



**Xi'an Jiaotong-Liverpool University**

**西交利物浦大學**

# **EEE102 C++ Group Project**

Group member:

Xinxin. Li 1717181

Xuhui. Gong 1718128

Qingwei. Lei 1715917

Jiayi. He 1718132

Date: 2019/05/26

# Content

<b>1. Specifications.....</b>	<b>3</b>
1.1 Overall specifications.....	3
1.2 Customer specification.....	3
1.3 System specification.....	3
<b>2. Analysis and Design.....</b>	<b>3</b>
2.1 User class.....	3
2.2 Movie class.....	7
2.3 Room class.....	10
2.4 Main function.....	18
<b>3. Testing.....</b>	<b>20</b>
<b>4. Bugs report.....</b>	<b>31</b>
User Manual.....	33

## **1. Specifications**

### **1.1 Overall specifications**

Our team is employed by a theatre for the implementation of a movie ticket booking system(Project B). The object of the program is to design a movie ticket booking system which is able to store the information about movies and customers. The customers use their ID to book the tickets and if they do not have, they must logon. Besides, the administrator and manager have higher authority to edit the information of customer and movie, while he/she can not modify customers' booking information. There are two types of rooms with rectangular arrangements of seats: 5 small rooms with  $15 \times 20$  seats and 3 large rooms with  $25 \times 30$  seats each. For different movie, time and type of room, the ticket price is varying.

### **1.2 Customer specification**

Customer user can browse the movie information, select room and at most 5 seats and pay.

### **1.3 System specification**

- The specific movie, room and customer and seat information should be displayed as the requirements while the user is searching;
- While adding a new movie or login a new customer, the user ID and movie ID can not be repeated;
- The administrator and manager can browse, add, modify and delete the information of movie, room and customer;
- Allow the manager to retrieve the cinema operation statistics such as the number of total customer and total income for each movie.

## **2. Analysis and Design**

### **2.1 User class**

The protected type of variables for user:

```
protected:  
    string user_name;  
    string user_password;  
    int user_id;
```

Including user's name, password and ID number.

Declaration of the normal constructor:

```
user(string User_name, string User_password, int User_id); //normal constructor
```

Definition of the constructor:

```
user::user(string User_name, string User_password, int User_id)
{
    user_name = User_name;
    user_password = User_password;
    user_id = User_id;
}
```

The getter and setter methods contain the attributes of user's name, password and ID number. Here is the definition of the getter and setter methods:

```
//getter and setter
string get_user_name();
string get_user_password();
int get_user_id();
void set_user_name(string u_name);
void set_user_password(string u_password);
void set_user_id(int u_id);
```

Getter methods mainly for returning the user's information while setter methods for changing respectively.

There are also friend methods in the user class. Here is the definition:

```
friend vector<user> read_user(); //read the data from user_data.txt
friend void save_user(vector<user> v_user); //save the data as a file(user_data.txt)
friend vector<user> delete_user(vector<user> v_user); //delete a user
```

The read\_user method is to read data from the file of user\_data.txt and the method named save\_user is for save the user's data as a file in reverse. In addition, the method of delete\_user is to delete a user.

To read the user, firstly, an instance object is created:

```
user theuser = user("Not Given", "Not Given", 0);
```

Besides, a vector v\_user is created to store the user objects.

```
vector<user> v_user;
```

Ifstream is user for read user's information from the file named 'user\_data.txt'.

```
ifstream if_read1("user_data.txt");
```

The user has four lines to record in file, the first three lines correspond to the user's name, password and ID number and the forth line 'OOOOOO' is used for marking. As a result, the for loop is used for record how many users are recorded in file.

```
for (int i = 0; i < 40000; i++){
    if_read1.getline(read, 50);
    strread = read;
    if (strread == "OOOOOO"){
        counter++;
    }
}
if_read1.close();
```

Here is the definition of read information from 'user\_data.txt'.

```
for (int i = 0; i < (4 * counter); i = i + 4)
    if_read2.getline(read, 50);
    strread = read;
    theuser.set_user_name(strread);

    if_read2.getline(read, 50);
    strread = read;
    theuser.set_user_password(strread);

    if_read2.getline(read, 50);
    strread = read;
    stringstream ss(strread);
    ss >> id;
    theuser.set_user_id(id);

    if_read2.getline(read, 50);

    v_user.push_back(theuser);
}
if_read2.close();
return v_user;
```

For i is less than the number of users, read each line's information of each user and after finishing reading, close read\_2.

The definition of save users' information is shown below:

```
void save_user(vector<user> v_user){  
    //save the data as a file  
    ofstream of_save("user_data.txt");  
    for (int i = 0; i < v_user.size(); i++){  
        of_save << v_user[i].get_user_name() << endl;  
        of_save << v_user[i].get_user_password() << endl;  
        of_save << v_user[i].get_user_id() << endl;  
        of_save << "000000" << endl;  
    }  
    of_save.close();  
}
```

If want to delete a user, if the user's vector is equal to 0, it indicates that no user has been added. In this case, the manipulation of deleting a user can not be implemented. Else the list number and information of the customers will be listed and can be chosen to erase.

