Algorithms Level 4

PROGRAMMING ADVICES

LEARN THE RIGHT WAY

26+ Years of Experience

Mohammed Abu-Hadhoud

MBA, PMOC, PgMP®, PMP®, PMI-RMP®, CM, ITILF, MCPD, MCSD

ProgrammingAdvices.com

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>

using namespace std;

struct stUser
{
    string UserName;
    string Password;
    int Permissions;
    bool MarkForDelete = false;
};

enum enTransactionsMenueOptions { eDeposit = 1, eWithdraw = 2,
eShowTotalBalance = 3, eShowMainMenue = 4 };

enum enManageUsersMenueOptions {
    eListUsers = 1, eAddNewUser = 2, eDeleteUser = 3,
    eUpdateUser = 4, eFindUser = 5, eMainMenue = 6
};

enum enMainMenueOptions {
    eListClients = 1, eAddNewClient = 2, eDeleteClient = 3,
    eUpdateClient = 4, eFindClient = 5, eShowTransactionsMenue =
6,
    eManageUsers = 7, eExit = 8
};

enum enMainMenuePermissions {
    eAll = -1, pListClients = 1, pAddNewClient = 2, pDeleteClient
= 4,
    pUpdateClients = 8, pFindClient = 16, pTranactions = 32,
pManageUsers = 64
};

const string ClientsFileName = "Clients.txt";
const string UsersFileName = "Users.txt";

stUser CurrentUser;

void ShowMainMenue();
void ShowTransactionsMenue();
void ShowManageUsersMenue();
bool CheckAccessPermission(enMainMenuePermissions Permission);
void Login();
```

```
struct sClient
{
    string AccountNumber;
    string PinCode;
    string Name;
    string Phone;
    double AccountBalance;
    bool MarkForDelete = false;
};

vector<string> SplitString(string S1, string Delim)
{

    vector<string> vString;

    short pos = 0;
    string sWord; // define a string variable

    // use find() function to get the position of the delimiters
    while ((pos = S1.find(Delim)) != std::string::npos)
    {
        sWord = S1.substr(0, pos); // store the word
        if (sWord != "")
        {
            vString.push_back(sWord);
        }

        S1.erase(0, pos + Delim.length());  /* erase() until
positon and move to next word. */
    }

    if (S1 != "")
    {
        vString.push_back(S1); // it adds last word of the string.
    }

    return vString;

}
```

```
stUser ConvertUserLinetoRecord(string Line, string Seperator =
"#//#")
{

    stUser User;
    vector<string> vUserData;

    vUserData = SplitString(Line, Seperator);

    User.UserName = vUserData[0];
    User.Password = vUserData[1];
    User.Permissions = stoi(vUserData[2]);

    return User;

}

sClient ConvertLinetoRecord(string Line, string Seperator =
"#//#")
{

    sClient Client;
    vector<string> vClientData;

    vClientData = SplitString(Line, Seperator);

    Client.AccountNumber = vClientData[0];
    Client.PinCode = vClientData[1];
    Client.Name = vClientData[2];
    Client.Phone = vClientData[3];
    Client.AccountBalance = stod(vClientData[4]);//cast string to
double


    return Client;

}
```

```
stUser ConvertUserLinetoRecord2(string Line, string Seperator =
"#//#")
{
    stUser User;
    vector<string> vUserData;

    vUserData = SplitString(Line, Seperator);

    User.UserName = vUserData[0];
    User.Password = vUserData[1];
    User.Permissions = stoi(vUserData[2]);

    return User;

}

string ConvertRecordToLine(sClient Client, string Seperator =
"#//#")
{

    string stClientRecord = "";

    stClientRecord += Client.AccountNumber + Seperator;
    stClientRecord += Client.PinCode + Seperator;
    stClientRecord += Client.Name + Seperator;
    stClientRecord += Client.Phone + Seperator;
    stClientRecord += to_string(Client.AccountBalance);

    return stClientRecord;

}

string ConvertUserRecordToLine(stUser User, string Seperator =
"#//#")
{

    string stClientRecord = "";

    stClientRecord += User.UserName + Seperator;
    stClientRecord += User.Password + Seperator;
    stClientRecord += to_string(User.Permissions);

    return stClientRecord;

