

《Data Structures》

Course Project Report

Topic: BANK MANAGEMENT SYSTEM

Class: 201c - SOFTWARE ENGINEERING

Name: RUDRA MD IFTEKHARUL ISLAM

Student Number: 2011562107

Contents

1. Problem definition	3 - 4
1.1 Description	3
1.2 Features	4
2. Outline design	4 - 9
2.1 corresponding data structures	5 - 8
2.2 screenshots of important codes (functions)	5 – 7
2.3 main function	8 - 9
3. Coding and test analysis	10 - 29
3.1 code description	10
3.2 diagrams (E-R, Use-case, DFD)	10 – 13
3.3 output screenshots of my compiler and analysis	14 – 20
3.4 name of tools	20
3.5 source code	21 - 29
4. Conclusion	29 -30

1. Problem definition

The Bank Management System (BMS) is a web-based application used for paying financial institutions for the services they provide to the Bureau of the Fiscal Service. BMS also provides analytical tools to review, and approve compensation, budgets, and outflows.

The main objective of the project is to develop online Banking system for banks. In present system all banking work is done manually. User have to visit bank to Withdrawal or Deposit amount. In present bank system it is also difficult to find account information of account holder. In this bank management system we will automate all the banking process. In our bank management system user can check his balance online and he can also transfer money to other account online. In this Software you can keep record for daily Banking transactions. The main purpose of developing bank management system is to design an application, which could store bank data and provide an interface for retrieving customer related details with 100% accuracy.

This bank management system also allow user to add new customer account, delete account and user can also modify existing user account information. Using this system user can also search any individual account in few seconds. Using our bank management system user can also check any translation in any account. Our system also provide security check to reduce fraud. The system will check the user' s existence in the database and provide the set of services with respect to the role of the user.

Following are the features of this Bank management system:

- *Create New Account*
- *Deposit Funds*
- *Withdraw Funds*
- *Check Balance*
- *View All Account Holder*
- *Close an Account*
- *Modify an Account*
- *Exit the Program*

2. Outline design

This is Bank management System in Cpp using Data Structures. The user information is in Linked List I have maintained logs of user transaction. So there are basically eight menu. The details of these with corresponding data structures are described below:

To Create New Account –

Following codes are used to create a new Account in a Bank Management System. If we add the Account Number, Account Name and Account Type and the initial balance, the Bank Account will be created.

```
void account::create_account()
{
    cout<<"\nEnter The account Number: ";
    cin>>accountnumber;
    cout<<"\nEnter The Name of The account Holder : ";
    cin.ignore();
    cin.getline(name,50);
    cout<<"\nEnter account type (enter s - saving or c - credit): ";
    cin>>type;
    type=toupper(type);
    cout<<"\nEnter The Initial amount(500 or more for Saving and 1000 or more for current ): ";
    cin>>deposit;
    cout<<"\n\nAccount Created Successfully...";
}
```

To Deposit Funds -

In this project, the following codes are used to deposit funds, draw funds, to get reports, to get account number, to get funds, to get account type in the bank management system.

```
void account::deposit_funds(int x)
{
    deposit+=x;
}

void account::draw_funds(int x)
{
    deposit-=x;
}

void account::get_report()
{
    cout<<accountnumber<<setw(10)<<" "<<name<<setw(10)<<" "<<type<<setw(6)<<deposit<<endl;
}

int account::get_accountnumber()
{
    return accountnumber;
}

int account::get_funds()
{
    return deposit;
}

char account::get_accounttype()
{
    return type;
}
```

To Withdraw Funds -

To withdraw the balance from a bank account in our bank management system, we will use the following codes -

```
void deposit_withdraw(int n, int option)
{
    int amt;
    bool found=false;
    account ac;
    fstream File;
    File.open("account.dat", ios::binary|ios::in|ios::out);
    if(!File)
    {
        cout<<"File could not be open !! Press any Key...";
        return;
    }
    while(!File.eof() && found==false)
    {
        File.read(reinterpret_cast<char*>(&ac), sizeof(account));
        if(ac.get_accountnumber()==n)
        {
            ac.show_account();
            if(option==1)
            {
                cout<<"\n\n\tTO DEPOSITE AMOUNT ";
                cout<<"\n\nEnter The amount to be deposited: ";
                cin>>amt;
                ac.deposit_funds(amt);
            }
            if(option==2)
            {
                cout<<"\n\n\tTO WITHDRAW AMOUNT ";
                cout<<"\n\nEnter The amount to be withdraw: ";
                cin>>amt;
                int bal=ac.get_funds()-amt;
                if((bal<500 && ac.get_accounttype()=='S') || (bal<1000 && ac.get_accounttype()=='C'))
                    cout<<"Insufficiency balance";
                else
                    ac.draw_funds(amt);
            }
            int pos=(-1)*static_cast<int>(sizeof(ac));
            File.seekp(pos,ios::cur);
            File.write(reinterpret_cast<char*>(&ac), sizeof(account));
            cout<<"\n\n\tRecord Updated";
            found=true;
        }
    }
    File.close();
    if(found==false)
        cout<<"\n\n Record Not Found ";
}
```