There are some other declarations of methods for the user:

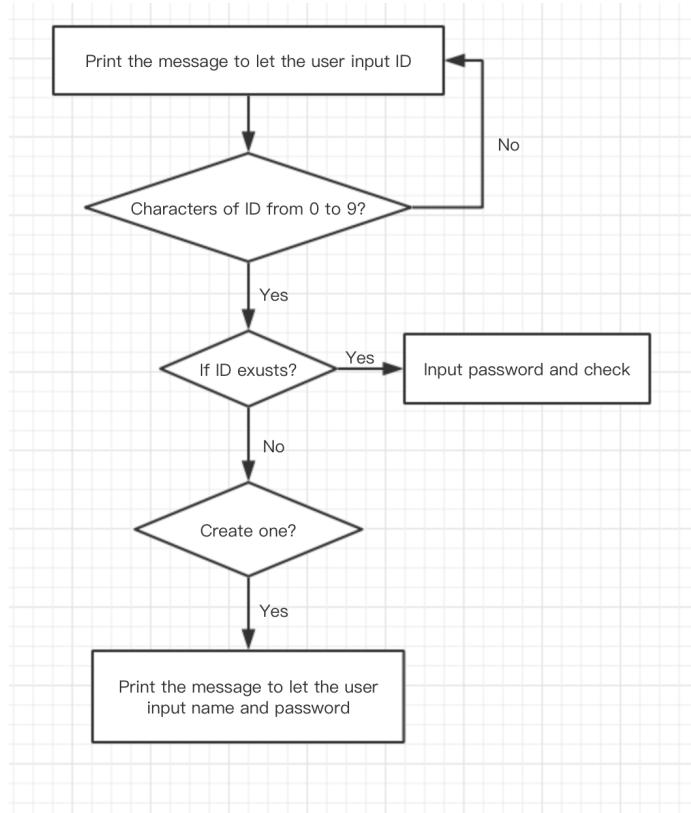
```
void operate_user();  
void showUsers();  
void searchUser(string Sname);  
int userLogin();  
int userIDsearch(int uID, bool &exist);
```

The operating part is divided into four parts:

1. Add a user
2. Delete a user
3. View user list
4. Exit

These manipulations are for the administrate to change and view the information of customers.

The userLogin method is for letting the user login their own accounts. Figure1 shows the flow chart of the userLogin method.



## 2.2 Movie class

The definition of movie class is similar with the user class which including the public methods of normal constructor and getter and setter. Here is the declaration of the public methods of the movie class:

```

public:
    movie(string Movie_name, int Movie_duration, int Movie_id); //normal constructor
    //getter and setter
    string get_movie_name();
    double get_movie_duration();
    int get_movie_id();
    void set_movie_name(string m_name);
    void set_movie_duration(double m_duration);
    void set_movie_id(int m_id);
  
```

The attributes of movie contain movies' name, duration and id number:

```

protected:
    string movie_name;
    int movie_duration;
    int movie_id;
    ...
  
```

The definition of constructor of movie class:

```
movie::movie(string Movie_name, int Movie_duration, int Movie_id)
{
    movie_name = Movie_name;
    movie_duration = Movie_duration;
    movie_id = Movie_id;
}
```

The getter and setter methods of the movie class is shown in below. The getter methods is for acquiring movies' name, duration and id beyond beyond the movie class and the setter method is for the administrator to change the chosen movie's information.

```
string movie::get_movie_name()
{
    return movie_name;
}

double movie::get_movie_duration()
{
    return movie_duration;
}

int movie::get_movie_id()
{
    return movie_id;
}

void movie::set_movie_name(string m_name)
{
    movie_name = m_name;
}

void movie::set_movie_duration(double m_duration)
{
    movie_duration = m_duration;
}

void movie::set_movie_id(int m_id)
{
    movie_id = m_id;
}
```

There are also friend methods for the movie class to read, save and delete movies.

```
friend vector<movie> read_movie(); //read the data from movie_data.txt
friend void save_movie(vector<movie> v_movie); //save the data as a file(movie_data.txt)
friend vector<movie> delete_movie(vector<movie> v_movie); //delete a movie
```

To read movie from the file 'movie\_data.txt', the instance object the movie is created. Each movie's information takes up 4 lines corresponding to the movie's name, time and film id. The forth line is 'OOOOOO' to separate and the number of 'OOOOOO' equals to the number of movies. Here we can use a for loop to realise and in this situation, we can record 100 movies.

```
for (int i = 0; i < 400; i++){
    if_read1.getline(read, 200);
    strread = read;
    if (strread == "OOOOOO"){
        counter++;
    }
}
if_read1.close();
```

The first line records movie's name, the second line records movie's duration and the third line record movie id.

```
for (int i = 0; i < (4 * counter); i = i + 4){
    if_read2.getline(read, 200);
    strread = read;
    themovie.set_movie_name(strread);

    if_read2.getline(read, 200);
    strread = read;
    stringstream ss1(strread);
    ss1 >> duration;
    themovie.set_movie_duration(duration);

    if_read2.getline(read, 200);
    strread = read;
    stringstream ss2(strread);
    ss2 >> id;
    themovie.set_movie_id(id);

    if_read2.getline(read, 200);

    v_movie.push_back(themovie);
}
if_read2.close();
```

Here is the definition of the method of save\_movie:

```
void save_movie(vector<movie> v_movie)
{
    //save the data as a file
    ofstream of_save("movie_data.txt");
    for (int i = 0; i < v_movie.size(); i++){
        of_save << v_movie[i].get_movie_name() << endl;
        of_save << v_movie[i].get_movie_duration() << endl;
        of_save << v_movie[i].get_movie_id() << endl;
        of_save << "OOOOOO" << endl;
    }
    of_save.close();
}
```

We use the ofstream to save.