}
```

```
bool ClientExistsByAccountNumber(string AccountNumber, string
FileName)
{

    vector <sClient> vClients;

    fstream MyFile;
    MyFile.open(FileName, ios::in);//read Mode

    if (MyFile.is_open())
    {

        string Line;
        sClient Client;

        while (getline(MyFile, Line))
        {

            Client = ConvertLinetoRecord(Line);
            if (Client.AccountNumber == AccountNumber)
            {
                MyFile.close();
                return true;
            }


            vClients.push_back(Client);
        }

        MyFile.close();

    }

    return false;


}
```

```
bool UserExistsByUsername(string Username, string FileName)
{


    fstream MyFile;
    MyFile.open(FileName, ios::in);//read Mode

    if (MyFile.is_open())
    {

        string Line;
        stUser User;

        while (getline(MyFile, Line))
        {

            User = ConvertUserLinetoRecord(Line);
            if (User.UserName == Username)
            {
                MyFile.close();
                return true;
            }

        }

        MyFile.close();

    }

    return false;


}
```

```
sClient ReadNewClient()
{
    sClient Client;

    cout << "Enter Account Number? ";

    // Usage of std::ws will extract allthe whitespace character
    getline(cin >> ws, Client.AccountNumber);

    while (ClientExistsByAccountNumber(Client.AccountNumber, 
ClientsFileName))
    {
        cout << "\nClient with [" << Client.AccountNumber << "] 
already exists, Enter another Account Number? ";
        getline(cin >> ws, Client.AccountNumber);
    }


    cout << "Enter PinCode? ";
    getline(cin, Client.PinCode);

    cout << "Enter Name? ";
    getline(cin, Client.Name);

    cout << "Enter Phone? ";
    getline(cin, Client.Phone);

    cout << "Enter AccountBalance? ";
    cin >> Client.AccountBalance;

    return Client;

}

int ReadPermissionsToSet()
{

    int Permissions = 0;
    char Answer = 'n';


    cout << "\nDo you want to give full access? y/n? ";
    cin >> Answer;
    if (Answer == 'y' || Answer == 'Y')
    {
        return -1;
    }
```

```
cout << "\nDo you want to give access to : \n ";

cout << "\nShow Client List? y/n? ";
cin >> Answer;
if (Answer == 'y' || Answer == 'Y')
{


    Permissions += enMainMenuePermissions::pListClients;
}


cout << "\nAdd New Client? y/n? ";
cin >> Answer;
if (Answer == 'y' || Answer == 'Y')
{
    Permissions += enMainMenuePermissions::pAddNewClient;
}

cout << "\nDelete Client? y/n? ";
cin >> Answer;
if (Answer == 'y' || Answer == 'Y')
{
    Permissions += enMainMenuePermissions::pDeleteClient;
}

cout << "\nUpdate Client? y/n? ";
cin >> Answer;
if (Answer == 'y' || Answer == 'Y')
{
    Permissions += enMainMenuePermissions::pUpdateClients;
}

cout << "\nFind Client? y/n? ";
cin >> Answer;
if (Answer == 'y' || Answer == 'Y')
{
    Permissions += enMainMenuePermissions::pFindClient;
}

cout << "\nTransactions? y/n? ";
cin >> Answer;
if (Answer == 'y' || Answer == 'Y')
{
    Permissions += enMainMenuePermissions::pTranactions;
}
```

```
    cout << "\nManage Users? y/n? ";
    cin >> Answer;
    if (Answer == 'y' || Answer == 'Y')
    {
        Permissions += enMainMenuePermissions::pManageUsers;
    }


    return Permissions;

}

stUser ReadNewUser()
{
    stUser User;

    cout << "Enter Username? ";

    // Usage of std::ws will extract allthe whitespace character
    getline(cin >> ws, User.UserName);

    while (UserExistsByUsername(User.UserName, UsersFileName))
    {
        cout << "\nUser with [" << User.UserName << "] already
exists, Enter another Username? ";
        getline(cin >> ws, User.UserName);
    }

    cout << "Enter Password? ";
    getline(cin, User.Password);

    User.Permissions = ReadPermissionsToSet();

    return User;

}
```

```
vector <stUser> LoadUsersDataFromFile(string FileName)
{

    vector <stUser> vUsers;

    fstream MyFile;
    MyFile.open(FileName, ios::in);//read Mode

    if (MyFile.is_open())
    {

        string Line;
        stUser User;

        while (getline(MyFile, Line))
        {

            User = ConvertUserLinetoRecord(Line);

            vUsers.push_back(User);
        }

        MyFile.close();

