



# **Bank Management System**

Project Done by:  
*Aneez Ahmed Jaheez*  
*Srivatsan T.V*



# **Acknowledgement**



The acknowledgement is the most beautiful page of any book.

I, Aneez Ahmed Jaheezuddin of class XII, find words quite inadequate to express my gratitude to many people. First and foremost, I would like to thank the Almighty. I would also like to express my gratitude to the Principal of my school, Mrs. Stella Punitha and the Campus Head, Mrs. Nancy James for their constant encouragement and support.

I would like to specially thank my computer science teacher, Mrs. Lovely Binesh, for making my practical sessions both enjoyable and informative which provoked my inquisitiveness and curiosity throughout the academic year 2018-19.

I would also like to thank my parents and friends for their continued help and support, without which I would not have been able to complete this project with such success.

Thank you very much.



# INDEX

S.NO	TOPIC	PAGE NO.
1.	Aim	3
2.	Synopsis	3
3.	Advantages of the Proposed System	4
4.	Uses of the Proposed System	4
5.	Why this project?	4
6.	System requirements	5
7.	System Flow Diagram	6
8.	Header Files and User-Defined Data	9
9.	Algorithm	12
10.	Source Code	16
11.	Screenshots	30
12.	Scope of Improvement	35
13.	Bibliography	36

## **Aim:**

The aim of the Bank Management System software is design a system through which a bank administrator can manage customer information , stock details, purchase details, etc.

## **Synopsis:**

The objective of this study is to make the customers of various banks do their account accessibility and transactions using the solution proposed in the following statement. Customers need not interact with various applications or websites of each bank. The admin will have full control over bank details and can update the existing details of the customer and the bank. The admin will handle the registration of a customer to use this application. The bank admin can access the application to see all customer account details and he/she can accept or reject the fund transfer of the customer. The admin should be able to provide response for the queries the customer puts forward. The customer should also be able to view the account related details through the admin. The customer should also be able to send queries to the bank admin.

Every day banks need to perform many activities which requires developed infrastructure and more staff members. But the online banking system allows the banks to perform these activities in a simpler way without involving the employees, for instance, online banking, mobile banking and ATM banking. Additionally the banking needs more security. Moreover, all the transactions are manual work, so this results in time delays. In this system, the user can easily perform the money transaction at any time, and from any place. Each and every transaction is sent to the user via message. There are two main modules called admin and user modules. In the admin module, the admin can maintain the user details and modify the details. In the user module, the user can perform the money transaction online. They can view their account details using balance enquiries.

### **Advantages of the Proposed System:**

1. It provides connectivity between banks.
2. Provides a single account from all banks.
3. 24/7 transactions through the application.
4. Efficient use of assets.

### **Uses of the Proposed System:**

The system in discussion can introduce a multi-bank system. This application will enable users to save more time and use available features in every bank. Transactions and updating are maintained by the admin and provide customer support for the users. The application will act as a mediator between banks and users. Users can maintain a single username and password.

### **Why this project?**

This project centralizes the user database management for the bank system and allows the administrator to oversee all entries and modifications. The user is aware of all changes being made to their accounts but only the administrator has direct access to the manipulation and modification of data on the database. The administrator can also prepare a summary of all the accounts stored in the database with all the necessary information included. Reduction of complexity is a key factor in making a system more user-friendly and this is exactly what the proposed system manages to achieve.

## **System Requirements:**

### **Hardware and Software Requirements:**

#### Computer:

- Pentium(R) Dual-Core CPU
- E5800 @ 3.20 GHz
- 3.20 GHz, 992 MB of RAM
- Physical Address Extension
- Hard Disk = 300 GB