The operate\_movie method includes four parts:

- Add a movie
- Delete a movie
- View movie list
- Return

The definition of searchMovie method is shown in below. The search is according to the movie's name. Once the input name is same as the one in the list, the information of the corresponding movie will be printed.

```
3 void searchMovie(string Sname) {  
4     vector<movie> v_movie;  
5     v_movie = read_movie();    //read data from movie_data.txt  
6     bool searchFlag = false;  
7     for (unsigned int i = 0; i < v_movie.size(); i++) {  
8         if (v_movie[i].get_movie_name() == Sname) {  
9             cout << "The movie have id " << v_movie[i].get_movie_id() << " with duration " << v_movie[i].get_movie_duration() << endl;  
10            searchFlag = true;  
11        }  
12    }  
13    if (!searchFlag) cout << "No movie found in this name" << endl;  
14}
```

## 2.3 Room class

The struct is a data set which consists a series of data with same or different data types. Here is the declaration of the room class's struct:

```
struct movieTime {  
    int moviePrice;  
    int movieID;  
    string movieName;  
    int movieStartTime;  
  
    int seat[25][30] = {};  
};
```

This struct contains the movie's attributes of moviePrice, movieID, movieName, movieStartTime and the ownership of the seats in the movie theatre which represents through two-dimensional array.

The type of room is divided into two types: small room and big room. We use enum to represent.

```
enum roomType
{
    smallroom,
    bigRoom
};
```

The declaration of the methods in the movie class is shown in below:

```
public:
Room(int RoomType, int RoomNumber, vector <movieTime> MovieTimeVec);
~Room();

//Room type Setter and Getter
int RoomTypeGetter();
void RoomTypeSetter(int);

//Print room's timeTable of a movie
void RoomTimeTableShown();

//Set room's timeTable
movieTime roomTimeTableSetter(int moviePrice, int movieId, string movieName, int movieStartTime, int seat[25][30]); //Single timetable setter for one room

//Put the movieTime(struct) into the vector which stores the movieTime
void roomTimeTableSetterVec(movieTime);

//Return the vector which stores the movieTime
vector <movieTime> roomTimeTableGetterVec();

//One step to arrange or delete room's venue
void arrange_the_venue(); // Arrange the venue
void delete_the_venue(); // modify the venue

//Set the seat to the room(user.buy may use it)
bool write_seat_info(int number, int x, int y, int user_id);

//Print one venue of all the seats(0 for nobody brought, id for someone has bought)
void print_the_seat(int);

//Save and Read
void save_room_info(int room_num);
void read_room_info(int room_num);

//Consider whether user has brought 5 ticket
bool ticket_limation(int venue, int user_id);
```

Here is the definition of the constructor part:

```
Room::Room(int RoomType, int RoomNumber, vector <movieTime> MovieTimeVec)
{
    roomType = RoomType;
    roomNumber = RoomNumber;
    movieTimeVec = MovieTimeVec;
}
```

In addition, the getter and setter method for getting and setting room's type is shown in below:

```

int Room::RoomTypeGetter() {
    return roomType;
}

void Room::RoomTypeSetter(int type) {
    roomType = type;
}

```

There is also the normal constructor part to set the information in the movie's struct.

```

movieTime Room::roomTimeTableSetter(int moviePrice, int movieId, string movieName, int movieStartTime, int seat[25][30]) {
    movieTime roomTimeTable;
    roomTimeTable.movieID = movieId;
    roomTimeTable.movieName = movieName;
    roomTimeTable.movieStartTime = movieStartTime; //Full time 0-24
    roomTimeTable.moviePrice = moviePrice;
    return roomTimeTable;
}

```

The roomTimeTableSetterVec method uses push\_back to add a new movie time at the end of the vector movieTimeVec.

```

void Room::roomTimeTableSetterVec(movieTime MovieTime) {
    movieTimeVec.push_back(MovieTime);
}

```

To show each room's information, if there are movies are arranged in a room, then the movies' information will be printed out. Else If there is no movie is arranged, a message will be printed out to inform.

```

void Room::RoomTimeTableShown() {
    cout << setfill('=') << setw(70) << "=" << endl << setfill(' ');
    if (movieTimeVec.size()) {
        cout << "Number " << roomNumber << "s room TimeTable" << endl;
        cout << setfill('·') << setw(70) << "·" << endl << setfill(' ');
    }
    else {
        cout << "Number " << roomNumber << "s room has no movie arranged." << endl;
    }
    for (int i = 0; i < movieTimeVec.size(); i++) {
        cout << i + 1 << ". Movie Id " << movieTimeVec[i].movieID << " " << " Movie Name " << movieTimeVec[i].movieName << " Movie Start " << movieTimeVec[i].movieStartTime << ":"00 " << "Price " << movieTimeVec[i].moviePrice << endl;
    }
    cout << endl ;
}