In this bank management system, users can check their bank balance and the details of their bank account whenever they want.

```
void account::show_account()
{
    cout<<"\nAccount Number: "<<accountnumber;
    cout<<"\nAccount Holder Name: "<<name;
    cout<<"\nType of Account: "<<type;
    cout<<"\nBalance amount: "<<deposit;
}
```

In this project, to view the list of all account holders, we will use the following codes. By using these codes the list of all users will be displayed.

```
void display_all()
{
    account ac;
    ifstream inFile;
    inFile.open("account.dat",ios::binary);
    if(!inFile)
    {
        cout<<"File could not be open !! Press any Key...";
        return;
    }

    cout<<"\n\n\t\tACCOUNT HOLDER LIST\n\n";
    cout<<"=====\n";
    cout<<"Account No.      Name      Type Balance\n";
    cout<<"=====\n";
    while(inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))
    {
        ac.get_report();
    }
    inFile.close();
}
```

To Close or Delete an Account –

If any user want to close or delete their bank account, it is possible in this bank management system. After deletion of any account, it will not have any access in our bank management system. We will use the following codes to delete an account from our bank management system.

```

void delete_account(int n)
{
    account ac;
    ifstream inFile;
    ofstream outFile;
    inFile.open("account.dat", ios::binary);
    if(!inFile)
    {
        cout<<"File could not be open !! Press any Key...";
        return;
    }
    outFile.open("Temp.dat", ios::binary);
    inFile.seekg(0, ios::beg);
    while(inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))
    {
        if(ac.get_accountnumber() != n)
        {
            outFile.write(reinterpret_cast<char *> (&ac), sizeof(account));
        }
    }

    inFile.close();
    outFile.close();
    remove("account.dat");
    rename("Temp.dat", "account.dat");
    cout<<"\n\n\tRecord Deleted...";
}

```

To Modify an Account –

In this bank management system, if both the users and the admin panel of the bank want, they can modify an account according to the corresponding information. The modification phase is very important in any bank management system. The account name, account type can be modified at any time and whenever any fund is deposited or withdrawn, it will automatically modified by the system. In this bank management project, we will use the following codes for the modification on any bank account.

```

void account::modify_account()
{
    cout<<"\nAccount Number: "<<accountnumber;
    cout<<"\nModify Account Holder Name: ";
    cin.ignore();
    cin.getline(name,50);
    cout<<"\nModify Type of Account: ";
    cin>>type;
    type=toupper(type);
    cout<<"\nModify Balance amount: ";
    cin>>deposit;
}