#### Mouse:

- Optical

#### Keyboard:

- Standard PS/2

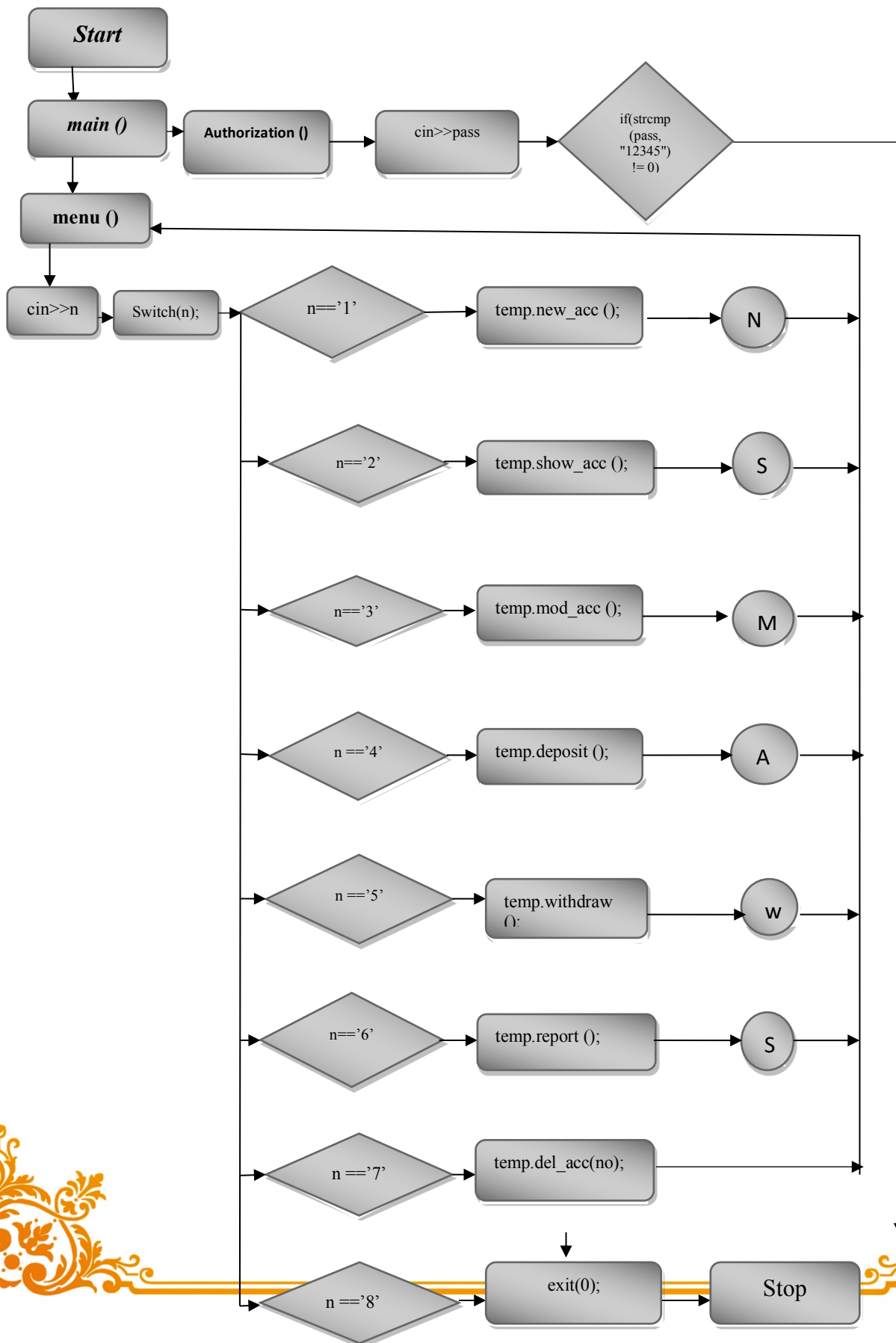
#### Monitor:

- 1366 by 768 Pixels

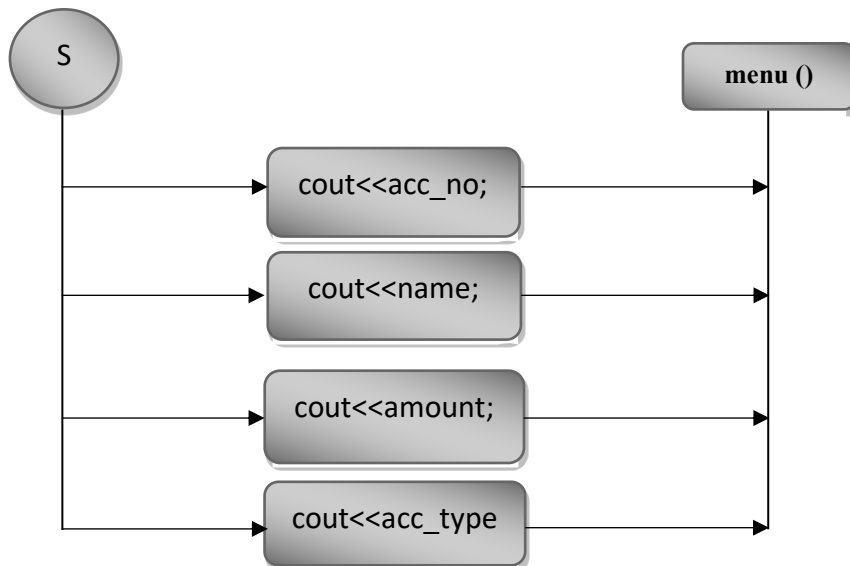
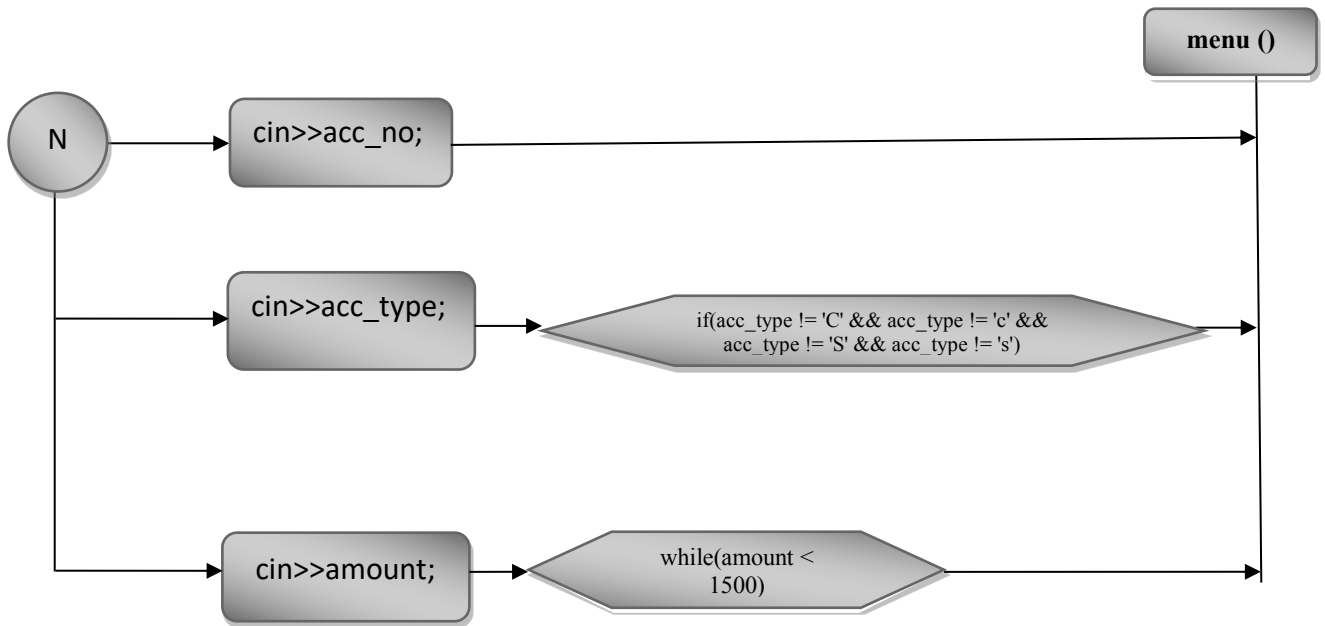
#### Operating System:

- Microsoft Windows XP Professional Version 2002 Service Pack 2
- TC Version = 3.1/4

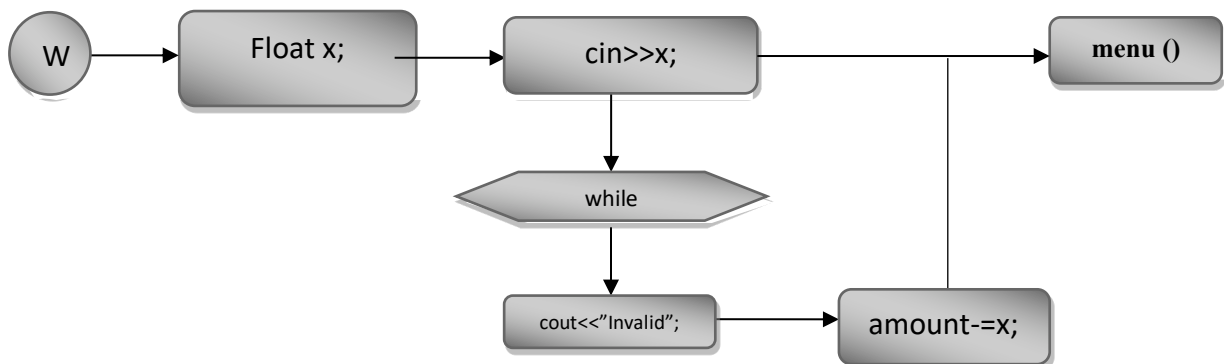
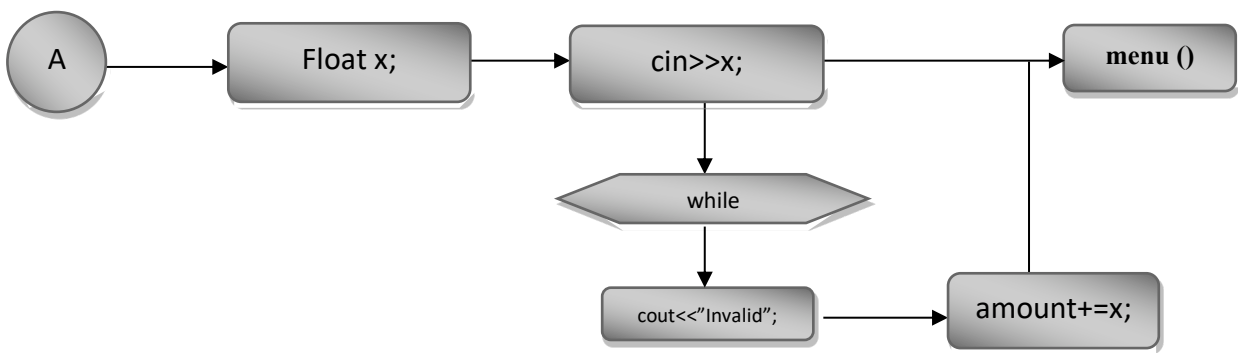
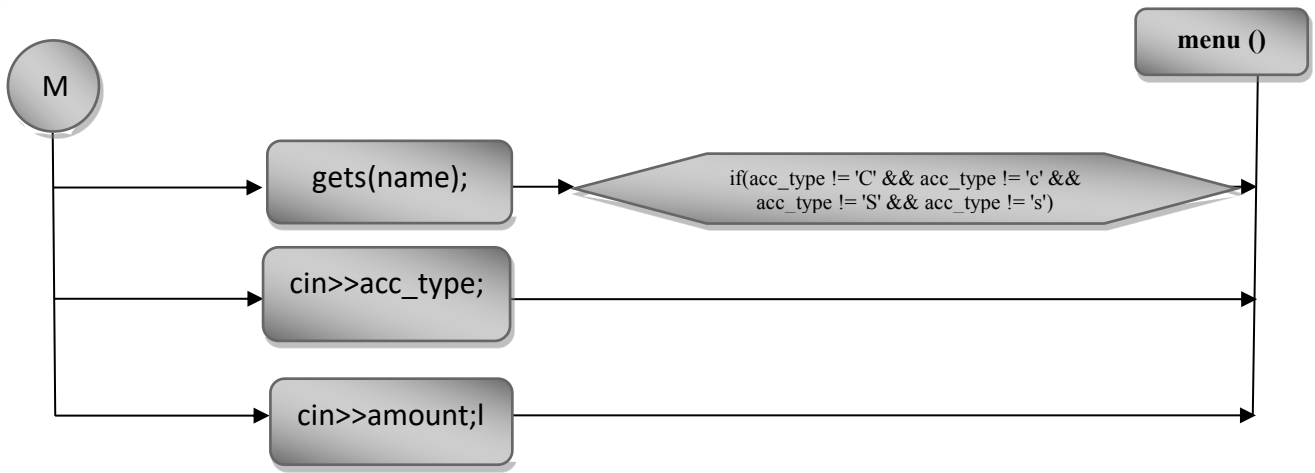
## System Flow Diagram:











## **Header Files and User-Defined Data:**

### **Library Functions:**

<b>S.NO</b>	<b>Header file</b>	<b>Purpose</b>
1.	iostream	This headerfile has been included to enable the input and output stream operations.
2.	string.h	This headefile is used for the strcmp() function.
3.	ctype.h	This headerfile is used for the toupper() function.
4.	iomanip.h	This headerfile has been included to use the setw() function.
5.	fstream.h	This headerfile has been included to enable file handling.
6.	process.h	This headerfile has been included to use the exit() function.
7.	conio.h	This headerfile is used for the functions getch() and clrscr().
8.	stdio.h	This headerfile has been included in order to use the gets(), remove and rename functions.

### **User-defined functions:**

<b>S.NO</b>	<b>Data type</b>	<b>Functions</b>	<b>Purpose</b>
1	void	new_acc()	To create a new account
2	void	show_acc()	To display the account details

3	void	mod_acc()	To modify the details of a particular account
4	void	deposit()	To deposit into the user account and make changes accordingly
5	void	withdraw()	To withdraw from the bank account and make changes accordingly
6	void	Report()	To generate a report of the user's account and display the data
7	int	ret_accno()	To return private data members 'account number'

**Class name: Bank\_Sys**

S.NO	Data type	Members	Purpose
	<b>Private:</b>	<b>Data members:</b>	
1	int	acc_no	User account number
2	char	name[30]	User account name
3	char	acc_type	User account type (credit/savings)
4	float	amount	User bank balance

5	<b>Public:</b> void	<b>Member functions:</b> new_acc()	To create a new account
6	void	show_acc()	To display the account details
7	void	mod_acc()	To modify the details of a particular account
8	void	deposit()	To deposit into the user account and make changes accordingly
9	void	withdraw()	To withdraw from the bank account and make changes accordingly
10	void	Report()	To generate a report of the user's account and display the data
11	int	ret_accno()	To return private data members 'account number'

#### **Data Files:**

<b>S.NO</b>	<b>File name</b>	<b>Purpose</b>
1.	Bank.dat	To store all user account data
2.	Temp.dat	Temporary container file that is used to delete specified accounts and retain the necessary data from the database.

## **Algorithm:**

**Step 1:** Start the process.

**Step 2:** Invoke the authorization() function.

- I. Declare a character array 'pass[10]' and read the data.
- II. If pass == "12345"  
Control returns back to the main function  
Else  
A message stating that the password entered is incorrect is  
Displayed and the exit() function is invoked.

**Step 3:** Invoke the introduction() function and display the system developers or the administrator.

**Step 4:** Invoke the menu() function.

- I. Declare a character 'n', integer 'no = 0' and 'flag', double pos, and an object temp of type Bank\_Sys.

**Step 5:** Accept the data for 'n' as the user's choice.

- I. If n == '1'
  - i. Invoke the new\_acc() function of the class using 'temp' and the required account data is accepted into 'temp'.
  - ii. Ensure the account number entered by the user is not redundant by checking it against the data stored in the file.
  - iii. The data stored in temp is written into the binary file "Bank.dat" under the append file mode.
  - iv. The data file "Bank.dat" is closed.
- II. If n == '2'
  - i. Initialize 'flag = 1'.
  - ii. Open the binary file "Bank.dat" in the 'in' filemode.
  - iii. Accept the account number from the user and store it in 'no'.
  - iv. The file is read by object till the end of the file and while 'flag' is not equal to zero. The object is stored in 'temp' after every iteration.

- v. If the returned value of 'temp.ret\_accno()' is equal to 'no' then invoke the 'show\_acc()' function and initialize 'flag = 0'.
- vi. The account details are displayed to the user using the 'show\_acc()' function.
- vii. The data file "Bank.dat" is closed.

III. If n == '3'

- i. Initialize 'flag = 1'.
- ii. Accept the account number from the user and store it in 'no'.
- iii. Open the binary file "Bank.dat" in the 'in' and 'out' filemode.
- iv. The data on the file is read by object till the end of the file while 'flag' is not equal to zero. The object is stored in 'temp' after every iteration.
- v. If the value returned by 'temp.ret\_accno()' matches 'no', then invoke the 'mod\_acc()' function and allow the user to make the desired modifications.
- vi. The memory location of the object is stored in 'pos' and the pointer is moved to this position after a match has been found.
- vii. Write the object back into the binary file after the modifications are made and initialize 'flag = 0'.
- viii. The data file "Bank.dat" is closed.

IV. If n == '4'

- i. Initialize 'flag = 1'.
- ii. Accept the account number from the user and store it in 'no'.
- iii. Open the binary file "Bank.dat" in the 'in' and 'out' filemode.
- iv. The data on the file is read by object till the end of the file while 'flag' is not equal to zero. The object is stored in 'temp' after every iteration.
- v. If the value returned by 'temp.ret\_accno()' matches 'no', then invoke the 'deposit()' function and allow the user to make the desired transaction and update the account details accordingly.