```

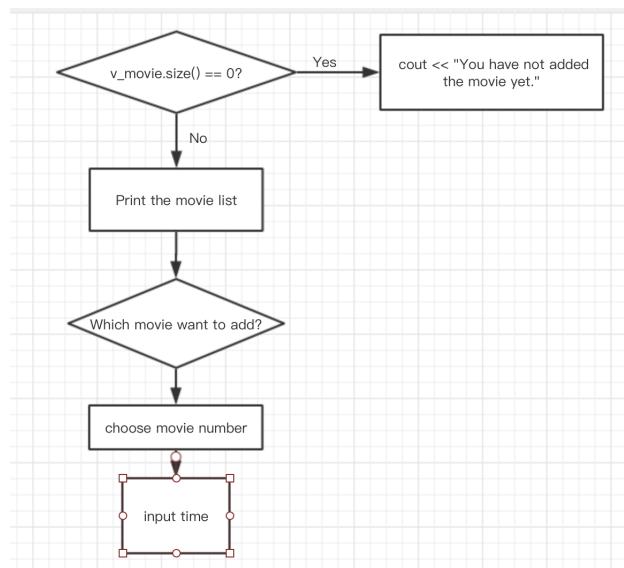
The definition of the method to print the information of movies:

```

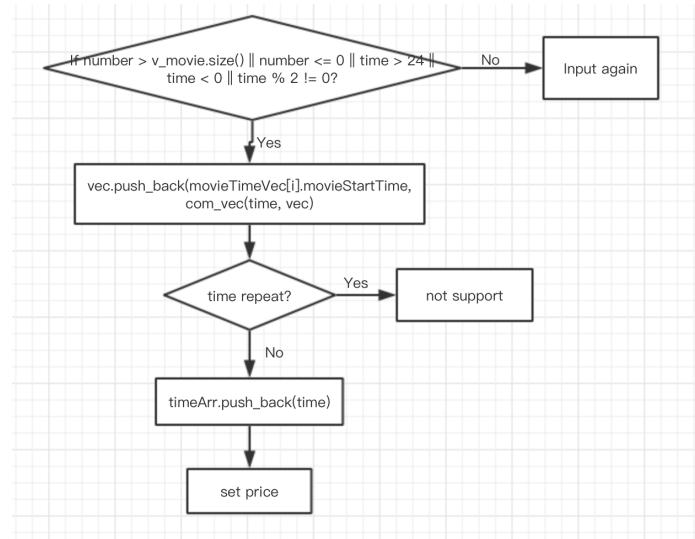
void Room::arrange_the_venue() {
    movie themovie = movie("Not Given", 0, 0);
    vector<movie> v_movie;
    vector <int>timeArr;//Collect the Time
    v_movie = read_movie();
    if (v_movie.size() == 0) {
        cout << "You have not added the movie yet." << endl;
    }
    else {
        cout << "The movie list:" << endl;
        for (int i = 0; i < v_movie.size(); i++) {
            cout << (i + 1) << " Movie name: " << v_movie[i].get_movie_name() << endl;
            cout << " Movie duration: " << v_movie[i].get_movie_duration() << " minutes" << endl;
            cout << " Movie ID: " << v_movie[i].get_movie_id() << endl;
        }
    }
}

```

Here is the flow chart of arranging the movie venue:



The limitations of time:



### The definition of delete the venue:

```

void Room::delete_the_venue() {
    cout << "Number " << roomNumber << "s TimeTable" << endl;
    for (int i = 0; i < movieTimeVec.size(); i++)
    {
        cout << i + 1 << "Movie Id " << movieTimeVec[i].movieID << " " << " Movie Name " << movieTimeVec[i].movieName << " Movie Start " << movieTimeVec[i].movieStartTime << ":00 " << endl;
    }
    cout << "Which number do you want to delete" << endl;

    int selection;
    cin >> selection;

    // movieTimeVec
    if (movieTimeVec.size() > selection)
    {
        movieTimeVec.erase(movieTimeVec.begin() + selection - 1);
    }
    else
    {
        cout << "CHECK VALUE" << endl;
    }
}

```

### The definition of save room as a file:

```

void Room::save_room_info(int room_num) {
    //save the data as a file
    string name = "Room_data.txt";
    char str[1];
    stringstream ss;
    ss << room_num;
    string s1 = ss.str();

    name = s1 + name;
    ofstream of_save(name);

    of_save << roomType << endl;
    of_save << 0 << endl;

    for (int i = 0; i < movieTimeVec.size(); i++) {
        of_save << movieTimeVec[i].movieID << endl;
        of_save << movieTimeVec[i].movieName << endl;
        of_save << movieTimeVec[i].movieStartTime << endl;
        of_save << movieTimeVec[i].moviePrice << endl;
        for (int j = 0; j < 25; j++)
        {
            for (int k = 0; k < 30; k++)
            {
                of_save << movieTimeVec[i].seat[j][k] << endl;
            }
        }
    }

    of_save.close();
}

