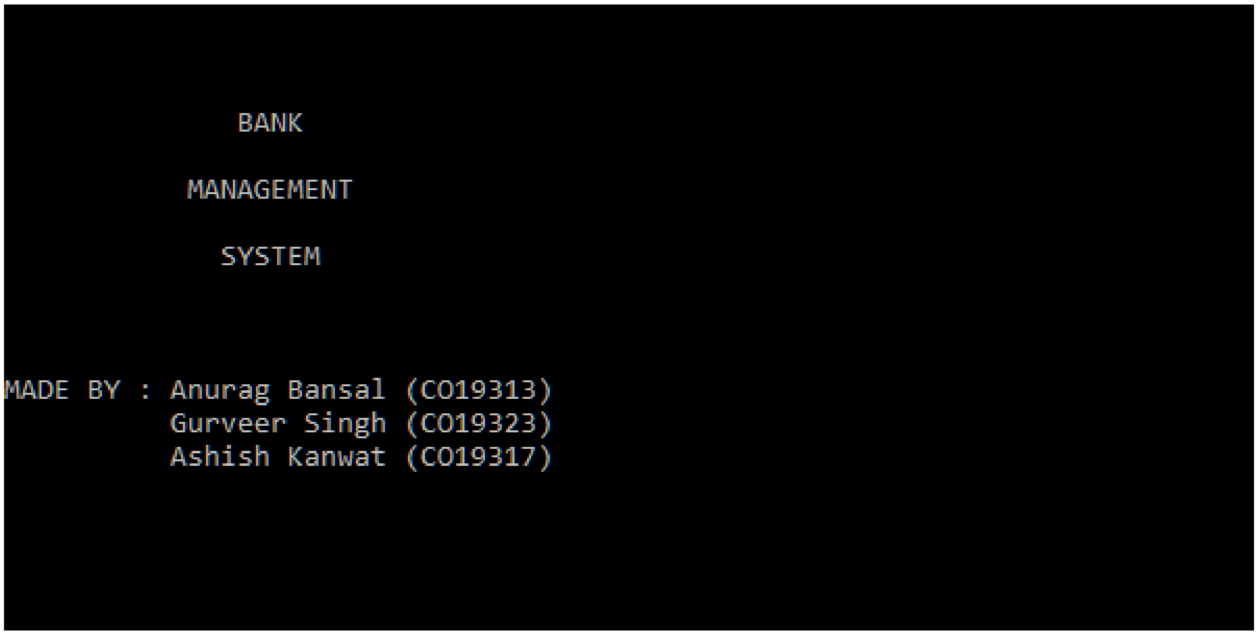
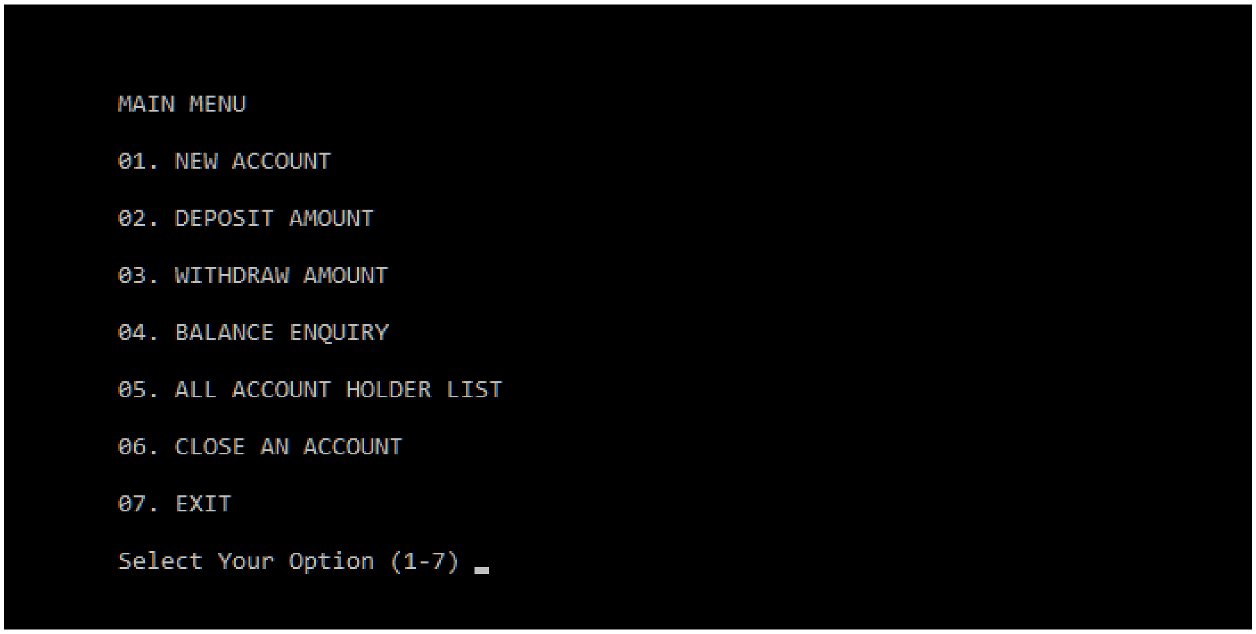
```
//***************************************************************
//HEADER FILE
//****************************************************************

#include <iostream>
#include <fstream>
#include <cctype>
#include <iomanip>
using namespace std;

//***************************************************************
//CLASS
//****************************************************************

class account
{
    int acno;
    char name[50];
    int deposit;
    char type;

public:
    void create_account();     //function to get data from user
    void show_account() const; //function to show data on screen
    void dep(int);             //function to accept amount and add to balance
 amount
    void draw(int);            //function to accept amount and subtract from
balance amount
    void reporttable() const;  //function to show data in tabular format
    int returnacno() const;    //function to return account number
    int returndeposit() const; //function to return balance amount
    char returntype() const;   //function to return type of account
};

void account::create_account()
{
    account ac;
label1:
    cout << "\nEnter The account No. : ";
    cin >> acno;
    ifstream inFile;
    inFile.open("account.dat", ios::app);
    while (inFile.read(reinterpret_cast<char *>(&ac), sizeof(account)))
    {
        if (ac.returnacno() == acno)
        {
            cout << "Account Number already in use\n";
            goto label1;
        }
    }
    cout << "\nEnter The Name of The account Holder : ";
    cin.ignore();
    cin.getline(name, 50);
label2:
    cout << "\nEnter Type of The account (C/S) : ";
    cin >> type;
    type = toupper(type);
    if (type == 'C')
    {
        goto label3;
    }
    else if (type == 'S')
    {
        goto label3;
    }
    else
    {
        cout << "Input the correct account type\n";
        goto label2;
    }
label3:
    cout << "\nEnter The Initial amount(>=500 for Saving and >=1000 for curre
nt ) : ";
    cin >> deposit;
    if (type == 'C' && deposit < 1000)
    {
        cout << "For Current Account deposit should be greater than 1000\n";
        goto label3;
    }
    else if (type == 'S' && deposit < 500)
    {
        cout << "For Savings Account deposit should be greater than 500\n";
        goto label3;
    }
    cout << "\n\n\nAccount Created..";
}

void account::show_account() const
{
    cout << "\nAccount No. : " << acno;
    cout << "\nAccount Holder Name : ";
    cout << name;
    cout << "\nType of Account : " << type;
    cout << "\nBalance Amount : " << deposit;
}

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

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

void account::reporttable() const
{
    cout << acno << setw(10) << " " << name << setw(10) << " " << type << set
w(6) << deposit << endl;
}

int account::returnacno() const
{
    return acno;
}

int account::returndeposit() const
{
    return deposit;
}

char account::returntype() const
{
    return type;
}

//*****************************************************************
//FUNCTION DECLARATION
//******************************************************************

void write_account();            //function to write record in binary file
void display_specific(int);      //function to display account details given
by user
void delte_account_info(int);    //function to delete record of file
void display_all_details();      //function to display all account details
void deposit_withdraw(int, int); // function to desposit/withdraw amount for
given account
void introduction();             //introductory screen function

//****************************************************************
//MAIN FUNCTION
//*****************************************************************

int main()
{
    char ch;
    int num;
    introduction();
    do
    {
        system("cls");
        cout << "\n\n\n\tMAIN MENU";
        cout << "\n\n\t01. NEW ACCOUNT";
        cout << "\n\n\t02. DEPOSIT AMOUNT";