    }

    return vUsers;

}
```

```
vector <sClient> LoadCleintsDataFromFile(string FileName)
{

    vector <sClient> vClients;

    fstream MyFile;
    MyFile.open(FileName, ios::in);//read Mode

    if (MyFile.is_open())
    {

        string Line;
        sClient Client;

        while (getline(MyFile, Line))
        {

            Client = ConvertLinetoRecord(Line);

            vClients.push_back(Client);
        }

        MyFile.close();

    }

    return vClients;

}

void PrintClientRecordLine(sClient Client)
{

    cout << "| " << setw(15) << left << Client.AccountNumber;
    cout << "| " << setw(10) << left << Client.PinCode;
    cout << "| " << setw(40) << left << Client.Name;
    cout << "| " << setw(12) << left << Client.Phone;
    cout << "| " << setw(12) << left << Client.AccountBalance;

}
```

```
void PrintUserRecordLine(stUser User)
{

    cout << "| " << setw(15) << left << User.UserName;
    cout << "| " << setw(10) << left << User.Password;
    cout << "| " << setw(40) << left << User.Permissions;
}

void PrintClientRecordBalanceLine(sClient Client)
{

    cout << "| " << setw(15) << left << Client.AccountNumber;
    cout << "| " << setw(40) << left << Client.Name;
    cout << "| " << setw(12) << left << Client.AccountBalance;

}

void ShowAccessDeniedMessage()
{
    cout << "\n------------------------------------\n";
    cout << "Access Denied, \nYou dont Have Permission To Do
this,\nPlease Conact Your Admin.";
    cout << "\n------------------------------------\n";
}
```

```
void ShowAllClientsScreen()
{


    if
(!CheckAccessPermission(enMainMenuePermissions::pListClients))
    {
        ShowAccessDeniedMessage();
        return;
    }

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);

    cout << "\n\t\t\t\t\tClient List (" << vClients.size() << ")
Client(s).";
    cout <<
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

    cout << "| " << left << setw(15) << "Accout Number";
    cout << "| " << left << setw(10) << "Pin Code";
    cout << "| " << left << setw(40) << "Client Name";
    cout << "| " << left << setw(12) << "Phone";
    cout << "| " << left << setw(12) << "Balance";
    cout <<
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

    if (vClients.size() == 0)
        cout << "\t\t\t\tNo Clients Available In the System!";
    else

        for (sClient Client : vClients)
        {

            PrintClientRecordLine(Client);
            cout << endl;
        }

    cout <<
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

}
```

```
void ShowAllUsersScreen()
{


    vector <stUser> vUsers = LoadUsersDataFromFile(UsersFileName);

    cout << "\n\t\t\t\t\tUsers List (" << vUsers.size() << ") 
User(s).";
    cout << 
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

    cout << "| " << left << setw(15) << "User Name";
    cout << "| " << left << setw(10) << "Password";
    cout << "| " << left << setw(40) << "Permissions";
    cout << 
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

    if (vUsers.size() == 0)
        cout << "\t\t\t\tNo Users Available In the System!";
    else

        for (stUser User : vUsers)
        {

            PrintUserRecordLine(User);
            cout << endl;
        }

    cout << 
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

}
```

```
void ShowTotalBalances()
{

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);

    cout << "\n\t\t\t\t\tBalances List (" << vClients.size() << ")
Client(s).";
    cout <<
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

    cout << "| " << left << setw(15) << "Accout Number";
    cout << "| " << left << setw(40) << "Client Name";
    cout << "| " << left << setw(12) << "Balance";
    cout <<
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;

    double TotalBalances = 0;

    if (vClients.size() == 0)
        cout << "\t\t\t\tNo Clients Available In the System!";
    else

        for (sClient Client : vClients)
        {

            PrintClientRecordBalanceLine(Client);
            TotalBalances += Client.AccountBalance;

            cout << endl;
        }

    cout <<
"\n_______________________________________________________";
    cout << "_________________________________________\n" << endl;
    cout << "\t\t\t\t\t   Total Balances = " << TotalBalances;

}
```

```
void PrintClientCard(sClient Client)
{
    cout << "\nThe following are the client details:\n";
    cout << "-----------------------------------";
    cout << "\nAccout Number: " << Client.AccountNumber;
    cout << "\nPin Code     : " << Client.PinCode;
    cout << "\nName         : " << Client.Name;
    cout << "\nPhone        : " << Client.Phone;
    cout << "\nAccount Balance: " << Client.AccountBalance;
    cout << "\n-----------------------------------\n";

