin.close();
        tmpfile.close();
        remove("Record.txt");
        rename("Tmpfile.txt", "Record.txt");
        }
};

int main()
{
        Account_Intro I;

        I.instruction();

Account_Data D;

Account_Rec R;

Account_Search S;

Account_Update U;

int choice;


while(true)

{

    cout<<endl<<endl<<"Enter your choice: ";

    cin>>choice;

    switch(choice)

    {

    case 1:

        R.write_rec();

        break;

    case 2:

        R.read_rec();

        break;

    case 3:

        S.search_rec();

        break;

    case 4:

        U.modify_rec();

        break;

    case 5:
            U.delete_rec();
            break;
        case 6:
            exit(0);
            break;
        default:
            cout<<endl<<"Enter the correct choice";
            exit(0);
        }
    }
    return 0;
}
```

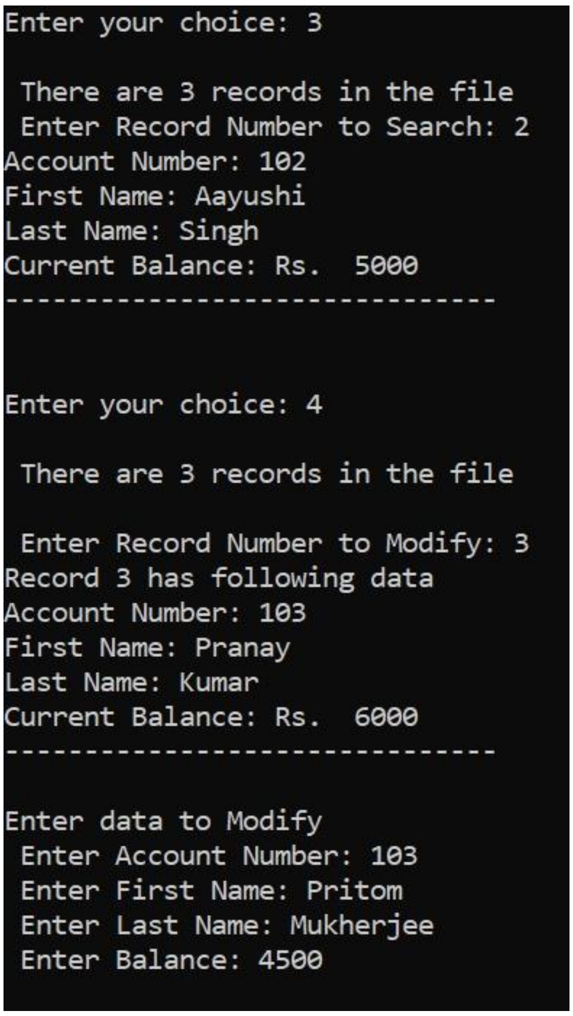
SCREENSHOTS OF OUTPUTS :



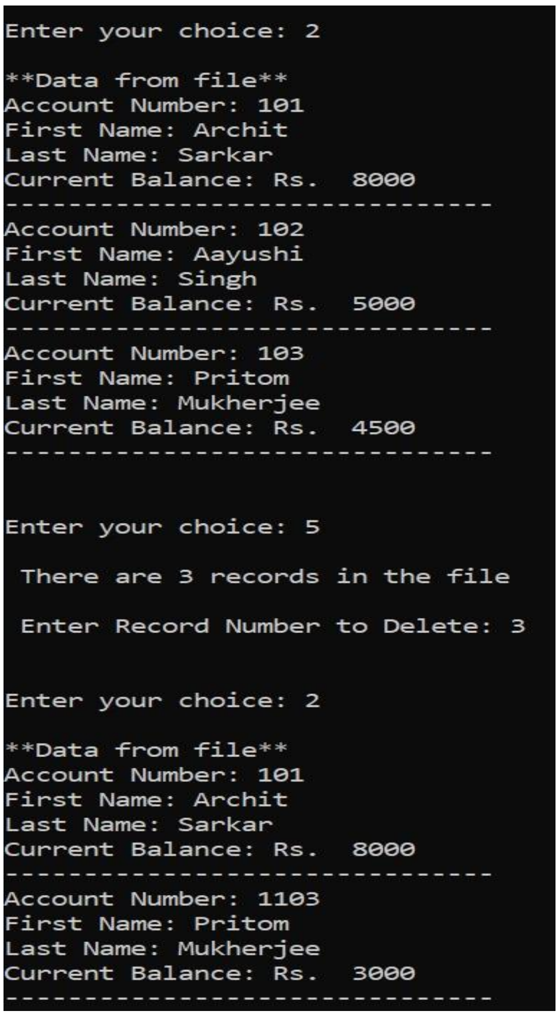
```
         BANK ACCOUNT