```

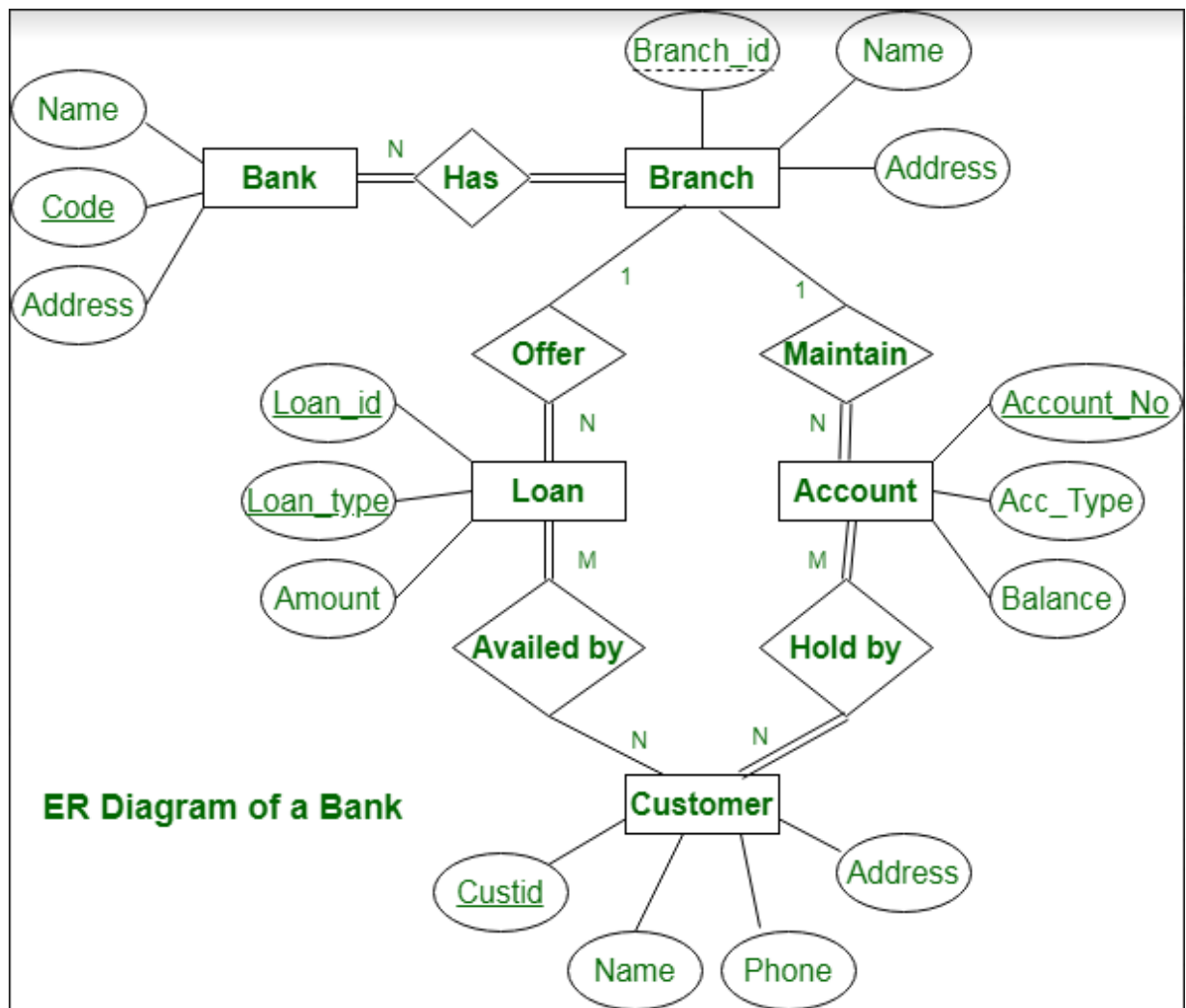

a project and compiled in **Code::Blocks** IDE using **GCC** compiler. In this system it is centered around client account administrations in bank, so it is named “Bank Management System”. Here, we can make record, update data of a current record, see and oversee exchanges, check the information of a current record, eliminate existing record and view clients’ lists. The source code for Bank Management System is generally short and straightforward. I have partitioned this C++ smaller than expected task into numerous capacities, a large portion of which are identified with various financial exercises.

ER diagram of Bank Management System

This bank ER diagram illustrates key information about bank, including entities such as branches, customers, accounts, and loans. It allows us to understand the relationships between entities.

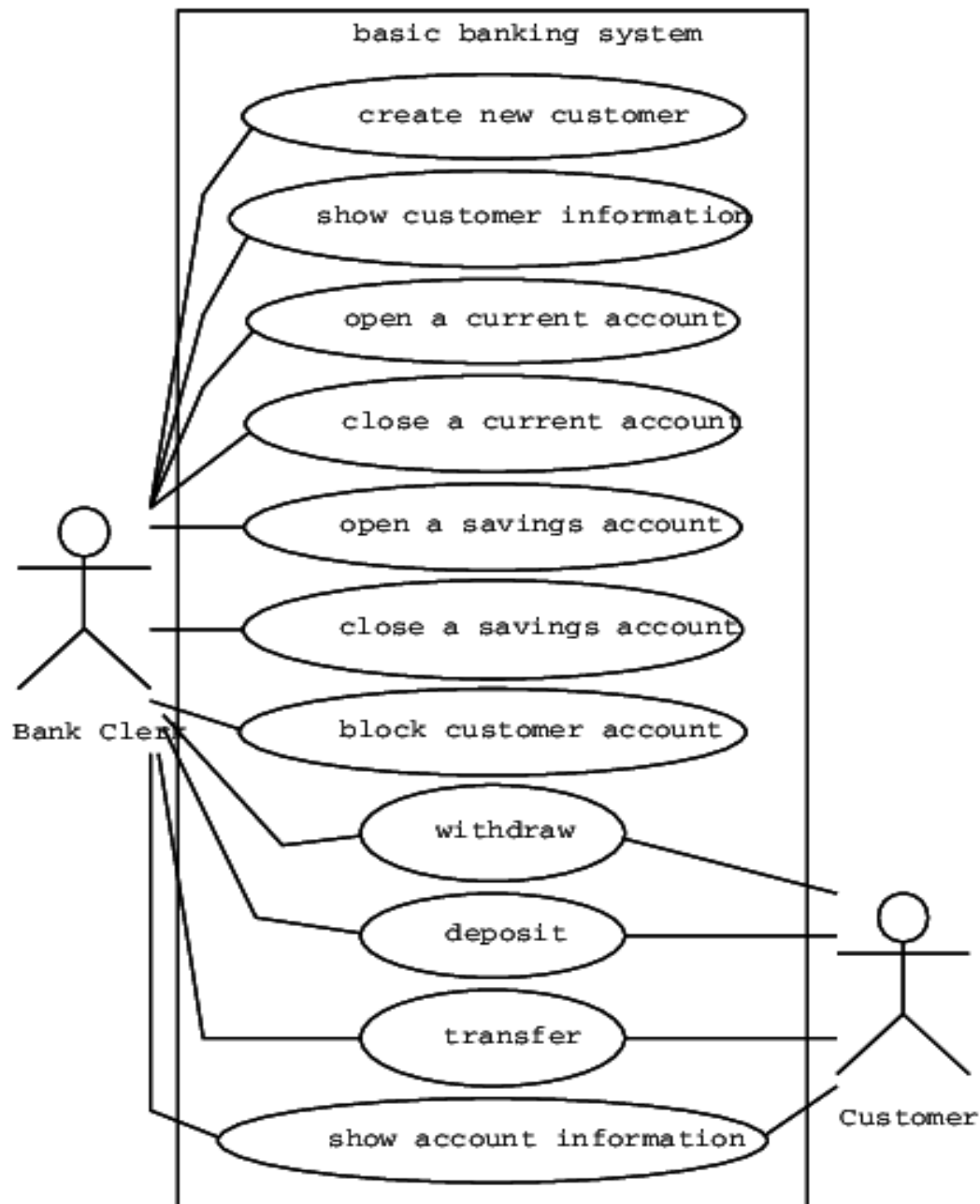
Entities and their Attributes are:

- **Bank Entity:** Attributes of Bank Entity are Bank Name, Code and Address.
Code is Primary Key for Bank Entity.
- **Customer Entity:** Attributes of Customer Entity are Customer_id, Name, Phone Number and Address.
Customer_id is Primary Key for Customer Entity.
- **Branch Entity:** Attributes of Branch Entity are Branch_id, Name and Address.
Branch_id is Primary Key for Branch Entity.
- **Account Entity:** Attributes of Account Entity are Account_number, Account_Type and Balance.
Account_number is Primary Key for Account Entity.
- **Loan Entity:** Attributes of Loan Entity are Loan_id, Loan_Type and Amount.
Loan_id is Primary Key for Loan Entity.



Use-case diagram of Bank Management System

The objective of a use case diagram is to show the interactions of numerous items called actors with the use case and to capture fundamental functionalities of a system. As you see through the diagrams, there are the use cases involved to define the core functions of a system. These processes were expected by the users to be connected to produce a certain output. Being a programmer, this could be an important role that the Bank Management should have.



Use-case diagram of Bank Management System

The Data Flow Diagram (DFD) represents the flow of data and the transformations in Bank Management System.

In the DFD, input, processing, and output are used to represent and define the overall system.

Upon reading through this article about Bank Management System you' ll then find how the system react with the user and handles data that the system is receiving from its customers.

The following are the flows that the Bank Management System can generate:

Manage Customers Information

Manage Bank Records

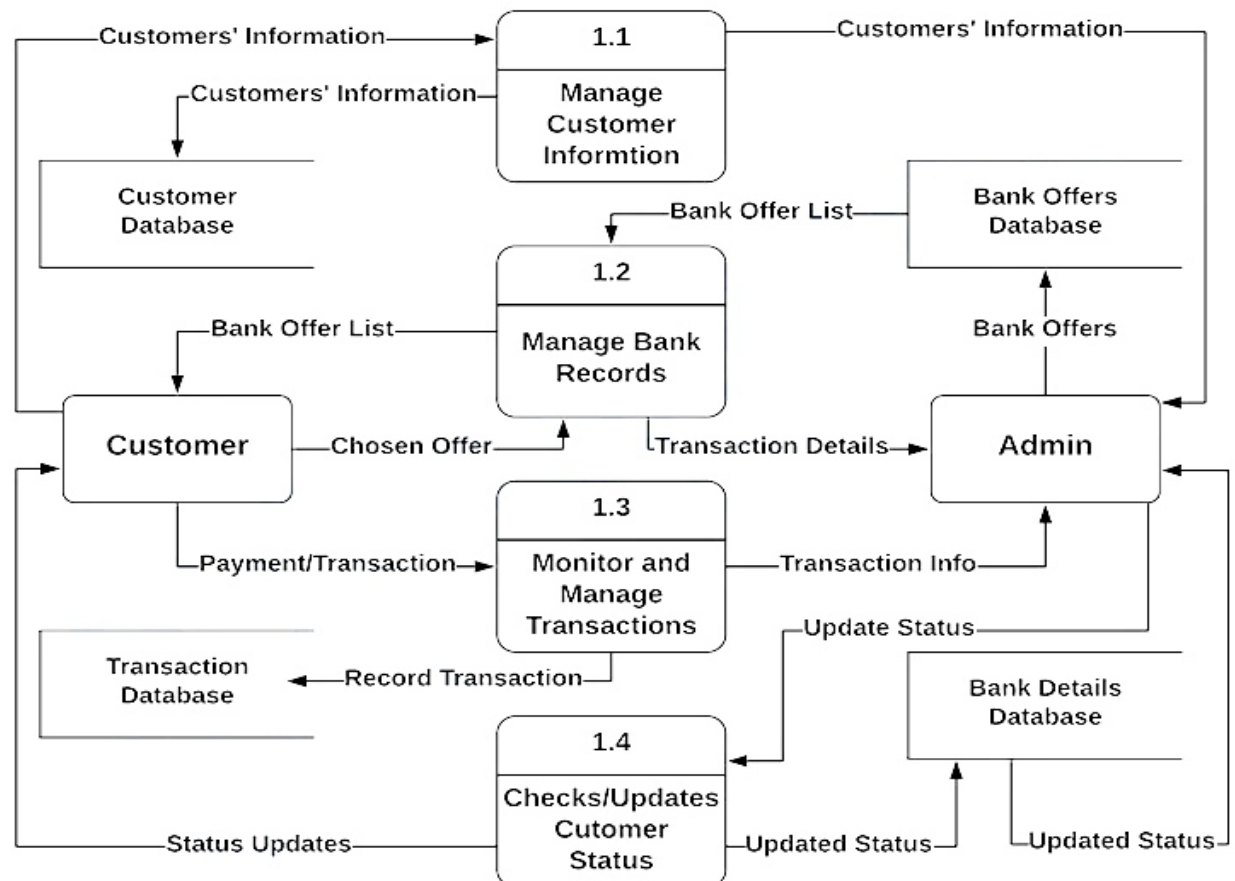
Checks Date of Transactions

Update Customers' Status

Monitor Debit/Credit Transactions

Manage Withdrawals/Savings


BANK MANAGEMENT SYSTEM



Screenshots of the outputs in the compiler of the project -

[illegible]

Now we will press 1 to create some new account.



```
D:\c++\Banking Record System.exe


Enter The account Number: 101

Enter The Name of The account Holder : Rudra

Enter account type (enter s - saving or c - credit): c

Enter The Initial amount(500 or more for Saving and 1000 or more for current ): 2000

Account Created Successfully...
```



```
"D:\c++\Banking Record System.exe"

Enter The account Number: 102

Enter The Name of The account Holder : Boult

Enter account type (enter s - saving or c - credit): c

Enter The Initial amount(500 or more for Saving and 1000 or more for current ): 3000

Account Created Successfully..._
```

```
"D:\c++\Banking Record System.exe"

Enter The account Number: 103

Enter The Name of The account Holder : Starc

Enter account type (enter s - saving or c - credit): s

Enter The Initial amount(500 or more for Saving and 1000 or more for current ): 3500

Account Created Successfully..._
```

Now we will press 2 for deposit fund in these accounts.

```
"D:\c++\Banking Record System.exe"

Enter The account No. : _
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 101

Account Number: 101
Account Holder Name: Rudra
Type of Account: C
Balance amount: 2000

TO DEPOSITE AMOUNT

Enter The amount to be deposited: 8000

Record Updated
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 102

Account Number: 102
Account Holder Name: Boul't
Type of Account: C
Balance amount: 3000

TO DEPOSITE AMOUNT

Enter The amount to be deposited: 3000

Record Updated
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 103

Account Number: 103
Account Holder Name: Starc
Type of Account: S
Balance amount: 3500

TO DEPOSITE AMOUNT

Enter The amount to be deposited: 4500

Record Updated
```

We have deposited the balance successfully. Now we will withdraw some amounts from these bank accounts. So we will press 3 for withdraw the balance.

```
"D:\c++\Banking Record System.exe"

Enter The account No. :
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 101

Account Number: 101
Account Holder Name: Rudra
Type of Account: C
Balance amount: 10000

TO WITHDRAW AMOUNT

Enter The amount to be withdraw: 1000

Record Updated_
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 102

Account Number: 102
Account Holder Name: Boulr
Type of Account: C
Balance amount: 6000

TO WITHDRAW AMOUNT

Enter The amount to be withdraw: 2000

Record Updated
```



```
"D:\c++\Banking Record System.exe"

Enter The account No. : 103
Account Number: 103
Account Holder Name: Starc
Type of Account: S
Balance amount: 8000

TO WITHDRAW AMOUNT
Enter The amount to be withdraw: 2000

Record Updated_
```

Now, we will check the balance of these accounts. So, we have to press 4 for checking the balance.