}

void PrintUserCard(stUser User)
{
    cout << "\nThe following are the user details:\n";
    cout << "-----------------------------------";
    cout << "\nUsername    : " << User.UserName;
    cout << "\nPassword    : " << User.Password;
    cout << "\nPermissions : " << User.Permissions;
    cout << "\n-----------------------------------\n";

}

bool FindClientByAccountNumber(string AccountNumber, vector
<sClient> vClients, sClient& Client)
{

    for (sClient C : vClients)
    {

        if (C.AccountNumber == AccountNumber)
        {
            Client = C;
            return true;
        }

    }
    return false;

}
```

```
bool FindUserByUsername(string Username, vector <stUser> vUsers,
stUser& User)
{

    for (stUser U : vUsers)
    {

        if (U.UserName == Username)
        {
            User = U;
            return true;
        }

    }
    return false;

}

bool FindUserByUsernameAndPassword(string Username, string
Password, stUser& User)
{

    vector <stUser> vUsers = LoadUsersDataFromFile(UsersFileName);

    for (stUser U : vUsers)
    {

        if (U.UserName == Username && U.Password == Password)
        {
            User = U;
            return true;
        }

    }
    return false;

}
```

```
sClient ChangeClientRecord(string AccountNumber)
{
    sClient Client;

    Client.AccountNumber = AccountNumber;

    cout << "\n\nEnter PinCode? ";
    getline(cin >> ws, Client.PinCode);

    cout << "Enter Name? ";
    getline(cin, Client.Name);

    cout << "Enter Phone? ";
    getline(cin, Client.Phone);

    cout << "Enter AccountBalance? ";
    cin >> Client.AccountBalance;

    return Client;

}

stUser ChangeUserRecord(string Username)
{
    stUser User;

    User.UserName = Username;

    cout << "\n\nEnter Password? ";
    getline(cin >> ws, User.Password);

    User.Permissions = ReadPermissionsToSet();

    return User;

}
```

```
bool MarkClientForDeleteByAccountNumber(string AccountNumber,
vector <sClient>& vClients)
{
    for (sClient& C : vClients)
    {
        if (C.AccountNumber == AccountNumber)
        {
            C.MarkForDelete = true;
            return true;
        }
    }
    return false;
}

bool MarkUserForDeleteByUsername(string Username, vector <stUser>&
vUsers)
{
    for (stUser& U : vUsers)
    {
        if (U.UserName == Username)
        {
            U.MarkForDelete = true;
            return true;
        }
    }
    return false;
}
```

```
vector <sClient> SaveCleintsDataToFile(string FileName, vector
<sClient> vClients)
{

    fstream MyFile;
    MyFile.open(FileName, ios::out);//overwrite

    string DataLine;

    if (MyFile.is_open())
    {

        for (sClient C : vClients)
        {

            if (C.MarkForDelete == false)
            {
                //we only write records that are not marked for
delete.
                DataLine = ConvertRecordToLine(C);
                MyFile << DataLine << endl;

            }

        }

        MyFile.close();

    }

    return vClients;

}
```

```
vector <stUser> SaveUsersDataToFile(string FileName, vector
<stUser> vUsers)
{

    fstream MyFile;
    MyFile.open(FileName, ios::out);//overwrite

    string DataLine;

    if (MyFile.is_open())
    {

        for (stUser U : vUsers)
        {

            if (U.MarkForDelete == false)
            {
                //we only write records that are not marked for
delete.
                DataLine = ConvertUserRecordToLine(U);
                MyFile << DataLine << endl;

            }

        }

        MyFile.close();

    }

    return vUsers;

}
```

```
void AddDataLineToFile(string FileName, string  stDataLine)
{
    fstream MyFile;
    MyFile.open(FileName, ios::out | ios::app);

    if (MyFile.is_open())
    {

        MyFile << stDataLine << endl;

        MyFile.close();
    }

}

void AddNewClient()
{
    sClient Client;
    Client = ReadNewClient();
    AddDataLineToFile(ClientsFileName,
ConvertRecordToLine(Client));

}

void AddNewUser()
{
    stUser User;
    User = ReadNewUser();
    AddDataLineToFile(UsersFileName,
ConvertUserRecordToLine(User));

