



Name of Student: YASIR ZAHID 191017

Section: BE MTS 2 B Date: 22-06-20

Name of Instructor: SIR ALI

Instructor's Signature: _____

COMPLEX ENGINEERING ACTIVITY



Name of Student: HAMNA FATIMA 190984

Section: BE MTS 2 B Date: 22-06-20

Name of Instructor: SIR ALI

Instructor's Signature: _____

COMPLEX ENGINEERING ACTIVITY



Name of Student: SYED FASIH 191029.

Section: **BE MTS 2 B** Date: **22-06-20**

Name of Instructor: **SIR ALI**

Instructor's Signature: _____

COMPLEX ENGINEERING ACTIVITY

COMPUTER PROGRAMMING LAB



Complex Engineering Activity Report

LAB ENGINEER: SIR ALI RAZA

MTS-2-B

- YASIR ZAHID 191017
- SYED FASIH 191029
- HAMNA FATIMA 190954

Complex Engineering Activity Report

Problem Statement:

Bank Management System is based on a concept of recording customer's account details. Here the user can perform all the tasks like creating an account, deposit amount, withdraw amount, check balance, view all account holders detail, close an account and modify an account.

Your system should work like a banking system and should only be accessible by correct username and pin number. Database for bank users and account should be maintain in a file and access into system through file handling in C++

Following should be the main features and the menu page of the system:

- Add new account
- Deposit amount
- Withdraw amount
- Balance enquiry
- List of all account holders in bank
- Close or delete an account
- Modify detail of an account
- Exit

An account can be of type Current or Saving and the minimum amount for opening an account in bank should be Rs.1000. Minimum amount that can be withdrawn will be Rs.500 and withdrawn amount should be in multiple of 500.

1. Use of structures and file handling is must.
2. Consider all cases for input , crashing of a system, will result in marks deduction, e.g if a user enters wrong password an error message should be shown , if amount is not valid to credit/withdraw a message should be displayed rather than system crashing
3. Other than mentioned functions bonus function related to banking system will result in bonus marks.
4. Marks will depend upon the difficulty level of systems working.
5. A project report containing the working code and screenshot needs to be submitted at the time of viva. Report should contain flow diagram representing the working of system, result and conclusion

WORKING CODE



FINAL CODE.cpp

CLICK ON ABOVE ICON TO OPEN CPP SOURCE FILE
OR GO TO BOTTOM OF DOCUMENT

SCREENSHOTS

MAIN INTERFACE:

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

WELCOME TO BANKING SYSTEM ORGANIZER
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:LOGIN
2:CREATE AN ACCOUNT
3.EXIT
_
```

CREATE AN ACCOUNT:

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

WELCOME TO BANKING SYSTEM ORGANIZER
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:LOGIN
2:CREATE AN ACCOUNT
3.EXIT
2
CREATE A NEW ACCOUNT
USERNAME: USER2
USERNAME ALREADY EXISTS
CREATE A NEW ACCOUNT
USERNAME: USER3
PASSWORD: PASSWORD3
OPENING DEPOSIT: 3000

MINIMUM ACCOUNT OPENING BALANCE IS RS:1000/-
OPENING DEPOSIT: 3000

CONGRATULATIONS!! ACCOUNT OPENED SUCCESSFULLY

WELCOME TO BANKING SYSTEM ORGANIZER
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:LOGIN
2:CREATE AN ACCOUNT
3.EXIT
```

MISCELLANEOUS INCASE OF OPENING DEPOSIT INVALID AMOUNT:

```
C:\Users\YASIRA~1\AppData\Local\Temp\FINAL CODE.exe

WELCOME TO BANKING SYSTEM ORGANIZER
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:LOGIN
2:CREATE AN ACCOUNT
3.EXIT
2
CREATE A NEW ACCOUNT
USERNAME: user4
PASSWORD: password4
OPENING DEPOSIT: abc

INVALID AMOUNT...

WOULD YOU LIKE TO
1:CONTINUE
2:EXIT
```

LOGIN:

ATTEMPT 1 WRONG USER NAME

ATTEMPT 2 WRONG PASSWORD (I.E...USED PASSWORD OF ANOTHER ACCOUNT)