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 101
BALANCE DETAILS
Account Number: 101
Account Holder Name: Rudra
Type of Account: C
Balance amount: 9000
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 102
BALANCE DETAILS
Account Number: 102
Account Holder Name: Boulton
Type of Account: C
Balance amount: 4000
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 103

BALANCE DETAILS

Account Number: 103
Account Holder Name: Starc
Type of Account: S
Balance amount: 6000
```

We have checked the details of all accounts and the balances.
Now we will press 5 to see all the account holders.

```
"D:\c++\Banking Record System.exe"

ACCOUNT HOLDER LIST

=====
Account No.      Name      Type Balance
=====
101      Rudra      C  9000
102      Boulton     C  4000
103      Starc      S  6000
_
```

Here is the list of all account holders.
Now if we want to delete any of them, we have to press 6. Now
we will delete an account from the bank management system.

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 103

Record Deleted...█
```

After the deletion, we will check the list again by pressing 5.

```
"D:\c++\Banking Record System.exe"

ACCOUNT HOLDER LIST

=====
Account No.      Name      Type Balance
=====
101              Rudra      C  9000
102              Boulton    C  4000
_
```

We can modify the information of any account. Now, we will press 7 to modify an account.

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 102_
```

```
"D:\c++\Banking Record System.exe"

Enter The account No. : 102
Account Number: 102
Account Holder Name: Boulton
Type of Account: C
Balance amount: 4000

Enter The New Details of account:
Account Number: 102
Modify Account Holder Name: Jhon
Modify Type of Account: s
Modify Balance amount: 7000

Record Updated...
```

After the record modified successfully, we will check the details of the updated account.



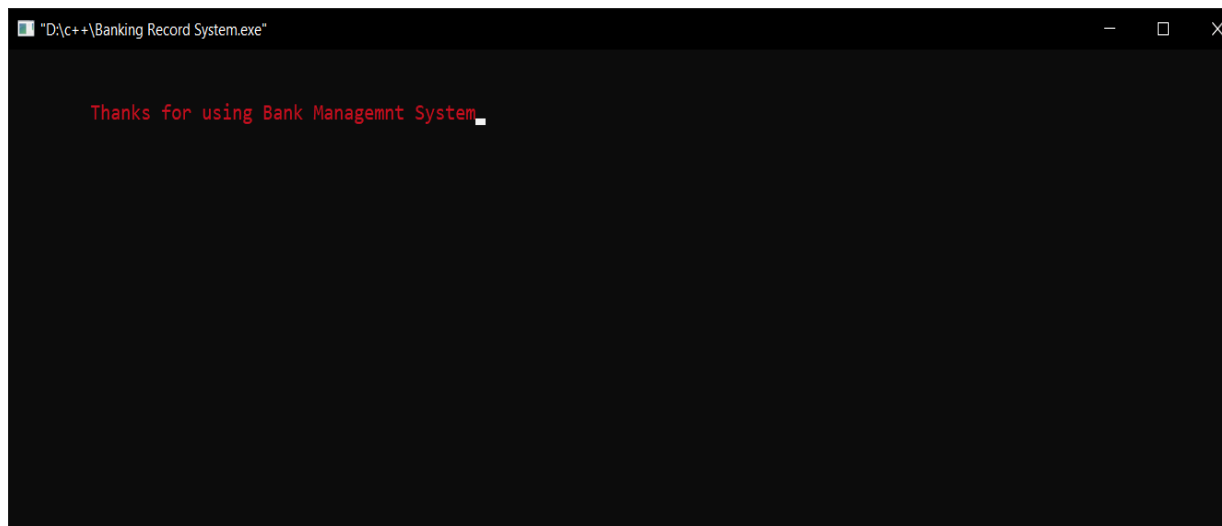
```
"D:\c++\Banking Record System.exe"

Enter The account No. : 102

BALANCE DETAILS

Account Number: 102
Account Holder Name: Jhon
Type of Account: S
Balance amount: 7000
```

And finally, we will exit the system by pressing 8.