}
```

```
void AddNewClients()
{
    char AddMore = 'Y';
    do
    {
        //system("cls");
        cout << "Adding New Client:\n\n";

        AddNewClient();
        cout << "\nClient Added Successfully, do you want to add
more clients? Y/N? ";


        cin >> AddMore;

    } while (toupper(AddMore) == 'Y');

}

void AddNewUsers()
{
    char AddMore = 'Y';
    do
    {
        //system("cls");
        cout << "Adding New User:\n\n";

        AddNewUser();
        cout << "\nUser Added Successfully, do you want to add
more Users? Y/N? ";


        cin >> AddMore;

    } while (toupper(AddMore) == 'Y');

}
```

```
bool DeleteClientByAccountNumber(string AccountNumber, vector
<sClient>& vClients)
{



    sClient Client;
    char Answer = 'n';

    if (FindClientByAccountNumber(AccountNumber, vClients,
Client))
    {

        PrintClientCard(Client);

        cout << "\n\nAre you sure you want delete this client? y/n
? ";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
        {
            MarkClientForDeleteByAccountNumber(AccountNumber,
vClients);
            SaveCleintsDataToFile(ClientsFileName, vClients);

            //Refresh Clients
            vClients = LoadCleintsDataFromFile(ClientsFileName);

            cout << "\n\nClient Deleted Successfully.";
            return true;
        }

    }
    else
    {
        cout << "\nClient with Account Number (" << AccountNumber
<< ") is Not Found!";
        return false;
    }

}
```

```
bool DeleteUserByUsername(string Username, vector <stUser>&
vUsers)
{

    if (Username == "Admin")
    {
        cout << "\n\nYou cannot Delete This User.";
        return false;

    }

    stUser User;
    char Answer = 'n';

    if (FindUserByUsername(Username, vUsers, User))
    {

        PrintUserCard(User);

     cout << "\n\nAre you sure you want delete this User? y/n ? ";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
        {

            MarkUserForDeleteByUsername(Username, vUsers);
            SaveUsersDataToFile(UsersFileName, vUsers);

            //Refresh Clients
            vUsers = LoadUsersDataFromFile(UsersFileName);

            cout << "\n\nUser Deleted Successfully.";
            return true;
        }

    }
    else
    {
        cout << "\nUser with Username (" << Username << ") is Not
Found!";
        return false;
    }

}
```

```
bool UpdateClientByAccountNumber(string AccountNumber, vector
<sClient>& vClients)
{

    sClient Client;
    char Answer = 'n';

    if (FindClientByAccountNumber(AccountNumber, vClients,
Client))
    {

        PrintClientCard(Client);
        cout << "\n\nAre you sure you want update this client? y/n
? ";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
        {

            for (sClient& C : vClients)
            {
                if (C.AccountNumber == AccountNumber)
                {
                    C = ChangeClientRecord(AccountNumber);
                    break;
                }

            }

            SaveCleintsDataToFile(ClientsFileName, vClients);

            cout << "\n\nClient Updated Successfully.";
            return true;
        }

    }
    else
    {
        cout << "\nClient with Account Number (" << AccountNumber
<< ") is Not Found!";
        return false;
    }

}
```

```
bool UpdateUserByUsername(string Username, vector <stUser>&
vUsers)
{

    stUser User;
    char Answer = 'n';

    if (FindUserByUsername(Username, vUsers, User))
    {

        PrintUserCard(User);
        cout << "\n\nAre you sure you want update this User? y/n ?
";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
        {

            for (stUser& U : vUsers)
            {
                if (U.UserName == Username)
                {
                    U = ChangeUserRecord(Username);
                    break;
                }

            }

            SaveUsersDataToFile(UsersFileName, vUsers);

            cout << "\n\nUser Updated Successfully.";
            return true;
        }

    }
    else
    {
        cout << "\nUser with Account Number (" << Username << ")
is Not Found!";
        return false;
    }

}
```

```
bool DepositBalanceToClientByAccountNumber(string AccountNumber,
double Amount, vector <sClient>& vClients)
{
    char Answer = 'n';
    cout << "\n\nAre you sure you want perfrom this transaction?
y/n ? ";
    cin >> Answer;
    if (Answer == 'y' || Answer == 'Y')
    {

        for (sClient& C : vClients)
        {
            if (C.AccountNumber == AccountNumber)
            {
                C.AccountBalance += Amount;
                SaveCleintsDataToFile(ClientsFileName, vClients);
                cout << "\n\nDone Successfully. New balance is: "
<< C.AccountBalance;

                return true;
            }

        }


        return false;
    }

}

string ReadClientAccountNumber()
{
    string AccountNumber = "";

    cout << "\nPlease enter AccountNumber? ";
    cin >> AccountNumber;
    return AccountNumber;