ATTEMPT 3 CORRECT USERNAME AND PASSWORD.....SUCCESSFULLY LOGGED IN

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

USERNAME: USER4
WRONG USERNAME
CONTINUE?
1:YES
2:NO
1

ENTER USERNAME AND PASSWORD
USERNAME: USER1
PASSWORD:PASSWORD2
WRONG PASSWORD
CONTINUE?
1:YES
2:NO
1

ENTER USERNAME AND PASSWORD
USERNAME: USER1
PASSWORD:PASSWORD1
YOU HAVE LOGGED IN SUCCESSFULLY

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```

POST LOGIN INTERFACE

```
ENTER USERNAME AND PASSWORD
USERNAME: USER1
PASSWORD:PASSWORD1
YOU HAVE LOGGED IN SUCCESSFULLY

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```

CHECKING ACCOUNT BALANCE:

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

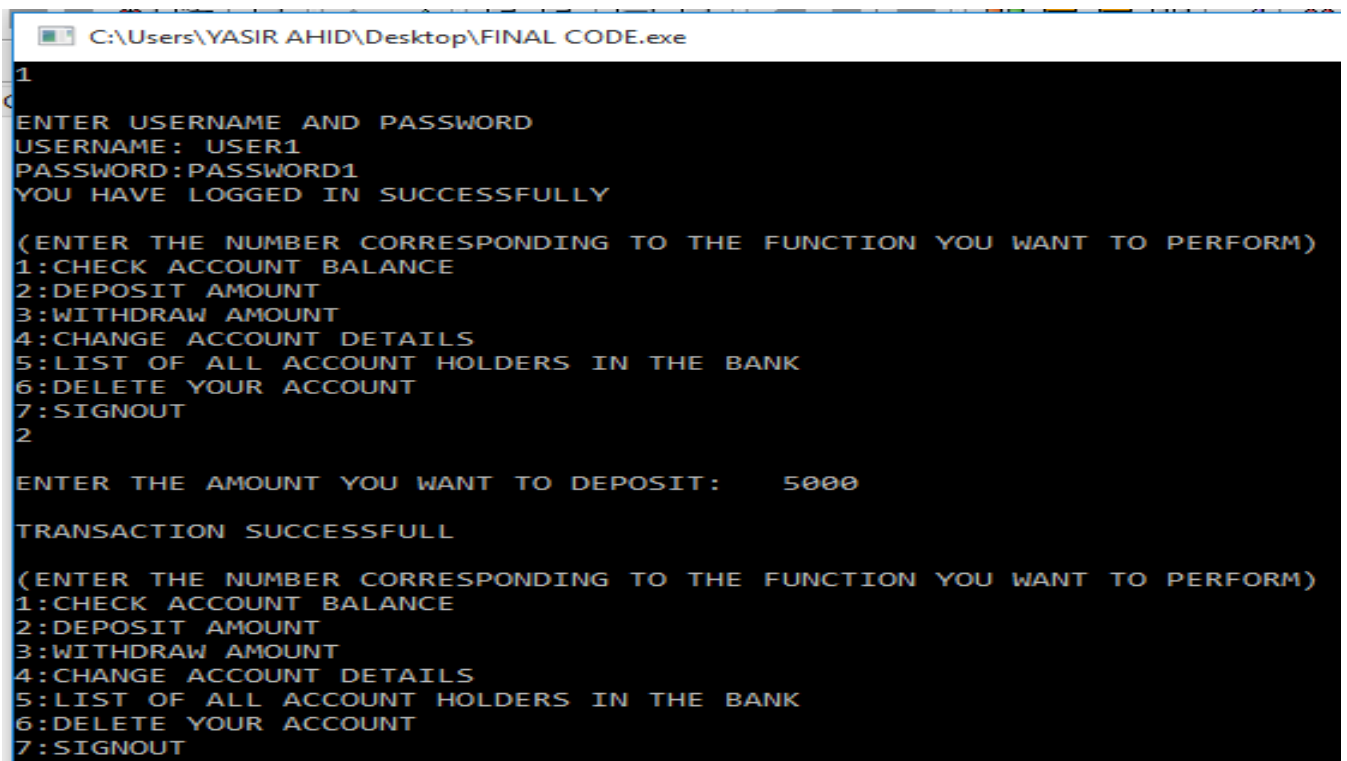
ENTER USERNAME AND PASSWORD
USERNAME: USER3
PASSWORD:PASSWORD3
YOU HAVE LOGGED IN SUCCESSFULLY

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
1

YOUR ACCOUNT BALANCE IS: 3000

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```


DEPOSIT AMOUNT:



```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe
1
ENTER USERNAME AND PASSWORD
USERNAME: USER1
PASSWORD:PASSWORD1
YOU HAVE LOGGED IN SUCCESSFULLY

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
2

ENTER THE AMOUNT YOU WANT TO DEPOSIT: 5000

TRANSACTION SUCCESSFULL

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```

WITHDRAW AMOUNT:

ATTEMPT 1 AMOUNT NOT MULTIPLE OF 500

ATTEMPT 2 AMOUNT MORE THAN BALANCE

ATTEMPT 3 CORRECT AMOUNT

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
3

ENTER THE AMOUNT YOU WANT TO WITHDRAW: 40
SORRY, AMOUNT SHOULD BE MULTIPLE OF 500
ENTER THE AMOUNT YOU WANT TO WITHDRAW: 40000
SORRY THIS WITHDRAWAL CAN,T BE PROCESSED YOU HAVE INSUFFICIENT BALANCE
ENTER THE AMOUNT YOU WANT TO WITHDRAW: 1000
TRANSACTION SUCCESSFULL

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```

LIST OF ALL ACCOUNT HOLDERS

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

ENTER USERNAME AND PASSWORD
USERNAME: USER5
PASSWORD:PASSWORD5
YOU HAVE LOGGED IN SUCCESSFULLY

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
5
USER1
USER2
USER3
USER4
USER5
LIST ENDS HERE
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```