```

The movie's id, name, start time, price and the ownership of each seat will be saved to the vector. Here is the definition of read the data from the room:

```

void Room::read_room_info(int room_num) {
    ifstream text;
    string name = "Room_data.txt";

    char str[1];
    stringstream ss;
    ss << room_num;
    string s1 = ss.str();

    name = s1 + name;
    text.open(name, ios::in);
    vector<string> strVec;
    while (!text.eof()) {
        string inbuf;
        getline(text, inbuf, '\n');
        strVec.push_back(inbuf);
    }

    int movieTime_value = strVec.size();

    movieTime_value = (movieTime_value - 3) / 754;

    for (int i = 0; i < movieTime_value; i++) {
        movieTime mV;
        int seat[25][30] = {};
        mV.movieID = strToInt(strVec[2 + i * 754]);
        mV.movieName = strVec[3 + i * 754];
        mV.movieStartTime = strToInt(strVec[4 + i * 754]);
        mV.moviePrice = strToInt(strVec[5 + i * 754]);
        int counter = 0;

        for (int j = 0; j < 25; j++) {
            for (int k = 0; k < 30; k++) {
                seat[j][k] = strToInt(strVec[6 + counter + i * 754]);
                counter++;
            }
        }

        for (int i = 0; i < 25; i++) {
            for (int j = 0; j < 30; j++) {
                mV.seat[i][j] = seat[i][j];
            }
        }

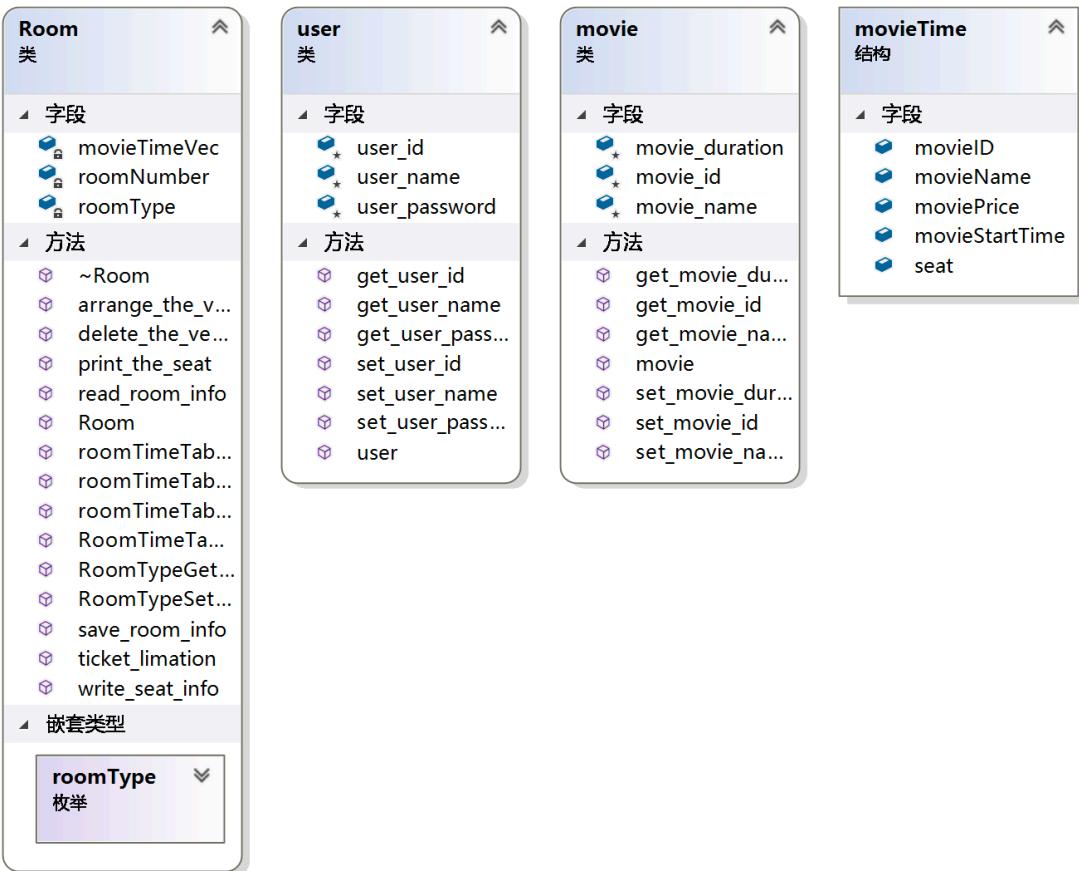
        movieTimeVec.push_back(mV);
    }
}