}
```

```
string ReadUserName()
{
    string Username = "";

    cout << "\nPlease enter Username? ";
    cin >> Username;
    return Username;

}

void ShowListUsersScreen()
{

    ShowAllUsersScreen();

}

void ShowAddNewUserScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tAdd New User Screen";
    cout << "\n-----------------------------------\n";

    AddNewUsers();


}

void ShowDeleteUserScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tDelete Users Screen";
    cout << "\n-----------------------------------\n";

    vector <stUser> vUsers = LoadUsersDataFromFile(UsersFileName);

    string Username = ReadUserName();
    DeleteUserByUsername(Username, vUsers);

}
```

```
void ShowUpdateUserScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tUpdate Users Screen";
    cout << "\n-----------------------------------\n";

    vector <stUser> vUsers = LoadUsersDataFromFile(UsersFileName);
    string Username = ReadUserName();

    UpdateUserByUsername(Username, vUsers);
}

void ShowDeleteClientScreen()
{


    if
(!CheckAccessPermission(enMainMenuePermissions::pDeleteClient))
    {
        ShowAccessDeniedMessage();
        return;
    }

    cout << "\n-----------------------------------\n";
    cout << "\tDelete Client Screen";
    cout << "\n-----------------------------------\n";
    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    string AccountNumber = ReadClientAccountNumber();
    DeleteClientByAccountNumber(AccountNumber, vClients);

}

void ShowUpdateClientScreen()
{
    if
(!CheckAccessPermission(enMainMenuePermissions::pUpdateClients))
    {
        ShowAccessDeniedMessage();
        return;
    }

    cout << "\n-----------------------------------\n";
    cout << "\tUpdate Client Info Screen";
    cout << "\n-----------------------------------\n";
    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    string AccountNumber = ReadClientAccountNumber();
    UpdateClientByAccountNumber(AccountNumber, vClients);

}

void ShowAddNewClientsScreen()
{

    if
(!CheckAccessPermission(enMainMenuePermissions::pUpdateClients))
    {
        ShowAccessDeniedMessage();
        return;
    }

    cout << "\n-----------------------------------\n";
    cout << "\tAdd New Clients Screen";
    cout << "\n-----------------------------------\n";

    AddNewClients();

}
```

```
void ShowFindClientScreen()
{


    if
(!CheckAccessPermission(enMainMenuePermissions::pFindClient))
    {
        ShowAccessDeniedMessage();
        return;
    }

    cout << "\n-----------------------------------\n";
    cout << "\tFind Client Screen";
    cout << "\n-----------------------------------\n";

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    sClient Client;
    string AccountNumber = ReadClientAccountNumber();
    if (FindClientByAccountNumber(AccountNumber, vClients,
Client))
        PrintClientCard(Client);
    else
        cout << "\nClient with Account Number[" << AccountNumber
<< "] is not found!";

}

void ShowFindUserScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tFind User Screen";
    cout << "\n-----------------------------------\n";

    vector <stUser> vUsers = LoadUsersDataFromFile(UsersFileName);
    stUser User;
    string Username = ReadUserName();
    if (FindUserByUsername(Username, vUsers, User))
        PrintUserCard(User);
    else
        cout << "\nUser with Username [" << Username << "] is not
found!";

}
```

```
void ShowEndScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tProgram Ends :-)";
    cout << "\n-----------------------------------\n";

}

void ShowDepositScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tDeposit Screen";
    cout << "\n-----------------------------------\n";


    sClient Client;

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    string AccountNumber = ReadClientAccountNumber();


    while (!FindClientByAccountNumber(AccountNumber, vClients,
Client))
    {
        cout << "\nClient with [" << AccountNumber << "] does not
exist.\n";
        AccountNumber = ReadClientAccountNumber();
    }


    PrintClientCard(Client);

    double Amount = 0;
    cout << "\nPlease enter deposit amount? ";
    cin >> Amount;

    DepositBalanceToClientByAccountNumber(AccountNumber, Amount,
vClients);