```
"D:\c++\Banking Record System.exe"

Thanks for using Bank Managemnt System
```

In this projects -

Language used	-	C++
IDE used	-	Code::Blocks
Operating System	-	Windows 10

The Codes of the Project -

```
#include<iostream>

#include<fstream>

#include<cctype>

#include<iomanip>

using namespace std;

class account

{

    int accountnumber;

    char name[50];

    int deposit;

    char type;

public:

    void create_account();

    void show_account();

    void modify_account();

    void deposit_funds(int);

    void draw_funds(int);

    void get_report();

    int get_accountnumber();

    int get_funds();

    char get_accounttype();

};

void account::create_account()

{

    cout<<"\nEnter The account Number: ";

    cin>>accountnumber;

    cout<<"\nEnter The Name of The account Holder : ";

    cin.ignore();

    cin.getline(name,50);

    cout<<"\nEnter account type (enter s - saving or c - credit): ";

    cin>>type;

    type=toupper(type);

    cout<<"\nEnter The Initial amount(500 or more for Saving and 1000 or more for current ): ";

    cin>>deposit;
```

```

        cout<<"\n\nAccount Created Successfully...";
    }

    void account::show_account()
    {
        cout<<"\nAccount Number: "<<accountnumber;

        cout<<"\nAccount Holder Name: "<<name;

        cout<<"\nType of Account: "<<type;

        cout<<"\nBalance amount: "<<deposit;
    }

    void account::modify_account()
    {
        cout<<"\nAccount Number: "<<accountnumber;

        cout<<"\nModify Account Holder Name: ";

        cin.ignore();

        cin.getline(name,50);

        cout<<"\nModify Type of Account: ";

        cin>>type;

        type=toupper(type);

        cout<<"\nModify Balance amount: ";

        cin>>deposit;
    }

    void account::deposit_funds(int x)
    {
        deposit+=x;
    }

    void account::draw_funds(int x)
    {
        deposit-=x;
    }

    void account::get_report()
    {
        cout<<accountnumber<<setw(10)<<" "<<name<<setw(10)<<" "<<type<<setw(6)<<deposit<<endl;
    }

    int account::get_accountnumber()
    {
        return accountnumber;
    }

```

[illegible]

```

cout<<"\t\t@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@";

    cout<<"\n\t\tPress 1 to Create New Account";

    cout<<"\n\t\tPress 2 to Deposit Funds";

    cout<<"\n\t\tPress 3 to Withdraw Funds";

    cout<<"\n\t\tPress 4 to Check Balance";

    cout<<"\n\t\tPress 5 to View All Account Holder";

    cout<<"\n\t\tPress 6 to Close an Account";

    cout<<"\n\t\tPress 7 to Modify an Account";

    cout<<"\n\t\tPress 8 to Exit the Program";

cout<<"\n\t\t@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\n";

    cout<<"\n\t\tOption: ";

    cin>>opt;

    system("cls");

    switch(opt)

    {

    case '1':

        system("color 02");

            write_account();

            break;

    case '2':

        system("color 03");

            cout<<"\n\n\tEnter The account No. : "; cin>>num;

            deposit_withdraw(num, 1);

            break;

    case '3':

        system("color 06");

            cout<<"\n\n\tEnter The account No. : "; cin>>num;

            deposit_withdraw(num, 2);

            break;

    case '4':

        system("color 08");

            cout<<"\n\n\tEnter The account No. : "; cin>>num;

            display_sp(num);

            break;

    case '5':

        system("color 9");

            display_all();

```



```

        break;

    case '6':
        system("color 10");

        cout<<"\n\n\tEnter The account No. : "; cin>>num;

        delete_account(num);

        break;

    case '7':
        system("color 11");

        cout<<"\n\n\tEnter The account No. : "; cin>>num;

        modify_account(num);

        break;

    case '8':
        system("color 04");

        cout<<"\n\n\tThanks for using Bank Managemnt System";

        break;

    default :cout<<"Invalid Option\n";

}

cin.ignore();

cin.get();

if(opt=='8')

    break;

}

return 0;

}

void write_account()

{

    account ac;

    ofstream outFile;

    outFile.open("account.dat", ios::binary|ios::app);

    ac.create_account();

    outFile.write(reinterpret_cast<char *> (&ac), sizeof(account));

    outFile.close();

}

void display_sp(int n)

{

    account ac;

    bool flag=false;

```

```

ifstream inFile;

inFile.open("account.dat", ios::binary);

if(!inFile)

{

    cout<<"File could not be open !! Press any Key...";

    return;

}

cout<<"\nBALANCE DETAILS\n";

while(inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))

{

    if(ac.get_accountnumber()==n)

    {

        ac.show_account();

        flag=true;

    }

}

inFile.close();

if(flag==false)

    cout<<"\n\nAccount number does not exist";

}

void modify_account(int n)

{

    bool found=false;

    account ac;

    fstream File;

    File.open("account.dat", ios::binary|ios::in|ios::out);

    if(!File)

    {

        cout<<"File could not be open !! Press any Key...";

        return;