        cout << "\n\n\t03. WITHDRAW AMOUNT";
        cout << "\n\n\t04. BALANCE ENQUIRY";
        cout << "\n\n\t05. ALL ACCOUNT HOLDER LIST";
        cout << "\n\n\t06. CLOSE AN ACCOUNT";
        cout << "\n\n\t07. EXIT";
        cout << "\n\n\tSelect Your Option (1-7) ";
        cin >> ch;
        system("cls");
        switch (ch)
        {
        case '1':
            write_account();
            break;
        case '2':
            cout << "\n\n\tEnter The account No. : ";
            cin >> num;
            deposit_withdraw(num, 1);
            break;
        case '3':
            cout << "\n\n\tEnter The account No. : ";
            cin >> num;
            deposit_withdraw(num, 2);
            break;
        case '4':
            cout << "\n\n\tEnter The account No. : ";
            cin >> num;
            display_specific(num);
            break;
        case '5':
            display_all_details();
            break;
        case '6':
            cout << "\n\n\tEnter The account No. : ";
            cin >> num;
            delte_account_info(num);
            break;
        case '7':
            cout << "\n\n\tThanks for using Bank Management System";
            break;
        default:
            cout << "\a";
        }
        cin.ignore();
        cin.get();
    } while (ch != '7');
    return 0;
}

//*****************************************************************
//WRITE FUNCTION
//******************************************************************

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();
}

//*****************************************************************
//READING SPECIFIC RECORD
//******************************************************************

void display_specific(int n)
{
    account ac;
    bool flag = false;
```

```cpp
    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.returnacno() == n)
        {
            ac.show_account();
            flag = true;
        }
    }
    inFile.close();
    if (flag == false)
        cout << "\n\nAccount number does not exist";
}

//****************************************************************
//DELETE FUNCTION
//*****************************************************************

void delte_account_info(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.returnacno() != 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 ..";
}

//****************************************************************
//DISPLAY FUNCTION
//*****************************************************************

void display_all_details()
{
    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 << "A/c no.      NAME           Type  Balance\n";
    cout << "====================================================\n";
    while (inFile.read(reinterpret_cast<char *>(&ac), sizeof(account)))
    {
        ac.reporttable();
    }
    inFile.close();
}

//****************************************************************
//DEPOSIT/WITHDRAW FUNCTION
//*****************************************************************

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.returnacno() == n)
        {
            ac.show_account();
            if (option == 1)
            {
                cout << "\n\n\tTO DEPOSITE AMOUNT ";
                cout << "\n\nEnter The amount to be deposited ";
                cin >> amt;
                ac.dep(amt);
            }
            if (option == 2)
            {
                cout << "\n\n\tTO WITHDRAW AMOUNT ";
                cout << "\n\nEnter The amount to be withdrawn ";
                cin >> amt;
                int bal = ac.returndeposit() - amt;
                if ((bal < 500 && ac.returntype() == 'S') || (bal < 1000 && a
c.returntype() == 'C'))
                    cout << "Insufficience balance ";
                else
                    ac.draw(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 ";
}

//*****************************************************************
//INTRODUCTION FUNCTION
//******************************************************************

void introduction()
```