CHANGE ACCOUNT DETAILS (SAME IS FOR PASSWORD CHANGE):

SEE IN LIST BEFORE AND AFTER NAME CHANGED....

```
C:\Users\YASIR AHID\Desktop\FINAL CODE.exe

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
5
USER1
USER2
USER3
USER4
USER5
LIST ENDS HERE
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
4

(ENTER 1 OR 2 TO PERFORM THE OPERATION)
CHANGE
1 : USERNAME
2 : PASSWORD
1

ENTER NEW USERNAME :
USER8

OPERATION COMPLETED SUCCESSFULLY

(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
```

DELETE YOUR ACCOUNT

OBSERVE THAT USER 8 WAS IN LIST AFTER DELETION LOGIN SAYS WRONG USERNAME HENCE ACCOUNT IS DELETED....

```

USER1
USER2
USER3
USER4
USER8
LIST ENDS HERE
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
6

(ENTER 1 OR 2 TO PERFORM THE OPERATION)
ARE YOU SURE YOU WANT TO DELETE YOUR ACCOUNT
1:YES
2:NO
1
PROCESSING...PROCESSING...PROCESSING...
YOUR REQUEST COMPLETED YOU WILL BE REFERRED BACK TO MAIN MENU

WELCOME TO BANKING SYSTEM ORGANIZER
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:LOGIN
2:CREATE AN ACCOUNT
3.EXIT
1

ENTER USERNAME AND PASSWORD
USERNAME: USER8
WRONG USERNAME
CONTINUE?
1:YES
2:NO

```

SIGNOUT

```

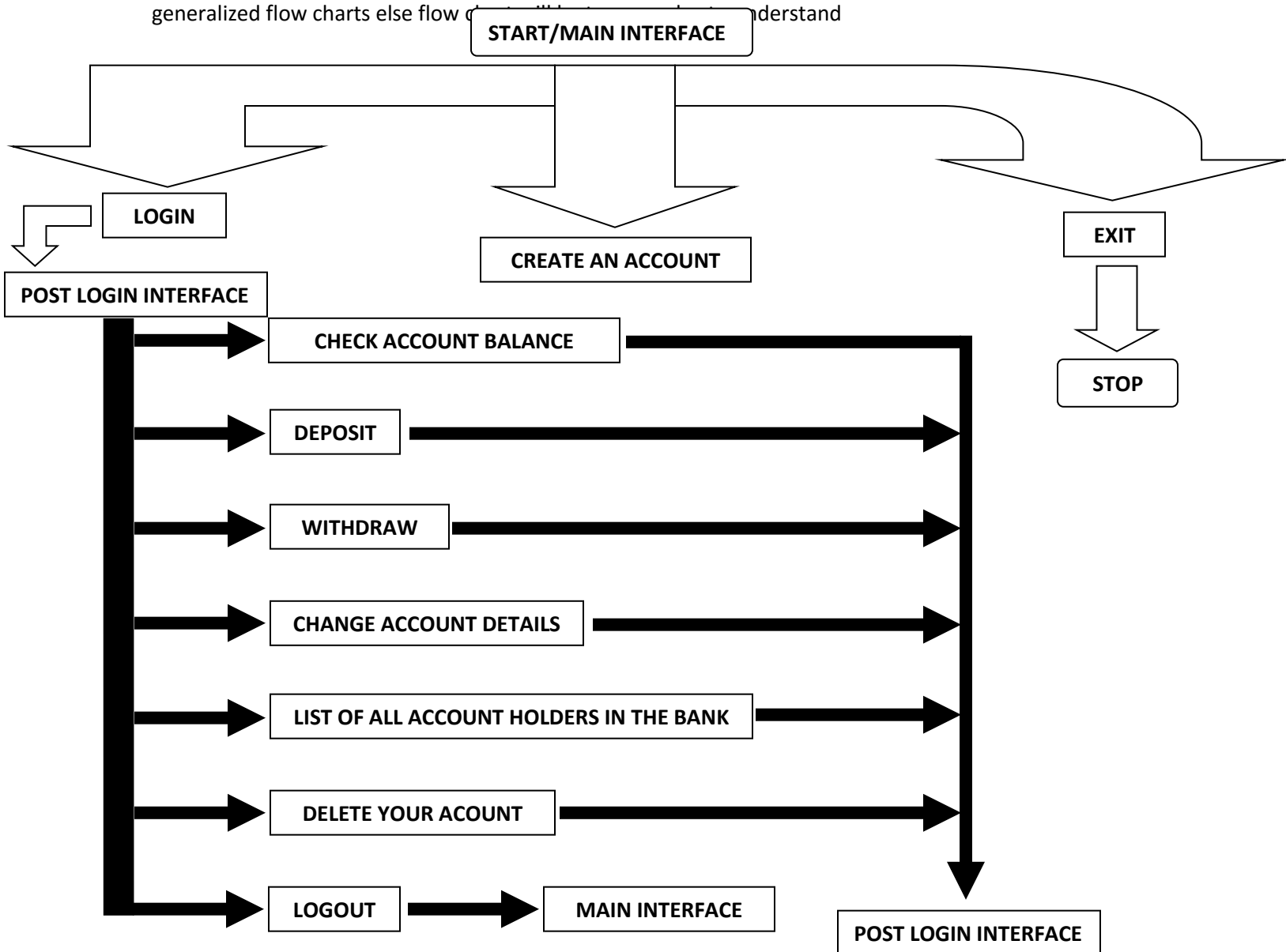
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:CHECK ACCOUNT BALANCE
2:DEPOSIT AMOUNT
3:WITHDRAW AMOUNT
4:CHANGE ACCOUNT DETAILS
5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK
6:DELETE YOUR ACCOUNT
7:SIGNOUT
7

WELCOME TO BANKING SYSTEM ORGANIZER
(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)
1:LOGIN
2:CREATE AN ACCOUNT
3.EXIT

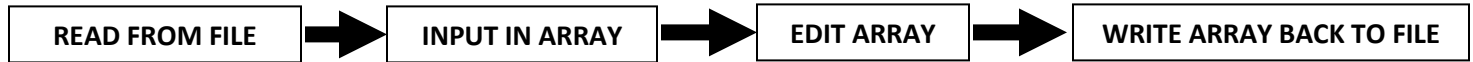
```