    }

    while(!File.eof() && found==false)

    {

        File.read(reinterpret_cast<char *> (&ac), sizeof(account));

        if(ac.get_accountnumber()==n)

        {

            ac.show_account();


```

```

        cout<<"\n\nEnter The New Details of account: "<<endl;

        ac.modify_account();

        int pos=(-1)*static_cast<int>(sizeof(account));

        File.seekp(pos, ios::cur);

        File.write(reinterpret_cast<char *> (&ac), sizeof(account));

        cout<<"\n\n\t Record Updated...";

        found=true;

    }

}

File.close();

if(found==false)

    cout<<"\n\n Record Not Found ";

}

void delete_account(int n)

{

    account ac;

    ifstream inFile;

    ofstream outFile;

    inFile.open("account.dat", ios::binary);

    if(!inFile)

    {

        cout<<"File could not be open !! Press any Key...";

        return;

    }

    outFile.open("Temp.dat", ios::binary);

    inFile.seekg(0, ios::beg);

    while(inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))

    {

        if(ac.get_accountnumber()!=n)

        {

            outFile.write(reinterpret_cast<char *> (&ac), sizeof(account));

        }

    }

    inFile.close();

    outFile.close();

    remove("account.dat");

    rename("Temp.dat", "account.dat");

```

```

        cout<<"\n\n\tRecord Deleted...";
    }

void display_all()
{
    account ac;

    ifstream inFile;

    inFile.open("account.dat", ios::binary);

    if(!inFile)

    {

        cout<<"File could not be open !! Press any Key...";

        return;

    }

    cout<<"\n\n\tACCOUNT HOLDER LIST\n\n";

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

    cout<<"Account No.      Name      Type Balance\n";

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

    while(inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))

    {

        ac.get_report();

    }

    inFile.close();
}

void deposit_withdraw(int n, int option)
{
    int amt;

    bool found=false;

    account ac;

    fstream File;

    File.open("account.dat", ios::binary|ios::in|ios::out);

    if(!File)

    {

        cout<<"File could not be open !! Press any Key...";

        return;

    }

    while(!File.eof() && found==false)

    {

        File.read(reinterpret_cast<char *> (&ac), sizeof(account));

```

```

        if(ac.get_accountnumber()==n)
        {
            ac.show_account();

            if(option==1)
            {
                cout<<"\n\n\tTO DEPOSITE AMOUNT ";

                cout<<"\n\nEnter The amount to be deposited: ";

                cin>>amt;

                ac.deposit_funds(amt);

            }

            if(option==2)
            {
                cout<<"\n\n\tTO WITHDRAW AMOUNT ";

                cout<<"\n\nEnter The amount to be withdraw: ";

                cin>>amt;

                int bal=ac.get_funds()-amt;

                if((bal<500 && ac.get_accounttype()=='S') || (bal<1000 && ac.get_accounttype()=='C'))

                    cout<<"Insufficiency balance";

                else

                    ac.draw_funds(amt);

            }

            int pos=(-1)*static_cast<int>(sizeof(ac));

            File.seekp(pos,ios::cur);

            File.write(reinterpret_cast<char*>(&ac), sizeof(account));

            cout<<"\n\n\t Record Updated";

            found=true;

        }

    }

    File.close();

    if(found==false)

        cout<<"\n\n Record Not Found ";

}

```

4. Conclusion

Bank management system is a virtualization of transactions in banking system. The banking system are used manual working but when we

used online banking system it is totally virtualization process which avoid manual process and converts it in automatic process . If user can make a transaction in bank management system it is available in any were also user can link other with account, change branch location easily. Bank management system is saving the time with accuracy than bank manual system.

“Bank Management System ” keeps the day by day tally record as a complete banking. It can keep the information of Account type, account opening form, Deposit, Withdrawal, and Searching the transaction, Transaction report, Individual account opening form, Group Account. The exciting part of this project is; it displays Transaction reports, Statistical Summary of Account type and Interest Information.

PROBLEM OF EXISTING SYSTEM:

Manual operator control:

Manual operator control is there and leads to a lot of chaos and errors.

.

Inability of sharing the data:

Data cannot be shared in the existing system. This means that no two persons can use the same data in existing system also the two departments in an organization cannot interact with each other without the actual movement of data.

.

No support in decision-making:

Existing system does not support managerial decision-making.

.

No support in strategic competitive advantage:

Existing system do not support strategic competitive advantages.

This was my project of System Design about “Bank Management System” . Development of this System takes a lot of efforts. I think this system gave a lot of satisfaction. Though every task is never said to be perfect in this development field even more improvement may be possible in this system. I learnt so many things and gained a lot of knowledge about development field. I hope this will prove fruitful.