- vi. The memory location of the object is stored in 'pos' and the pointer is moved to this position after a match has been found.
- vii. Write the object back into the binary file after the transaction is made and initialize 'flag = 0'.
- viii. The data file "Bank.dat" is closed.

V. If n == '5'

- i. Initialize 'flag = 1'.
- ii. Accept the account number from the user and store it in 'no'
- iii. Open the binary file "Bank.dat" in the 'in' and 'out' filemode.
- iv. The data on the file is read by object till the end of the file while 'flag' is not equal to zero. The object is stored in 'temp' after every iteration.
- v. If the value returned by 'temp.ret\_accno()' matches 'no', then invoke the 'withdraw()' function and allow the user to make the desired transaction and update the account details accordingly.
- vi. The memory location of the object is stored in 'pos' and the pointer is moved to this position after a match has been found.
- vii. Write the object back into the binary file after the transaction is made and initialize 'flag = 0'.
- viii. The data file "Bank.dat" is closed.

VI. If n == '6'

- i. Open the binary file "Bank.dat" in the 'in' and 'out' filemode.
- ii. The data on the file is read by object till the end of the file.
- iii. The object is stored in temp after every iteration.
- iv. The 'report()' function is invoked after every iteration using which the account details are displayed in the form of a report.
- v. The data file "Bank.dat" is closed.

VII. If n == '7'

- i. Invoke the 'exit()' function.
- ii. Stop the process.



- VIII. If `n == '8'`
- i. Accept the account number from the user and store it in `'no'`.
  - ii. Invoke the `'del_acc()'` function.
    - a) Initialize `'flag = 1'`.
    - b) Declare two file streams, `instream` and `outstream`, and open the binary files `"Bank.dat"` and `"Temp.dat"` in the `'in'` and `'out'` filemode respectively.
    - c) The data on the file `"Bank.dat"` is read by object. Simultaneously write the object onto the binary file `"Temp.dat"` if the value returned by `'temp.ret_accno()'` does not match the value stored in `'n'`.
    - d) Initialize `'flag = 0'` if any of the account numbers match the account number entered by the user.
    - e) Close the datafiles `"Bank.dat"` and `"temp.dat"`.
    - f) If `'flag == 0'`, delete the datafile `"Bank.dat"` using the `'remove()'` function.
    - g) If `'flag == 0'`, rename the datafile `"temp.dat"` to `"Bank.dat"` using the `'rename()'` function.
    - h) Else display a message to the administrator stating that the account to be deleted does not exist and exit the function.
- IX. If the user enters any value that does not match the values included above, display an error message and repeat step 5.

**Step 6:** Stop the process.

## **Source Code:**

```
/*bank management system to create account, display, account details, modify  
account, deposit or withdraw from account, or generate a report of a particular  
account */
```

```
/******
```

### **HEADER FILE USED IN PROJECT**

```
*****/
```

```
#include <iostream.h>  
#include <iomanip.h>  
#include <string.h>  
#include <ctype.h>  
#include <fstream.h>  
#include <conio.h>  
#include <process.h>  
#include <stdio.h>
```

```
fstream file; //data file path to facilitate write and read operations
```

```
/******
```

### **CLASS USED IN PROJECT**

```
*****/
```

```
class Bank_Sys//object that represents bank account of one particular user
```

```
{  
private:  
    int acc_no;        //user account number  
    char name[30];     //user account name  
    char acc_type;     //user account type(credit/saving)  
    float amount;      //user bank balance
```

```
public:
```

```
/******
```

### **Function declaration**

```
*****/
```

```

void new_acc();           //function prototype to create a new account
void show_acc();         //function prototype to display the account details
void mod_acc();          //function prototype to modify the details of
                           particular account

void deposit();          //function prototype to deposit into user account and
                           make changes accordingly

void withdraw();         //function prototype to withdraw from the account and
                           make changes accordingly

void report();           //function to prototype generate a report of the user's
                           account and display relevant data

void del_acc(int);       //function to delete a user's account from the database

int ret_accno()          //function to return private data member 'account
                           number'
{
    return acc_no;
}
};

void Bank_Sys :: new_acc() //function definition to create a new account
{
    Bank_Sys temp;         //temporary container class object

    file.open("Bank.dat", ios::binary | ios::in);

    if(!file)
    {
        cout <<"File does not exist." <<endl;
        exit(0);
    }

    cout <<"Enter the required account details." <<endl;
    cout <<"Account Number: "; cin>>acc_no;

    while(!file.eof())     //prevents redundancy of account numbers
    {

```

```

file.read((char*)&temp, sizeof(temp));
while(acc_no == temp.ret_accno())
{
    cout <<"Account number already in use. Please enter a new account
        number." <<endl;
    cin >>acc_no;
}
}
file.close();

cout <<"Name of Account Holder: "; gets(name);
cout <<"Type of Account ('C' for Credit/ 'S' for Savings): "; cin >>acc_type;

while(acc_type != 'C' && acc_type != 'c' && acc_type != 'S' && acc_type !=
's')
    //prevents invalid entry
{
    cin >>acc_type;

    if(acc_type != 'C' && acc_type != 'c' && acc_type != 'S' && acc_type !=
's')
    {
        cout <<"Invalid entry. Re-enter account type." <<endl;
    }
}

acc_type = toupper(acc_type);
cout <<"Enter the initial amount (Minimum Rs. 1500): "; cin >>amount;

while(amount < 1500)
{
    cout <<"The initial account balance is too low. Please re-enter the starting
        amount." <<endl;
    cin >>amount;
}

cout <<"The account has been created successfully." <<endl;
getch();
}

void Bank_Sys :: show_acc() //function definition to show the account details
of a particular account