FLOW DIAGRAM

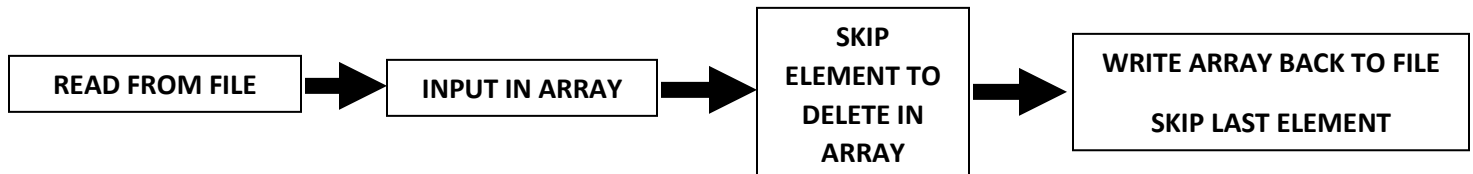
This program uses a class containing 25 different functions to complete its operation hence we will go for an overall and individual flow diagrams to thoroughly go through the working however these will be generalized flow charts else flow charts will be too complex to understand



**METHOD OF EDITING FOLLOWED FOR
DEPOSIT/WITDRAWAL/CHANGE USERNAME/PASSWORD**



METHOD OF EDITING FOLLOWED FOR DELETING AN ACCOUNT



WORKING CODE

```
#include <iostream>

#include <string>    //INTEGER TO STRING CONVERSION
#include <sstream>    //STRING TO INTEGER CONVERSION
#include <fstream>    //FILE HANDLING
#include <cstring>    //STRING ARRAY HANDLING
#include <math.h>    //TO USE ADVANCED MATHEMATICAL FUNCTIONS IN ENCRYPTION
DECRYPTION

using namespace std;

class MANAGER{
public:
    MANAGER(){
        ACCESS = 0;
    }
    void LOGIN(){
        ID="";

        cout << endl << "ENTER USERNAME AND PASSWORD " << endl << "USERNAME: ";
        cin >> USERNAMEATTEMPT;

        int USERID = CHECKID(USERNAMEATTEMPT, "USERS.txt");
```

```

if(USERID != 0){
    cout << "PASSWORD:";
    cin >> PASSWORDATTEMPT;
    int PASSWORDID = CHECKID(PASSWORDATTEMPT, "PASSWORDS.txt");
    if(USERID == PASSWORDID){
        cout << "YOU HAVE LOGGED IN SUCCESSFULLY" << endl;
        ID1=PASSWORDID;
        stringstream x;
        x<<PASSWORDID;
        x>>ID;
        POSTLOGININTERFACE();
    }
    else{
        cout << "WRONG PASSWORD" << endl;
        cout<<"CONTINUE?"<<endl<<"1:YES"<<endl<<"2:NO"<<endl;
        int command;
        cin>>command;
        if(command==1){
            LOGIN();
        }else
            if(command==2){
                MAININTERFACE();
            }
        else{
            cout<<endl<<"INVALID COMMAND"<<endl;
            MAININTERFACE();
        };
    }
}

```

```

    }
    else{
        cout << "WRONG USERNAME" << endl;
        cout<<"CONTINUE?"<<endl<<"1:YES"<<endl<<"2:NO"<<endl;
        int command;
        cin>>command;
        if(command==1){
            LOGIN();
        }else
        if(command==2){
            MAININTERFACE();
        }
        else{
            cout<<endl<<"INVALID COMMAND"<<endl;
            MAININTERFACE();
        };
    }
}

void OPENANACCOUNTSTAGE1(){
    cout<<"CREATE A NEW ACCOUNT"<<endl<<"USERNAME: ";
    cin>>NEWUSERNAME;
    if(CHECKID(NEWUSERNAME, "USERS.txt") != 0){
        cout << "USERNAME ALREADY EXISTS" << endl;
        OPENANACCOUNTSTAGE1();
        return;
    }
    cout<<"PASSWORD: ";
    cin>>NEWPASSWORD;