```

Each room's data takes place 754 rows. In addition, we can read data from three specified rows. For the small room, if the user input the row number is larger than 15 or the column number is larger than 20, then the input is invalid. For the large room, the limitation is 25 and 30. Besides, if the attribute of the seat is not equal to 0, it states that the seat has been occupied. As a consequence, the customer can not buy it any more. Here is the definition:

```
bool Room::write_seat_info(int number, int x, int y, int user_id) {
    if (roomType == 0)
    {
        if (x > 15 || y > 20)
        {
            cout << "Invalid Value" << endl;
            return false;
        }
        else if (movieTimeVec[number].seat[x][y] != 0)
        {
            cout << "Invalid Value, this seat has been bought" << endl;
            return false;
        }
        else
        {
            movieTimeVec[number].seat[x][y] = user_id;
            return true;
        }
    }
    else
    {
        if (x > 25 || y > 30)
        {
            cout << "Invalid Value" << endl;
            return false;
        }
        else if (movieTimeVec[number].seat[x][y] != 0)
        {
            cout << "Invalid Value, this seat has been bought" << endl;
            return false;
        }
        else
        {
            movieTimeVec[number].seat[x][y] = user_id;
            return true;
        }
    }
}
```

(UML Diagram for all classes)



## 2.4 Main function

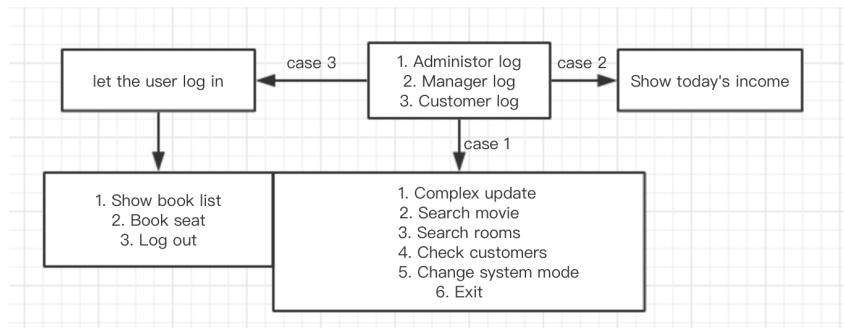
Firstly, the message will be printed out to let the user choose module.

```

cout << "1.\t Admain Login." << endl;
cout << "2.\t Manager Login." << endl;
cout << "3.\t Customer Mode." << endl;
cout << "====="

```

This is the structure of main function:



### 3. Testing

Firstly, the message will be printed out to let the user choose the identity.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Movie system activating:

Please set system mode:
1.      Admain Login.
2.      Manager Login.
3.      Customer Mode.
=====
```

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Administrator mode:
1.      Complex Update.
2.      Search Movie.
3.      Search Rooms.
4.      Check Customers.
5.      Change System Mode.
6.      Exit.
=====
Please input your command:
=====
Complex Update=====
1. Update Movies.
2. Update Room.
3. Update Customer Informatieion.
4. Update Rooms' Movie Timetable
=====
Please input a number to select:
1. Add a movie
2. Delete a movie
3. View movie list
4. Return
```

At first, we choose 1. Then six choice of manipulation is printed out and we choose Complex Update. The movie is hoped to be updated firstly, so we choose 1 and add a movie.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
The movie list:
1
    Movie name: Test
    Movie duration: 44 minutes
    Movie ID: 1

2
    Movie name: lxx
    Movie duration: 1206 minutes
    Movie ID: 112

3
    Movie name: star
    Movie duration: 120 minutes
    Movie ID: 12345

    Please press any key to continue. . .
```

Then we input the movie's name, duration and ID:

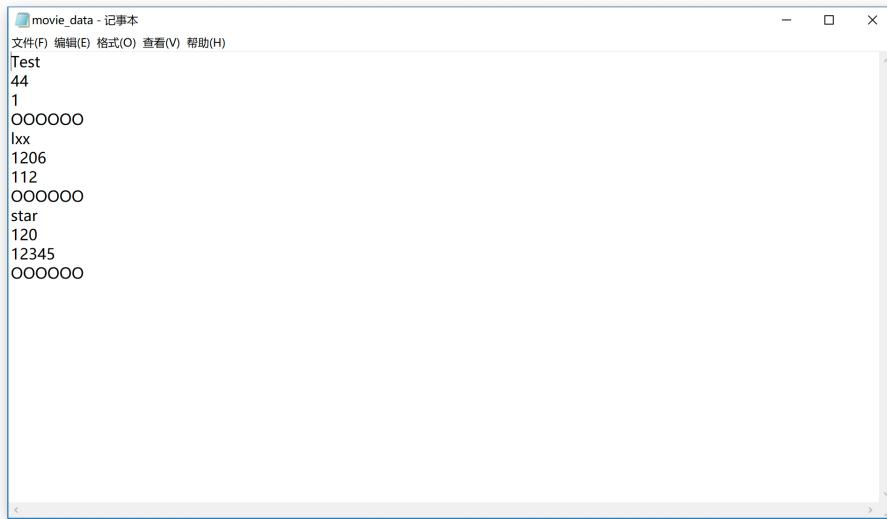
```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
Please input movie name: star
Please input movie duration (Unit: minute): 120
Please input movie ID (Only integer number): 12345
```

Then we choose 3 to view the movie list.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
Please input a number to select:
1. Add a movie
2. Delete a movie
3. View movie list
4. Return
3
```

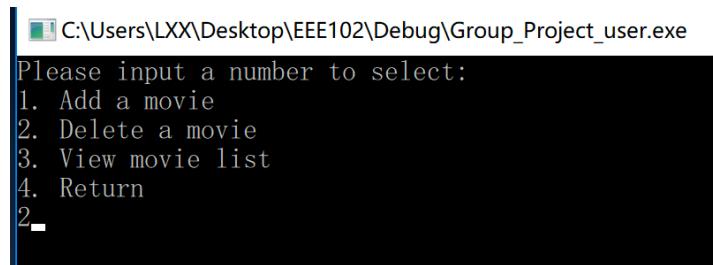
As shown on the screen, the movie named 'star' has already been added.