```

```

{
    cout <<"The details of the desired account are given below." <<endl
    <<"Account Number: " <<acc_no <<endl
    <<"Name of Account Holder: " <<name <<endl
    <<"Type of account: " <<acc_type <<endl
    <<"Balance: " <<amount <<endl;

    getch();
}

void Bank_Sys :: mod_acc()    //function definition to modify the details of a
                             particular account
{
    cout <<"Please make the desired changes to the account details." <<endl;
    cout <<"Name of Account Holder: "; gets(name);
    cout <<"Type of Account ('C' for Credit/ 'S' for Savings): "; cin >>acc_type;

    while(acc_type != 'C' && acc_type != 'c' && acc_type != 'S' && acc_type !=
's')    //prevents invalid entry
    {
        cin >>acc_type;

        if(acc_type != 'C' && acc_type != 'c' && acc_type != 'S' && acc_type !=
's')
        {
            cout <<"Invalid entry. Re-enter account type." <<endl;
        }
    }
    acc_type = toupper(acc_type);

    cout <<"Enter the initial amount: "; cin >>amount;
    cout <<"The changes have successfully been made." <<endl;

    getch();
}

void Bank_Sys :: deposit()    //function definition to deposit into a particular
                             account
{

```

```

float x = 0;//variable to store the amount of money to be deposited
cout <<"Enter the amount you would like to deposit into this account.
      (Cannot be more than Rs. 20000 at a time)" <<endl;
cin >>x;

while(x > 20000)//ensures that more than Rs.20000 is not deposited in one
transaction
{
    cout <<"Invalid amount. Please enter an amount less than 20000." <<endl;
    cin >>x;
}

amount += x;
cout <<"The account balance has been updated successfully." <<endl;

getch();
}

void Bank_Sys :: withdraw()    //function definition to withdraw from a certain
                                account
{
    float x;                    //variable to store the amount of money to be
                                withdrawn

    cout <<"Enter the amount you would like to withdraw from your account.
          (Must be less than 10000 at a time)" <<endl;
    cin >> x;

    while(x > 10000)//ensures that more than Rs. 10000 is not withdrawn in a
single transaction
    {
        cout <<"Invalid amount. Please withdraw less than 10000 at a time."
        <<endl;
        cin >>x;
    }

    amount -= x;
    cout <<"The account balance has been updated successully." <<endl;

```



```

    getch();
}

void Bank_Sys :: report()           //function definition to generate a report for a
                                    particular account
{
    cout <<endl;
    cout <<acc_no <<setw(10)<<" " <<name <<setw(10) <<" " <<acc_type
        <<setw(6) <<amount <<endl;

    getch();
}

void Bank_Sys :: del_acc(int n) //function to delete the desired account from the
                                database
{
    int flag = 1;
    Bank_Sys temp;//temporary container variable
    ifstream infile;
    ofstream outfile;

    infile.open("Bank.dat", ios::binary | ios::beg);
    outfile.open("temp.dat", ios::binary);

    if(!infile || !outfile)
    {
        cout <<"The file does not exist." <<endl;
        exit(0);
    }

    while(!infile.eof())           //fucnton to transfer all the required data to a new file
                                    and delete the old file which contains all the
                                    unnecessary data
    {
        infile.read((char*)&temp, sizeof(temp));
        if(temp.ret_accno() != n)
        {
            outfile.write((char*)&temp, sizeof(temp));
        }else
    }
}

```



```
{
    flag = 0;
}

infile.close();
outfile.close();

if(flag == 0)           //renames the new file with the name of the old file in
                        case the condition is fulfilled
{
    remove("Bank.dat");
    rename("temp.dat", "Bank.dat");
    cout <<"The account has been deleted successfully." <<endl;
} else
{
    cout <<"Account does not exist." <<endl;
}

getch();
}

void menu()             //menu function to help the user traverse the bank
                        system and perform the desired operation using the
                        system.
{

    clrscr();
    char n;              //the data stored in this variable decides the course of
                        the program by satisfying one of the eight given
                        options

    int no = 0, flag;     //n takes in user input to determine the desired
                        operation, no holds the account number to check
                        against the bank records during search operations

    double pos;          //posistion is used to determine and place the pointer
                        in a particular memory location

    Bank_Sys temp;       //temporary container object to faciitate storage of
                        data on the binary file system
```

```
do
{
```

```
    cout <<endl <<"Enter the desired operation." <<endl
    <<"01. Create a new account." <<endl
    <<"02. Display the details of an account." <<endl
    <<"03. Modify the details of an account." <<endl
    <<"04. Deposit into account." <<endl
    <<"05. Withdraw from account." <<endl
    <<"06. View account report." <<endl
    <<"07. Exit System." <<endl
    <<"08. Delete an account." <<endl;
```

```
    cin >>n;
```

```
    switch (n)
    {
```

```
        //case 1 calls the function to create a new account and
        stores the same on the binary file
```

```
        case '1':
```

```
            clrscr();
```

```
            temp.new_acc();
```

```
            file.open("Bank.dat", ios::binary | ios::app);
```

```
            if(!file)
```

```
            {
```

```
                cout <<"File does not exist." <<endl;
```

```
                exit(0);
```

```
            }
```

```
            file.write((char*) &temp, sizeof(Bank_Sys));
```

```
            file.close();
```

```
            break;
```

```
        //case 2 calls the function to display the account
        details by obtaining the desired data from the binary
        file in which it is stored
```

```
        case '2':
```

```
            flag = 1;
```

```

file.open("Bank.dat", ios::in | ios::binary);
if(!file)
{
    cout <<"File does not exist." <<endl;
    exit(0);
}

cout <<"Enter the Account Number." <<endl;
cin >>no;

while((!file.eof()) && flag != 0)
{
    pos = file.tellg();
    file.read((char*) &temp, sizeof(Bank_Sys));
    if(temp.ret_accno() == no)
    {
        clrscr();
        temp.show_acc();
        flag = 0;
    }
}
file.close();
if(flag == 1)
    cout <<"Account does not exist. Please enter a valid account
number." <<endl;

break;

//case 3 calls the function to modify the account details and makes
the desired changes in the binary
file holding the data
case '3':
    flag = 1;

    cout <<"Enter the Account Number." <<endl;
    cin >>no;

    file.open("Bank.dat",ios::binary|ios::in|ios::out);
    if(!file)
    {
        cout <<"File does not exist." <<endl;
    }