```



```

OPENANACCOUNTSTAGE2();
}

void OPENANACCOUNTSTAGE2(){
cout<<"OPENING DEPOSIT: ";
OPENINGDEPOSIT;
int OPENINGDEPOSIT1;
cin>>OPENINGDEPOSIT;
istringstream(OPENINGDEPOSIT)>>OPENINGDEPOSIT1;
int WHETHERNUMBER=ISVALUEENTEREDNUMERIC(OPENINGDEPOSIT);
if(WHETHERNUMBER==1){
    if(OPENINGDEPOSIT1>=1000){
        int ID = 1 + LASTID();
        saveFile(NEWUSERNAME, "USERS.txt", ID);
        saveFile(NEWPASSWORD, "PASSWORDS.txt", ID);
        saveFile(OPENINGDEPOSIT,"ACCOUNTBALANCE.txt",ID);
        cout<<endl<<"CONGRATULATIONS!! ACCOUNT OPENED SUCCESSFULLY"<<endl;
        MAININTERFACE();
    }
    else{
        cout<<endl<<"MINIMUM ACCOUNT OPENING BALANCE IS
RS:1000/-"<<endl;
        OPENANACCOUNTSTAGE2();
    }
}
else{
    cout<<endl<<"INVALID AMOUNT..."<<endl;
    int PROCEED;
    cout<<endl<<"WOULD YOU LIKE
TO"<<endl<<"1:CONTINUE"<<endl<<"2:EXIT"<<endl;

```

```

        cin>>PROCEED;
        if(PROCEED==1){
            OPENANACCOUNTSTAGE2();
        }else
        if(PROCEED==2){
            MAININTERFACE();
        }else{
            cout<<endl<<"INVALID COMMAND"<<endl;
            MAININTERFACE();
        };
    };
}

void DEPOSITAMOUNT(){
    int DEPOSIT=0;
    int BALANCE=0;
    string DEPOSIT1;
    cout<<endl<<"ENTER THE AMOUNT YOU WANT TO DEPOSIT:   ";
    cin>>DEPOSIT;
    stringstream ss;
    ss<<DEPOSIT;
    ss>>DEPOSIT1;
    int WHETHERANUMBER=ISVALUEENTEREDNUMERIC(DEPOSIT1);
    if(WHETHERANUMBER==1){
        string temporary;
        BALANCEINQUIRYSILENT();
        istringstream(BALANCEINQUIRYSILENT1)>>BALANCE;
        BALANCE = BALANCE + DEPOSIT;
        READREQUESTHANDLER(3);
    }
}

```

```

        stringstream ss;
        ss << BALANCE;
        ss>>temporary;
        ADUMPSITE[ID1-1]=temporary;
        WRITEREQUESTHANDLER(3,1);
        cout<<endl<<"TRANSACTION SUCCESSFULL"<<endl;
        POSTLOGININTERFACE();
    }
    else{
        cout<<endl<<"INVALID AMOUNT... "<<endl;
        int PROCEED;

        cout<<endl<<"WOULD YOU LIKE
TO"<<endl<<"1:CONTINUE"<<endl<<"2:EXIT"<<endl;

        cin>>PROCEED;
        if(PROCEED==1){
            DEPOSITAMOUNT();
        }else
        if(PROCEED==2){
            POSTLOGININTERFACE();
        }else{
            cout<<endl<<"INVALID COMMAND"<<endl;
            POSTLOGININTERFACE();
        };
    };
};

void WITHDRAWAMOUNT(){
    int WITHDRAWAL=0;
    int BALANCE=0;
    string WITHDRAWAL1;

```

```

cout<<endl<<"ENTER THE AMOUNT YOU WANT TO WITHDRAW: ";
cin>>WITHDRAWAL;
stringstream ss;
ss << WITHDRAWAL;
ss>>WITHDRAWAL1 ;
int WHETHERANUMBER=ISVALUEENTEREDNUMERIC(WITHDRAWAL1);
if(WHETHERANUMBER==1){
    BALANCEINQUIRYSILENT();
    istringstream(BALANCEINQUIRYSILENT1)>>BALANCE;
    BALANCE=BALANCE-WITHDRAWAL;
    if(BALANCE<0){
        cout<<endl<<"SORRY THIS WITHDRAWAL CAN,T BE PROCESSED
YOU HAVE INSUFFICIENT BALANCE"<<endl;
        BALANCE=BALANCE+WITHDRAWAL;
    }else
    if(WITHDRAWAL%500!=0&&WITHDRAWAL>0){
        cout<<"SORRY, AMOUNT SHOULD BE MULTIPLE OF 500";
        WITHDRAWAMOUNT();
    }else{
        cout<<endl<<"TRANSACTION SUCCESSFULL"<<endl;
        READREQUESTHANDLER(3);
        stringstream x;
        x << BALANCE;
        x>>ADUMPSITE[ID1-1];
        WRITEREQUESTHANDLER(3,1);
        POSTLOGININTERFACE();
    };
}
else{