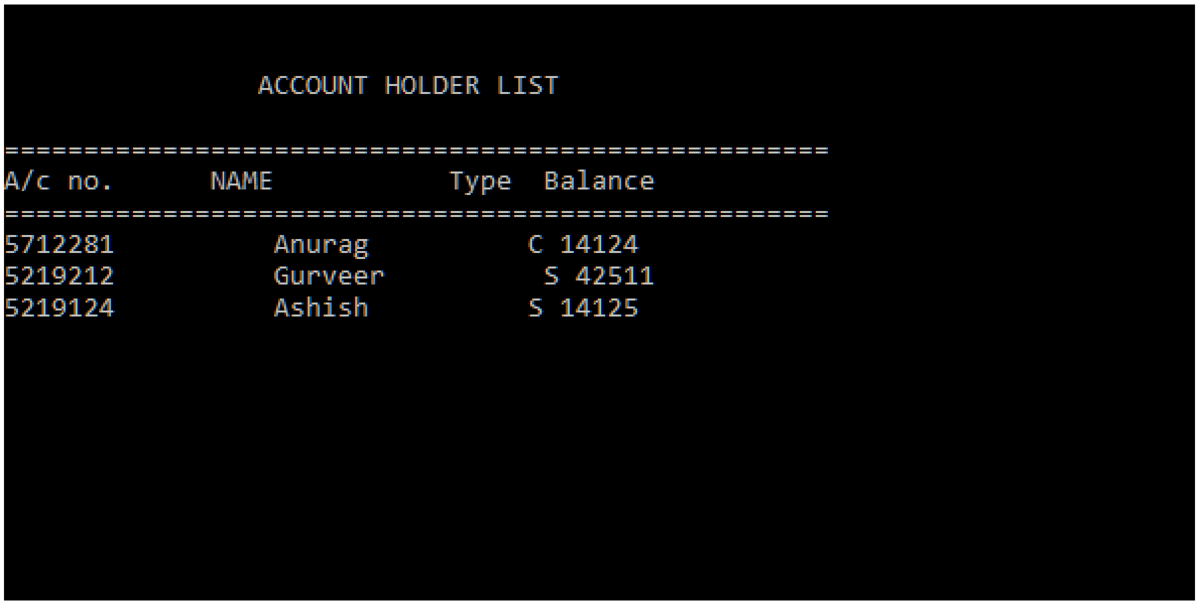
```
{
    cout << "\n\n\n\t      BANK";
    cout << "\n\n\t   MANAGEMENT";
    cout << "\n\n\t     SYSTEM";
    cout << "\n\n\n\nMADE BY : Anurag Bansal (CO19313)";
    cout << "\n          Gurveer Singh (CO19323)";
    cout << "\n          Ashish Kanwat (CO19317)";
    cin.get();
}
```