Besides, the information of the movie star has been saved in the file 'movie\_data.txt' as shown in below.

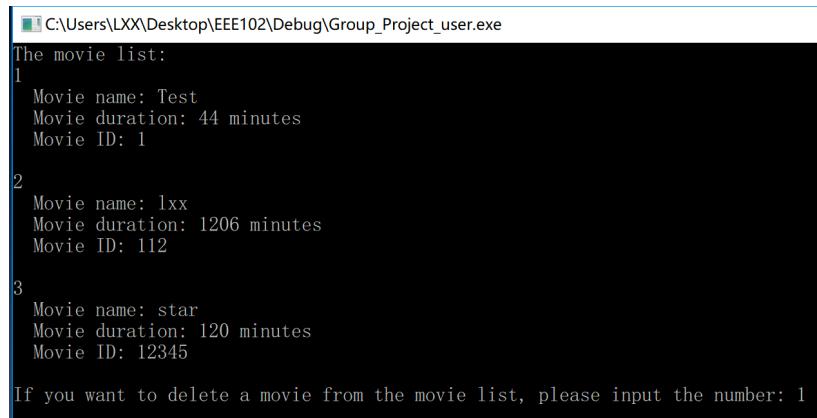


```
movie_data - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Test
44
1
OOOOOO
lxx
1206
112
OOOOOO
star
120
12345
OOOOOO
```

Then we choose 2 to delete a movie.



Then we choose the No.1 movie to delete it.



Then we view the movie list and the file 'movie\_data.txt' again. The results are shown in below:

C:\Users\LXX\Desktop\EEE102\Debug\Group\_Project\_user.exe

The movie list:

1  
Movie name: lxx  
Movie duration: 1206 minutes  
Movie ID: 112

2  
Movie name: star  
Movie duration: 120 minutes  
Movie ID: 12345

请按任意键继续. . . ■

movie\_data - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

|lx  
1206  
112  
000000  
star  
120  
12345  
000000

We can also choose 3 to search the corresponding movies in the room:

C:\Users\LXX\Desktop\EEE102\Debug\Group\_Project\_user.exe

=====Administrator mode=====

1. Complex Update.  
2. Search Movie.  
3. Search Rooms.  
4. Check Customers.  
5. Change System Mode.  
6. Exit.

=====

Please input your command:

Input Room Number:

=====

Number 7's room TimeTable

=====

1. Movie Id 1 Movie Name Test Movie Start 8:00 Price 500

请按任意键继续. . .

Choose 2 to search movie by name:

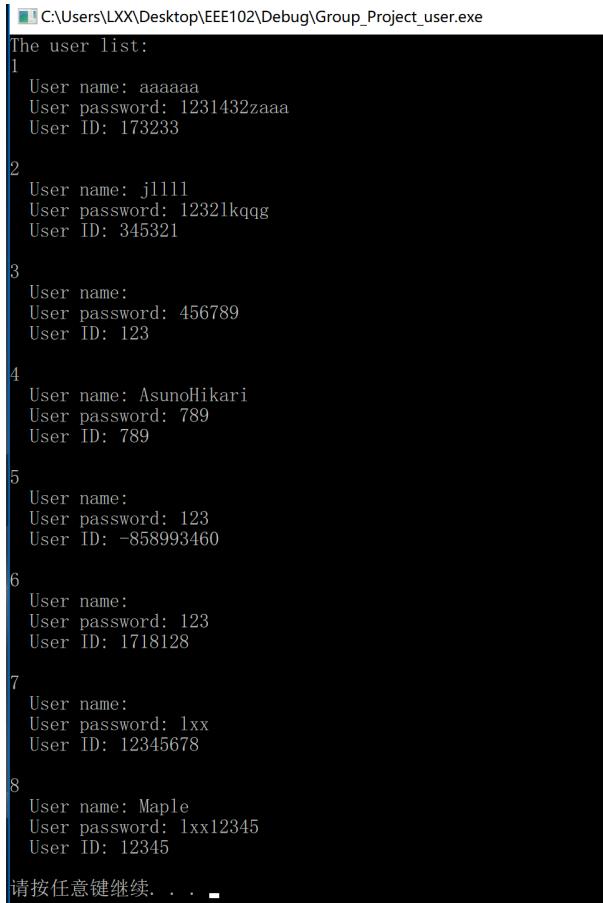
```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Administrator mode:
1. Complex Update.
2. Search Movie.
3. Search Rooms.
4. Check Customers.
5. Change System Mode.
6. Exit.
=====
Please input your command:
Input movie name you want to search:
lxx
The movie have id 112 with duration 1206
请按任意键继续. . . -
```