         MANAGEMENT

         SYSTEM

PROJECT MADE BY :-
Archit Sarkar
Aayushi Singh
Pranay Kumar
Pritom Mukherjee


COLLEGE : KIIT University

STREAM : Electronics and Computer Science Engineering

Select one option below:

        1-->Add Record to file
        2-->Show Record from file
        3-->Search Record from file
        4-->Update Record
        5-->Delete Record
        6-->Quit

Enter your choice: 1
 Enter Account Number: 101
 Enter First Name: Archit
 Enter Last Name: Sarkar
 Enter Balance: 8000


Enter your choice: 1
 Enter Account Number: 102
 Enter First Name: Aayushi
 Enter Last Name: Singh
 Enter Balance: 5000
```



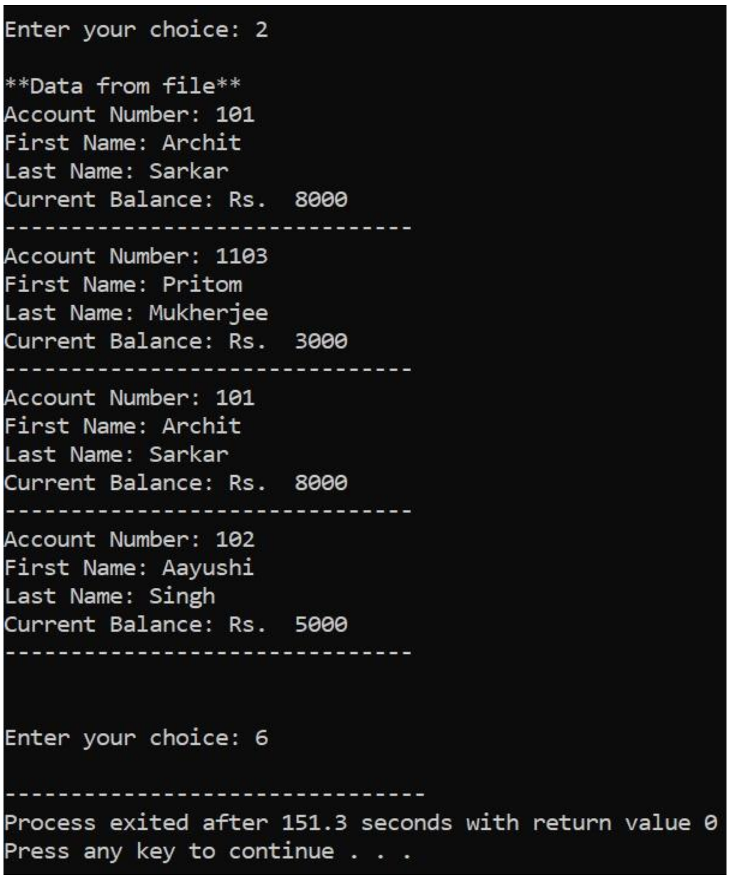
```
Enter your choice: 1
 Enter Account Number: 102
 Enter First Name: Aayushi
 Enter Last Name: Singh
 Enter Balance: 5000



Enter your choice: 1
 Enter Account Number: 103
 Enter First Name: Pranay
 Enter Last Name: Kumar
 Enter Balance: 6000



Enter your choice: 2

**Data from file**
Account Number: 101
First Name: Archit
Last Name: Sarkar
Current Balance: Rs.  8000
-------------------------------
Account Number: 102
First Name: Aayushi
Last Name: Singh
Current Balance: Rs.  5000
-------------------------------
Account Number: 103
First Name: Pranay
Last Name: Kumar
Current Balance: Rs.  6000
-------------------------------
```



```
Enter your choice: 3

 There are 3 records in the file
 Enter Record Number to Search: 2
Account Number: 102
First Name: Aayushi
Last Name: Singh
Current Balance: Rs.  5000
-------------------------------