```

```

        cout<<endl<<"INVALID AMOUNT"<<endl;

        int PROCEED;

        cout<<endl<<"WOULD YOU LIKE
TO"<<endl<<"1:CONTINUE"<<endl<<"2:EXIT"<<endl;

        cin>>PROCEED;

        if(PROCEED==1){

            WITHDRAWAMOUNT();

        }else

        if(PROCEED==2){

            POSTLOGININTERFACE();

        }else{

            cout<<endl<<"INVALID COMMAND"<<endl;

            POSTLOGININTERFACE();

        };

    };

}

void BALANCEENQUIRY(){

    READREQUESTHANDLER(3);

    cout<<endl<<"YOUR ACCOUNT BALANCE IS: "<<ADUMPSITE[ID1-1]<<endl;

    return;

}

void BALANCEINQUIRYSILENT(){

    READREQUESTHANDLER(3);

    BALANCEINQUIRYSILENT1=ADUMPSITE[ID1-1];

}

void LISTOFALLACCONTHOLDERS(){

    string STRING;

    ifstream MYFILE;

    string CURRENTCHARACTER;

```

```

long long ENCRYPTEDCHARACTER;
MYFILE.open("USERS.txt");
while(1){
    MYFILE >> CURRENTCHARACTER;
    if(CURRENTCHARACTER.find("#ID:") != string::npos){
        if("0" != STRING){
            cout<<STRING<<endl;
            STRING.erase(STRING.begin(), STRING.end());
        }
    }
    else{
        istream(CURRENTCHARACTER) >> ENCRYPTEDCHARACTER;
        STRING += (char)DECRYPTION(ENCRYPTEDCHARACTER);
        CURRENTCHARACTER = "";
    }
    if(MYFILE.peek() == EOF){
        MYFILE.close();
        cout<<"LIST ENDS HERE";
        return;
    };
}
}

void CLOSEORDELETEANACCOUNT(){
    cout<<endl<<"(ENTER 1 OR 2 TO PERFORM THE OPERATION)"<<endl<<"ARE YOU SURE YOU WANT TO DELETE YOUR ACCOUNT"<<endl<<"1:YES"<<endl<<"2:NO"<<endl;
    int RESPONSE;
    cin>>RESPONSE;
    if(RESPONSE==1){
        READREQUESTHANDLER(1);
    }
}

```

```

        CLOSEORDELETEANACCOUNTSTAGE2();
        WRITEREQUESTHANDLER(1,2);
        READREQUESTHANDLER(2);
        CLOSEORDELETEANACCOUNTSTAGE2();
        WRITEREQUESTHANDLER(2,2);
        READREQUESTHANDLER(3);
        CLOSEORDELETEANACCOUNTSTAGE2();
        WRITEREQUESTHANDLER(3,2);

        cout<<endl<<"YOUR REQUEST COMPLETED YOU WILL BE REFERRED
BACK TO MAIN MENU"<<endl;

        MAININTERFACE();
    }else
    {
        if(RESPONSE==2){
            cout<<endl<<"YOU HAVE CHOSEN TO NOT DELETE YOUR
ACCOUNT"<<endl;

            POSTLOGININTERFACE();
        }
        else{
            cout<<endl<<"INVALID COMMAND..."<<endl;
            CLOSEORDELETEANACCOUNT();
        }
    }

    void CLOSEORDELETEANACCOUNTSTAGE2(){
        int START=0;
        int START1=0;
        int END=0;
        int ACCOUNTSLEFT;
        START=ID1-1;
        START1=ID1;

```

```

END=LASTID()-1;
ACCOUNTSLEFT=END-START;
for(int LOOPS=0;LOOPS<ACCOUNTSLEFT;LOOPS++){
    ADUMPSITE[START]=ADUMPSITE[START1];
    START++;
    START1++;
};    cout<<"PROCESSING...";
}

void MODIFYDETAILSOFANACCOUNT(){
    cout<<endl<<"(ENTER 1 OR 2 TO PERFORM THE OPERATION)"<<endl<<"CHANGE
"<<endl<<"1 : USERNAME"<<endl<<"2 : PASSWORD"<<endl;
    int REQUEST;
    cin>>REQUEST;
    if(REQUEST==1){
        cout<<endl<<"ENTER NEW USERNAME: "<<endl;
        string CHANGEDUSERNAME;
        cin>>CHANGEDUSERNAME;
        READREQUESTHANDLER(1);
        ADUMPSITE[ID1-1]=CHANGEDUSERNAME;
        WRITEREQUESTHANDLER(1,1);
        cout<<endl<<"OPERATION COMPLETED SUCCESSFULLY"<<endl;
    }else
    if(REQUEST==2){
        cout<<endl<<"ENTER NEW PASSWORD: "<<endl;
        string CHANGEDPASSWORD;
        cin>>CHANGEDPASSWORD;
        READREQUESTHANDLER(2);
        ADUMPSITE[ID1-1]=CHANGEDPASSWORD;
        WRITEREQUESTHANDLER(2,1);
    }
}