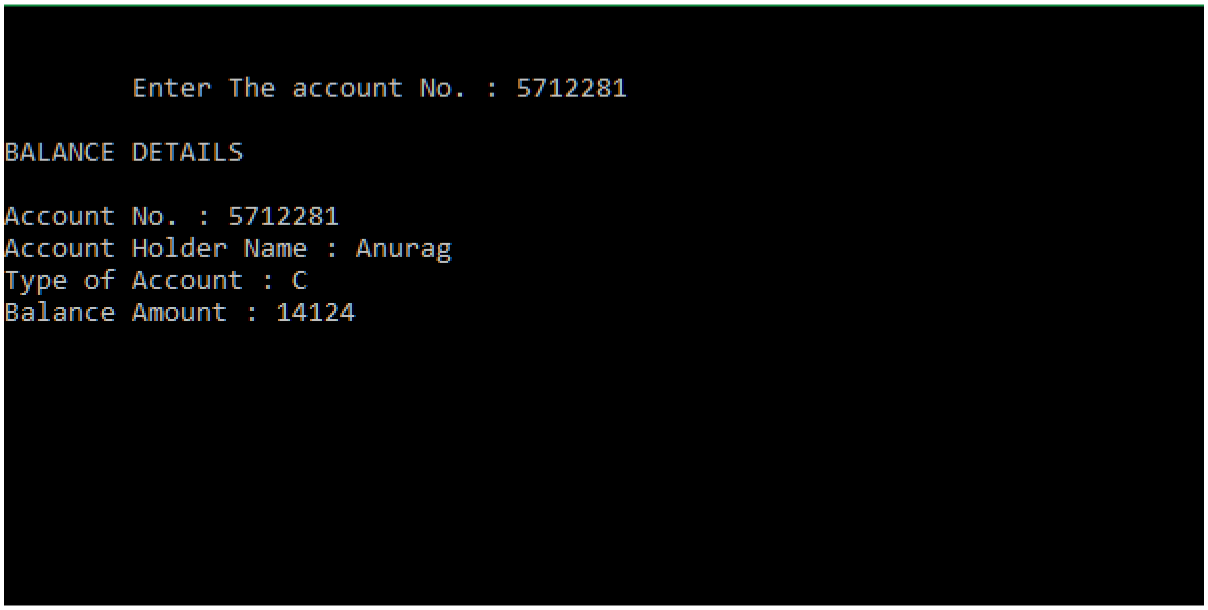
```
            BANK

         MANAGEMENT

           SYSTEM



MADE BY : Anurag Bansal (CO19313)
          Gurveer Singh (CO19323)
          Ashish Kanwat (CO19317)
```



```
        MAIN MENU

        01. NEW ACCOUNT

        02. DEPOSIT AMOUNT

        03. WITHDRAW AMOUNT

        04. BALANCE ENQUIRY

        05. ALL ACCOUNT HOLDER LIST

        06. CLOSE AN ACCOUNT

        07. EXIT

        Select Your Option (1-7) _
```



```
                ACCOUNT HOLDER LIST

====================================================
A/c no.      NAME           Type  Balance
====================================================
5712281          Anurag          C 14124
5219212          Gurveer          S 42511
5219124          Ashish          S 14125
```



```
        Enter The account No. : 5712281

BALANCE DETAILS

Account No. : 5712281
Account Holder Name : Anurag
Type of Account : C
Balance Amount : 14124
```

Bank Management System

made by :

gurveer(CO19323)

anurag(CO19313)

ashish(CO19317)

AIM

To develop a bank management system for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks

Problem Description

The bank management system is an application for maintaining a person's account in a bank . The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. The following presentation provides the specification for the system.

Language used : c++

The c++ code used for banking management system , includes file system access, so all the data gets stored in external '.dat' binary file that will be created during runtime

CLASS

Account

MODULES

- **Create Account** Opens a new account for the user by accepting input such as account number ,name and account type.

- **Show Account** Function to show the data given by the user.

- **Deposit** Function to accept amount and add to balance amount.

- **Withdraw** Provides options to withdraw amount from the given account number.

- Report Provides the options to show the data in tabular format.
- Return Account no Function to return account number
- Return Deposit Provides options to return the balance amount.
- Return type Function to return the type of the account .

CONCLUSION

- This banking system will serve as useful approach to deposit and withdraw the money for the person.
- It reduces the time taken by the user to save the money.
- Bank system developed is user friendly.
- It reduces manual work

- THANK YOU