```

```

        exit(0);
    }

    while(!file.eof() && flag == 1)
    {
        pos = file.tellg();
        file.read((char*)&temp, sizeof(temp));
        if(temp.ret_accno()==no)
        {
            clrscr();
            temp.mod_acc();
            file.seekp(pos);
            file.write((char*) (&temp), sizeof(temp));
            cout<<"The record has been updated successfully." <<endl;
            flag = 0;
        }
    }
    file.close();
    if(flag == 1)
        cout <<"Account does not exist. Please enter a valid account
number." <<endl;
    break;

    //case 4 calls the function that allows the user to deposit the desired
    amount and makes the changes in the account details in the binary file
    case '4':
        flag = 1;

        file.open("Bank.dat", ios::in | ios::out | ios::binary);
        if(!file)
        {
            cout <<"File does not exist." <<endl;
            exit(0);
        }

        cout <<"Enter the Account Number." <<endl;
        cin >>no;

        while(!file.eof() && flag!= 0)
        {
            pos = file.tellg();

```

```

file.read((char*) &temp, sizeof(Bank_Sys));
if(temp.ret_accno() == no)
{
    clrscr();
    temp.deposit();
    flag = 0;
    file.seekg(pos);
    file.write((char*) &temp, sizeof(Bank_Sys));
}
file.close();
}
if(flag == 1)
    cout <<"Account does not exist. Please enter a valid account
number." <<endl;

break;

//case 5 calls the function that allows the user to withdraw from the
account and updates the balance in the binary file
case '5':
    flag = 1;

    file.open("Bank.dat", ios::in | ios::out | ios::binary);

    if(!file)
    {
        cout <<"File does not exist." <<endl;
        exit(0);
    }

    cout<<"Enter the Account Number." <<endl;
    cin >>no;

    while(!file.eof() && flag!= 0)
    {
        pos = file.tellg();
        file.read((char*) &temp, sizeof(Bank_Sys));
        if(temp.ret_accno() == no)
        {
            clrscr();
            temp.withdraw();

```

```

        flag = 0;
        file.seekg(pos);
        file.write((char*) &temp, sizeof(Bank_Sys));
    }
    file.close();
}
if(flag == 1)
    cout <<"Account does not exist. Please enter a valid account
number." <<endl;

    break;

    //case 6 obtains the details of the desired account from the binary
    File and generates a report based on the account details and
    displays the tabulated details in the form of a report
case '6':
    file.open("Bank.dat", ios::in | ios::binary);

    if(!file)
    {
        cout <<"File does not exist." <<endl;
        exit(0);
    }

    cout<<"The account holder report is given below." <<endl;

    cout<<"=====
    =\n";

    cout<<"A/c no.    NAME        Type  Balance\n";

    cout<<"=====
    =\n";

    while(file.read((char*) &temp, sizeof(Bank_Sys)))
    {
        clrscr ();
        temp.report();
    }
    file.close();
    break;

case '7':

```

```

        exit(0); //invocation of this case ends the program

    case '8':    //case 8 allows the administrator to delete an account from
                 the database
        cout <<"Enter the account number of the account you would like to
                 delete." <<endl;
        cin >>no;

        clrscr();
        temp.del_acc(no);
        break;

    default:    //default case that displays an error message whenever the
                 user tries to follow a path that is not present
        cout <<"Invalid entry." <<endl;
        break;
    }
}while(n!=7);
}

void introduction()//function to introduce the developers or administrators of the
                    bank management system
{
    cout<<"\n\n\n\t BANK";
    cout<<"\n\n\tMANAGEMENT";
    cout<<"\n\n\t SYSTEM";
    cout<<"\n\n\n\nMADE BY : Aneez Jaheez and Srivatsan T.V\n\n\n\n";
}

void authorization()//funtion to ensure only the administrator is able to gain
                    direct access to the database
{
    char pass[10];

    cout <<"Password: "; cin >>pass;

    if(strcmp (pass, "12345") != 0)
    {
        cout <<"Incorrect Password." <<endl;
        exit(0);
    }
}