Enter your choice: 4

 There are 3 records in the file

 Enter Record Number to Modify: 3
Record 3 has following data
Account Number: 103
First Name: Pranay
Last Name: Kumar
Current Balance: Rs.  6000
-------------------------------

Enter data to Modify
 Enter Account Number: 103
 Enter First Name: Pritom
 Enter Last Name: Mukherjee
 Enter Balance: 4500
```



```
Enter your choice: 2

**Data from file**
Account Number: 101
First Name: Archit
Last Name: Sarkar
Current Balance: Rs.  8000
-------------------------------
Account Number: 102
First Name: Aayushi
Last Name: Singh
Current Balance: Rs.  5000
-------------------------------
Account Number: 103
First Name: Pritom
Last Name: Mukherjee
Current Balance: Rs.  4500
-------------------------------


Enter your choice: 5

 There are 3 records in the file

 Enter Record Number to Delete: 3


Enter your choice: 2

**Data from file**
Account Number: 101
First Name: Archit
Last Name: Sarkar
Current Balance: Rs.  8000
-------------------------------
Account Number: 1103
First Name: Pritom
Last Name: Mukherjee
Current Balance: Rs.  3000
-------------------------------
```



```
Enter your choice: 2

**Data from file**
Account Number: 101
First Name: Archit
Last Name: Sarkar
Current Balance: Rs.  8000
-------------------------------
Account Number: 1103
First Name: Pritom
Last Name: Mukherjee
Current Balance: Rs.  3000
-------------------------------
Account Number: 101
First Name: Archit
Last Name: Sarkar
Current Balance: Rs.  8000
-------------------------------
Account Number: 102
First Name: Aayushi
Last Name: Singh
Current Balance: Rs.  5000
-------------------------------


Enter your choice: 6

--------------------------------
Process exited after 151.3 seconds with return value 0
Press any key to continue . . .
```

TEST CASES :

Test case 1 (Input data/show data):

```
Enter your choice: 1
Enter Account number : 101
Enter First Name : Archit
Enter Last name : Sarkar
Enter Balance : 8000

Enter your choice: 2
```

Output :

```
**Data from file**
Account number : 101
First Name : Archit
Last name : Sarkar
Balance : Rs. 8000
```

Test case 2 (Search data): // record.txt contains 3 account details.

```
Enter your choice: 3

There are 3 records in the file
```

```
Enter Record Number to Search: 2
```
Output :

```
**Data from file**
Account number : 101
First Name : Archit
Last name : Sarkar
Balance : Rs. 8000
```

Test case 3(Update record): // record.txt contains 3 account details.

```
Enter your choice: 4

There are 3 records in the file

Enter Record Number to edit: 2
Record 2 has following data
Account number : 101
First Name : Archit
Last name : Sarkar
Current Balance : Rs. 8000
- - - - - - - - - - - - - - - - - - -

Enter data to Modify
Enter Account Number: 123
Enter First Name : Pritom
Enter Last name : Mukherjee
Enter Balance : 9000

Enter your choice: 2
```

Output :

```
**Data from the file**
// Data 1

Account Number: 123
First Name : Pritom
Last name : Mukherjee
Current Balance : Rs. 9000

// Data 3
```

Test case 4(Delete Record):

```
Enter your choice: 5
There are 3 records in the file
Enter Record Number to Delete: 2
Enter your choice:2
```

Output:

```
**Data from the file**
//Data 1
//Data 3 as Data 2
```

FUTURE SCOPE :

Our present system allows employees to manage , retrieve and modify the customer account easily.
We can expand our program so that the customers can perform the basic banking transactions by sitting at their office or at homes through PC or laptop. The system may provide the access to the customer to perform the transactions on account as per their requirements. We can switch from file system to DBMS for more efficiency.

CONCLUSION :

This project is developed to nurture the needs of an employee in a banking sector by embedding all the tasks of maintaining an account in a bank. Future versions of this project will still be much enhanced than the current version. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable in a volatile and competitive marketplace of today.

Thus, the Bank Account Management System is developed and executed successfully.

REFERENCES :

1 **https://www.researchgate.net/publication/301293322_Bank_Account_Management_System**

2 **Class Notes**