}
```

```
void ShowWithDrawScreen()
{
    cout << "\n-----------------------------------\n";
    cout << "\tWithdraw Screen";
    cout << "\n-----------------------------------\n";

    sClient Client;

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    string AccountNumber = ReadClientAccountNumber();


    while (!FindClientByAccountNumber(AccountNumber, vClients,
Client))
    {
        cout << "\nClient with [" << AccountNumber << "] does not
exist.\n";
        AccountNumber = ReadClientAccountNumber();
    }

    PrintClientCard(Client);

    double Amount = 0;
    cout << "\nPlease enter withdraw amount? ";
    cin >> Amount;

    //Validate that the amount does not exceeds the balance
    while (Amount > Client.AccountBalance)
    {
        cout << "\nAmount Exceeds the balance, you can withdraw up
to : " << Client.AccountBalance << endl;
        cout << "Please enter another amount? ";
        cin >> Amount;
    }

    DepositBalanceToClientByAccountNumber(AccountNumber, Amount *
-1, vClients);

}

void ShowTotalBalancesScreen()
{

    ShowTotalBalances();

}
```

```
bool CheckAccessPermission(enMainMenuePermissions Permission)
{
    if (CurrentUser.Permissions == enMainMenuePermissions::eAll)
        return true;

    if ((Permission & CurrentUser.Permissions) == Permission)
        return true;
    else
        return false;

}

void GoBackToMainMenue()
{
    cout << "\n\nPress any key to go back to Main Menue...";
    system("pause>0");
    ShowMainMenue();
}

void GoBackToTransactionsMenue()
{
    cout << "\n\nPress any key to go back to Transactions 
Menue...";
    system("pause>0");
    ShowTransactionsMenue();

}

void GoBackToManageUsersMenue()
{
    cout << "\n\nPress any key to go back to Transactions 
Menue...";
    system("pause>0");
    ShowManageUsersMenue();

}

short ReadTransactionsMenueOption()
{
    cout << "Choose what do you want to do? [1 to 4]? ";
    short Choice = 0;
    cin >> Choice;

    return Choice;
}
```

```
void PerfromTranactionsMenueOption(enTransactionsMenueOptions 
TransactionMenueOption)
{
    switch (TransactionMenueOption)
    {
    case enTransactionsMenueOptions::eDeposit:
    {
        system("cls");
        ShowDepositScreen();
        GoBackToTransactionsMenue();
        break;
    }

    case enTransactionsMenueOptions::eWithdraw:
    {
        system("cls");
        ShowWithDrawScreen();
        GoBackToTransactionsMenue();
        break;
    }


    case enTransactionsMenueOptions::eShowTotalBalance:
    {
        system("cls");
        ShowTotalBalancesScreen();
        GoBackToTransactionsMenue();
        break;
    }


    case enTransactionsMenueOptions::eShowMainMenue:
    {

        ShowMainMenue();

    }
    }

}
```

```
void ShowTransactionsMenue()
{

    if
(!CheckAccessPermission(enMainMenuePermissions::pTranactions))
    {
        ShowAccessDeniedMessage();
        GoBackToMainMenue();
        return;
    }

    system("cls");
    cout << "===========================================\n";
    cout << "\t\tTransactions Menue Screen\n";
    cout << "===========================================\n";
    cout << "\t[1] Deposit.\n";
    cout << "\t[2] Withdraw.\n";
    cout << "\t[3] Total Balances.\n";
    cout << "\t[4] Main Menue.\n";
    cout << "===========================================\n";

PerfromTranactionsMenueOption((enTransactionsMenueOptions)ReadTran
sactionsMenueOption());
}

short ReadMainMenueOption()
{
    cout << "Choose what do you want to do? [1 to 8]? ";
    short Choice = 0;
    cin >> Choice;

    return Choice;
}
```

```
void PerfromManageUsersMenueOption(enManageUsersMenueOptions
ManageUsersMenueOption)
{
    switch (ManageUsersMenueOption)
    {
    case enManageUsersMenueOptions::eListUsers:
    {
        system("cls");
        ShowListUsersScreen();
        GoBackToManageUsersMenue();
        break;
    }

    case enManageUsersMenueOptions::eAddNewUser:
    {
        system("cls");
        ShowAddNewUserScreen();
        GoBackToManageUsersMenue();
        break;
    }

    case enManageUsersMenueOptions::eDeleteUser:
    {
        system("cls");
        ShowDeleteUserScreen();
        GoBackToManageUsersMenue();
        break;
    }

    case enManageUsersMenueOptions::eUpdateUser:
    {
        system("cls");
        ShowUpdateUserScreen();
        GoBackToManageUsersMenue();
        break;
    }

    case enManageUsersMenueOptions::eFindUser:
    {
        system("cls");