```



```

        cout<<endl<<"OPERATION COMPLETED SUCCESSFULLY"<<endl;
    }else{
        cout<<"INVALID REQUEST";
    };

}

void WRITEREQUESTHANDLER(const int& GUIDE,const int& POSTDELETION){ // HERE
1 IS NO DELETION 2 IS POST DELETION INDICATOR
    if(GUIDE==1){
        WRITER("USERS.txt",POSTDELETION);
    }else
    if(GUIDE==2){
        WRITER("PASSWORDS.txt",POSTDELETION);
    }else
    if(GUIDE==3){
        WRITER("ACCOUNTBALANCE.txt",POSTDELETION);
    }
    else{
        cout<<endl<<"FAULT IN PROGRAM...WRH"<<endl;
    };
}

void WRITER(const char* FILENAME,const int& WRITETYPE){
    if(WRITETYPE==1){
        int END=0;
        END=LASTID();
        saveFileOverWrite(ADUMPSITE[0],FILENAME,1);
        for(int LOOPS=1;LOOPS<END;LOOPS++){
            int ID = 1 + LASTID();
            saveFile(ADUMPSITE[LOOPS],FILENAME,ID);
        }
    }
}

```

```

        }
    }else
    if(WRITETYPE==2){
        int END=0;
        END=LASTID()-1;
        saveFileOverWrite(ADUMPSITE[0],FILENAME,1);
        for(int LOOPS=1;LOOPS<END;LOOPS++){
            int ID = 1 + LASTID();
            saveFile(ADUMPSITE[LOOPS],FILENAME,ID);
        }
    }
    else{
        cout<<"ERROR ..... W";
        POSTLOGININTERFACE();
    }
}

void READREQUESTHANDLER(const int& ACCESS){
    if(ACCESS==1){
        READER("USERS.txt");
    }else
    if(ACCESS==2){
        READER("PASSWORDS.txt");
    }else
    if(ACCESS==3){
        READER("ACCOUNTBALANCE.txt");
    }
    else{cout<<endl<<"FAULT IN PROGRAM...RRH"<<endl;
    };
}

```

```

//THIS PROGRAM READS THE FILE AND FEEDS IT INTO ARRAY OF STRING
}

void READER(const char* FILENAME){
    int X=0;
    string STRING;
    ifstream MYFILE;
    string CURRENTCHARACTER;
    long long ENCRYPTEDCHARACTER;
    MYFILE.open(FILENAME);
    while(1){
        MYFILE >> CURRENTCHARACTER;
        if(CURRENTCHARACTER.find("#ID:") != string::npos){
            if("0" != STRING){
                ADUMPSITE[X]=STRING;
                STRING.erase(STRING.begin(), STRING.end());
                X++;
            }
        }
        else{
            istringstream(CURRENTCHARACTER) >> ENCRYPTEDCHARACTER;
            STRING += (char)DECRYPTION(ENCRYPTEDCHARACTER);
            CURRENTCHARACTER = "";
        }
        if(MYFILE.peek() == EOF){
            MYFILE.close();
            return;
        }
    }
}

```

```

    }
    bool ISVALUEENTEREDNUMERIC(string OPENINGDEPOSIT){
    for(int loops=0;loops<OPENINGDEPOSIT.length();loops++){
        if(isdigit(OPENINGDEPOSIT[loops])==false){
            return false;
        }
        else{
            return true;
        }
    }
}

int LASTID(){
    ifstream MYFILE;
    MYFILE.open("USERS.txt");
    MYFILE.seekg(0, ios::end);
    if(MYFILE.tellg() == -1)
        return 0;

    string ASTRING;
    for(int LOOPS = -1; ASTRING.find("#") == string::npos; LOOPS--){
        MYFILE.seekg(LOOPS, ios::end);
        MYFILE >> ASTRING;
    }
    MYFILE.close();
    ASTRING.erase(0, 4);
    int ID;
    istringstream(ASRING) >> ID;
    return ID;
}

```

```

int CHECKID(string attempt, const char* FILENAME){
    string STRING;
    ifstream MYFILE;
    string CURRENTCHARACTER;
    long long ENCRYPTEDCHARACTER;
    MYFILE.open(FILENAME);
    while(1){
        MYFILE >> CURRENTCHARACTER;
        if(CURRENTCHARACTER.find("#ID:") != string::npos){
            if(attempt == STRING){
                MYFILE.close();
                CURRENTCHARACTER.erase(0, 4);
                int ID;
                istringstream(CURRENTCHARACTER) >> ID;
                return ID;
            }
            else{
                STRING.erase(STRING.begin(), STRING.end());
            }
        }
        else{
            istringstream(CURRENTCHARACTER) >> ENCRYPTEDCHARACTER;
            STRING += (char)DECRYPTION(ENCRYPTEDCHARACTER);
            CURRENTCHARACTER = "";
        }
        if(MYFILE.peek() == EOF){
            MYFILE.close();
            return 0;
        }
    }
}