```



```
clrscr();
}

//*****

//  THE MAIN FUNCTION OF PROGRAM
//*****

void main() //main funtion; program execution begins here
{
    clrscr();

    authorization();


    introduction();

    menu();

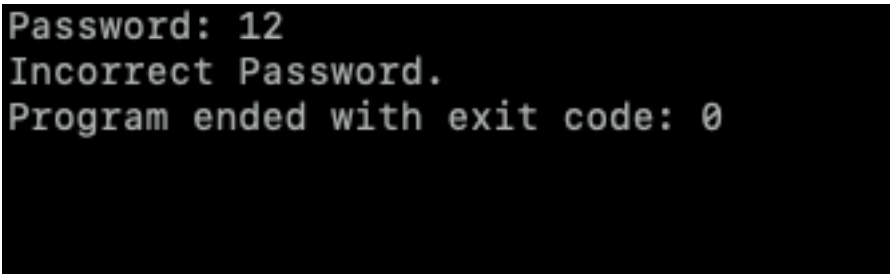
    getch();
} //end of the program
```

## Screenshots:

### Front screen:



```
Password:
```



```
Password: 12  
Incorrect Password.  
Program ended with exit code: 0
```

## Menu Screen:

Password: 12345

BANK  
MANAGEMENT  
SYSTEM

MADE BY : Aneez Jaheez and Srivatsan T.V

Enter the desired operation.  
01. Create a new account.  
02. Display the details of an account.  
03. Modify the details of an account.  
04. Deposit into account.  
05. Withdraw from account.  
06. View account report.  
07. Exit System.  
08. Delete an account.

Enter the desired operation.  
01. Create a new account.  
02. Display the details of an account.  
03. Modify the details of an account.  
04. Deposit into account.  
05. Withdraw from account.  
06. View account report.  
07. Exit System.  
08. Delete an account.  
9  
Invalid entry.

Enter the desired operation.  
01. Create a new account.  
02. Display the details of an account.  
03. Modify the details of an account.  
04. Deposit into account.  
05. Withdraw from account.  
06. View account report.  
07. Exit System.  
08. Delete an account.

### Account Creation:

```
Enter the desired operation.  
01. Create a new account.  
02. Display the details of an account.  
03. Modify the details of an account.  
04. Deposit into account.  
05. Withdraw from account.  
06. View account report.  
07. Exit System.  
08. Delete an account.
```

```
1  
Enter the required account details.  
Account Number: 1  
Account number already in use. Please enter a new account number.
```

```
1  
Enter the required account details.  
Account Number: 1  
Name of Account Holder: Aneez  
Type of Account ('C' for Credit/ 'S' for Savings): T  
t  
Invalid entry. Re-enter account type.  
C  
Enter the initial amount (Minimum Rs. 1500): 1000  
The initial account balance is too low. Please re-enter the starting amount.  
3000  
The account has been created successfully.
```

### Display Details:

```
2  
Enter the Account Number.  
1  
The details of the desired account are given below.  
Account Number: 1  
Name of Account Holder: Aneez  
Type of account: C  
Balance: 3000
```

### Modify Existing Account:

```
3
Enter the Account Number.
1
Please make the desired changes to the account details.
Name of Account Holder: Aneez
Type of Account ('C' for Credit/ 'S' for Savings): S
Enter the initial amount: 2000
The changes have successfully been made.
The record has been updated successfully.
```

### Deposit:

```
4
Enter the Account Number.
1
Enter the amount you would like to deposit into this account. (Cannot be more than Rs. 20000 at a time)
25000
Invalid amount. Please enter an amount less than 20000.
19000
The account balance has been updated successfully.
```

### Withdraw:

```
5
Enter the Account Number.
1
Enter the amount you would like to withdraw from your account. (Must be less than 10000 at a time)
11000
Invalid amount. Please withdraw less than 10000 at a time.
5000
The account balance has been updated successfully.
```

### View Account Report:

```
6
The account holder report is given below.
=====
A/c no.      NAME                Type  Balance
=====
1            Aneez                  C  17000
20           Shahid                  S4500.5
7            Erika                   C   6700
9            Tamara                   S   4000
```

### Delete account:

```
8
Enter the account number of the account you would like to delete.
1
The account has been deleted successfully.
```

```
6
The account holder report is given below.
=====
A/c no.      NAME                Type  Balance
=====
20           Shahid                  S4500.5
7            Erika                   C   6700
9            Tamara                   S   4000
```

```
8
Enter the account number of the account you would like to delete.
29
Account does not exist.
```

### **End Session:**

```
Enter the desired operation.  
01. Create a new account.  
02. Display the details of an account.  
03. Modify the details of an account.  
04. Deposit into account.  
05. Withdraw from account.  
06. View account report.  
07. Exit System.  
08. Delete an account.  
7  
Program ended with exit code: 0|
```

### **Scope of Improvement:**

1. It does not provide transactions from one bank to another bank.
2. Separate accounts for each bank.
3. Transactions only during bank timings.
4. Time consumption for performing and recording transactions due to the number of people and processes involved.



## **Bibliography:**

- <https://www.ijariit.com/manuscripts/v3i1/V3I1-1233.pdf?04d616&04d616> – An insight into multi-banking systems.
- <http://www.fsb.org/wp-content/uploads/P011117.pdf> - The use of computer science in financial services.
- <http://www.continuitysoftware.com/wp-content/uploads/2013/07/Banking-Case-Study.pdf> - A case study on banking services.
- Computer Science with C++ for Class XII - Sumitha Arora (11th Edition)