        ShowFindUserScreen();
        GoBackToManageUsersMenue();
        break;
    }
```

```
 case enManageUsersMenueOptions::eMainMenue:
    {
        ShowMainMenue();
    }
    }

}

short ReadManageUsersMenueOption()
{
    cout << "Choose what do you want to do? [1 to 6]? ";
    short Choice = 0;
    cin >> Choice;

    return Choice;
}

void ShowManageUsersMenue()
{

    if
(!CheckAccessPermission(enMainMenuePermissions::pManageUsers))
    {
        ShowAccessDeniedMessage();
        GoBackToMainMenue();
        return;
    }

    system("cls");
    cout << "===========================================\n";
    cout << "\t\tManage Users Menue Screen\n";
    cout << "===========================================\n";
    cout << "\t[1] List Users.\n";
    cout << "\t[2] Add New User.\n";
    cout << "\t[3] Delete User.\n";
    cout << "\t[4] Update User.\n";
    cout << "\t[5] Find User.\n";
    cout << "\t[6] Main Menue.\n";
    cout << "===========================================\n";
```

```
PerfromManageUsersMenueOption((enManageUsersMenueOptions)ReadManag
eUsersMenueOption());
}

void PerfromMainMenueOption(enMainMenueOptions MainMenueOption)
{
    switch (MainMenueOption)
    {
    case enMainMenueOptions::eListClients:
    {
        system("cls");
        ShowAllClientsScreen();
        GoBackToMainMenue();
        break;
    }
    case enMainMenueOptions::eAddNewClient:
        system("cls");
        ShowAddNewClientsScreen();
        GoBackToMainMenue();
        break;

    case enMainMenueOptions::eDeleteClient:
        system("cls");
        ShowDeleteClientScreen();
        GoBackToMainMenue();
        break;

    case enMainMenueOptions::eUpdateClient:
        system("cls");
        ShowUpdateClientScreen();
        GoBackToMainMenue();
        break;

    case enMainMenueOptions::eFindClient:
        system("cls");
        ShowFindClientScreen();
        GoBackToMainMenue();
        break;

    case enMainMenueOptions::eShowTransactionsMenue:
        system("cls");
        ShowTransactionsMenue();
        break;
    case enMainMenueOptions::eManageUsers:
        system("cls");
        ShowManageUsersMenue();
        break;

     case enMainMenueOptions::eExit:
        system("cls");
        // ShowEndScreen();
        Login();

        break;
    }

}

void ShowMainMenue()
{
    system("cls");
    cout << "===========================================\n";
    cout << "\t\tMain Menue Screen\n";
    cout << "===========================================\n";
    cout << "\t[1] Show Client List.\n";
    cout << "\t[2] Add New Client.\n";
    cout << "\t[3] Delete Client.\n";
    cout << "\t[4] Update Client Info.\n";
    cout << "\t[5] Find Client.\n";
    cout << "\t[6] Transactions.\n";
    cout << "\t[7] Manage Users.\n";
    cout << "\t[8] Logout.\n";
    cout << "===========================================\n";



PerfromMainMenueOption((enMainMenueOptions)ReadMainMenueOption());
}

bool  LoadUserInfo(string Username, string Password)
{

    if (FindUserByUsernameAndPassword(Username, Password,
CurrentUser))
        return true;
    else
        return false;

}
```

```cpp
void Login()
{
    bool LoginFaild = false;

    string Username, Password;
    do
    {
        system("cls");

        cout << "\n---------------------------------\n";
        cout << "\tLogin Screen";
        cout << "\n---------------------------------\n";

        if (LoginFaild)
        {
            cout << "Invlaid Username/Password!\n";
        }

        cout << "Enter Username? ";
        cin >> Username;

        cout << "Enter Password? ";
        cin >> Password;

        LoginFaild = !LoadUserInfo(Username, Password);

    } while (LoginFaild);

    ShowMainMenue();

}

int main()

{
    Login();

    system("pause>0");
    return 0;
}
```