```

```

    }
}

void saveFile(string STRINGTOSAVE, const char* FILENAME, const int& ID){
    fstream MYFILE;
    MYFILE.open(FILENAME, ios::app);
    MYFILE.seekg(0, ios::end);
    if(MYFILE.tellg() != 0)
        MYFILE << endl;
    MYFILE.seekg(0, ios::beg);
    for(int LOOPS = 0; LOOPS < STRINGTOSAVE.length(); LOOPS++){
        MYFILE << ENCRYPTION(STRINGTOSAVE[LOOPS]);
        MYFILE << endl;
    }
    MYFILE << "#ID:" << ID;
    MYFILE.close();
}

```

```

void saveFileOverWrite(string STRINGTOSAVE, const char* FILENAME, const int& ID){
    fstream MYFILE;
    MYFILE.open(FILENAME, ios::out);
    MYFILE.seekg(0, ios::end);
    if(MYFILE.tellg() != 0)
        MYFILE << endl;
    MYFILE.seekg(0, ios::beg);
    for(int LOOPS = 0; LOOPS < STRINGTOSAVE.length(); LOOPS++){
        MYFILE << ENCRYPTION(STRINGTOSAVE[LOOPS]);
        MYFILE << endl;
    }
}

```

```

        MYFILE << "#ID:" << ID;

        MYFILE.close();
    }

    long long ENCRYPTION(int CHARACTER){
        return powf(CHARACTER, 6) * 7 - 8;
    }

    int DECRYPTION(long long CHARACTER){
        return powf((CHARACTER + 8) / 7, 1/6.f);
    }

    void EXIT(){
        cout<<endl<<"EXITING....."<<endl<<"HAVE A NICE DAY"<<endl;
    }

    void MAININTERFACE(){
        int COMMAND;

        cout<<endl<<"WELCOME TO BANKING SYSTEM ORGANIZER"<<endl<<"(ENTER THE
NUMBER CORRESPONDING TO THE FUNCTION YOU WANT TO PERFORM)"<<endl;

        cout<<"1:LOGIN"<<endl<<"2:CREATE AN ACCOUNT"<<endl<<"3.EXIT"<<endl;

        cin>>COMMAND;

        if(COMMAND==1){

            LOGIN();

        }else
        if(COMMAND==2){
            OPENANACCOUNTSTAGE1();
        }else
        if(COMMAND==3){
            EXIT();
        }
        else{

```

```

        cout<<endl<<"INVALID COMMAND"<<endl;

        MAININTERFACE();

    }

}

void POSTLOGININTERFACE(){

    int COMMAND=0;

    cout<<endl<<"(ENTER THE NUMBER CORRESPONDING TO THE FUNCTION YOU
WANT TO PERFORM)"<<endl;

    cout<<"1:CHECK ACCOUNT BALANCE"<<endl;
    cout<<"2:DEPOSIT AMOUNT"<<endl;
    cout<<"3:WITHDRAW AMOUNT"<<endl;
    cout<<"4:CHANGE ACCOUNT DETAILS"<<endl;
    cout<<"5:LIST OF ALL ACCOUNT HOLDERS IN THE BANK"<<endl;
    cout<<"6:DELETE YOUR ACCOUNT"<<endl;
    cout<<"7:SIGNOUT"<<endl;

    cin>>COMMAND;

    if(COMMAND==1){

        BALANCEENQUIRY();

        POSTLOGININTERFACE();

    }else

    if(COMMAND==2){

        DEPOSITAMOUNT();

    }else

    if(COMMAND==3){

        WITHDRAWAMOUNT();

    }else

    if(COMMAND==4){

        MODIFYDETAILSOFANACCOUNT();

        POSTLOGININTERFACE();

```



```

        }else
        if(COMMAND==5){
            LISTOFALLACCOUNTHOLDERS();
            POSTLOGININTERFACE();
        }else
        if(COMMAND==6){
            CLOSEORDELETEANACCOUNT();
        }else
        if(COMMAND==7){
            MAININTERFACE();
        }
        else{
            cout<<endl<<"INVALID COMMAND"<<endl;
            POSTLOGININTERFACE();
        }
    }

private:
    string ADUMPSITE[100]={""};
    string ID;
    int ID1;
    string TOCHANGE;
    string OPENINGDEPOSIT;
    string BALANCEINQUIRYSILENT1;
    string NEWUSERNAME;
    string NEWPASSWORD;
    string USERNAMEATTEMPT;
    string PASSWORDATTEMPT;
    bool ACCESS;

```

```
};  
int main(){  
    MANAGER MANAGEROBJ;  
    MANAGEROBJ.MAININTERFACE();  
}
```