After return, we choose 3 to update the customer's information and choose 1 to add a user.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
Please input user name: Maple
Please input user password: lxx12345
Please input user ID (Only integer number): 12345-
```

Then choose 3 to view the user list, it can been seen that the user Maple has already been added.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
Please input a number to select:
1. Add a user
2. Delete a user
3. View user list
4. Exit
3
```



```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
The user list:
1
User name: aaaaaa
User password: 1231432zaaa
User ID: 173233

2
User name: j1111
User password: 12321kqqg
User ID: 345321

3
User name:
User password: 456789
User ID: 123

4
User name: AsunoHikari
User password: 789
User ID: 789

5
User name:
User password: 123
User ID: -858993460

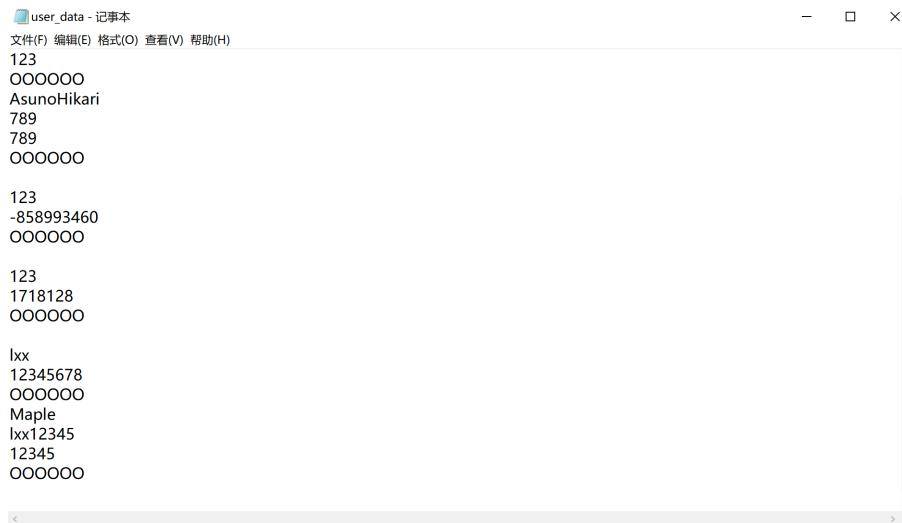
6
User name:
User password: 123
User ID: 1718128

7
User name:
User password: lxx
User ID: 12345678

8
User name: Maple
User password: lxx12345
User ID: 12345

请按任意键继续. . .
```

Then we view the file 'user\_data.txt', the user Maple has already been added into the file.



We can choose 4 to check customers which can show all the customers or search specific customer. Firstly, we choose 1 to show all the customers. The result is shown in below:

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Administrator mode:
1. Complex Update.
2. Search Movie.
3. Search Rooms.
4. Check Customers.
5. Change System Mode.
6. Exit.
=====
Please input your command:
=====
1. Show all customers.
2. Search specific customer.
=====
User name: aaaaaa User password: 1231432zaaa User ID: 173233
User name: j1111 User password: 12321kqqg User ID: 345321
User name: User password: 456789 User ID: 123
User name: AsunoHikari User password: 789 User ID: 789
User name: User password: 123 User ID: -858993460
User name: User password: 123 User ID: 1718128
User name: User password: lxx User ID: 12345678
User name: Maple User password: lxx12345 User ID: 12345
请按任意键继续. . .
```

For the manager model, if the manager logs in, he/she can check today's total income as shown below:

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Manager mode
=====
This is today's income:
800
=====
Please set system mode:
1. Admain Login.
2. Manager Login.
3. Customer Mode.
```

Next, we change to the customer model.

We input a ID that not exists so that create a new one. After inputting the name and password, a new user is created successfully.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====User Mode Activating=====
Input your account ID: (Input unexist ID to create user)
1234567
ID does not exist, do you want to create one in this ID? [1 for YES or else for NO]
请按任意键继续. . .
```

The manipulations that the user can select are showing the book list and booking a seat. Firstly, we choose 2 to book a ticket. Then all the information of rooms and movies are shown.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
Please input user name: lxxxx
Please input user password: lxxxx123
```

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
1. Show book list
2. Book seat
3. Log Out
4. Change System Mode
=====
Number 0's room TimeTable
1. Movie Id 1     Movie Name Test   Movie Start 2:00 Price 100
=====
Number 1's room has no movie arranged.
=====
Number 2's room has no movie arranged.
=====
Number 3's room has no movie arranged.
=====
Number 4's room TimeTable
1. Movie Id 4     Movie Name 456   Movie Start 4:00 Price 100
=====
Number 5's room has no movie arranged.
=====
Number 6's room has no movie arranged.
=====
Number 7's room TimeTable
1. Movie Id 1     Movie Name Test   Movie Start 8:00 Price 500
=====
Number 8's room has no movie arranged.
=====
Number 9's room has no movie arranged.

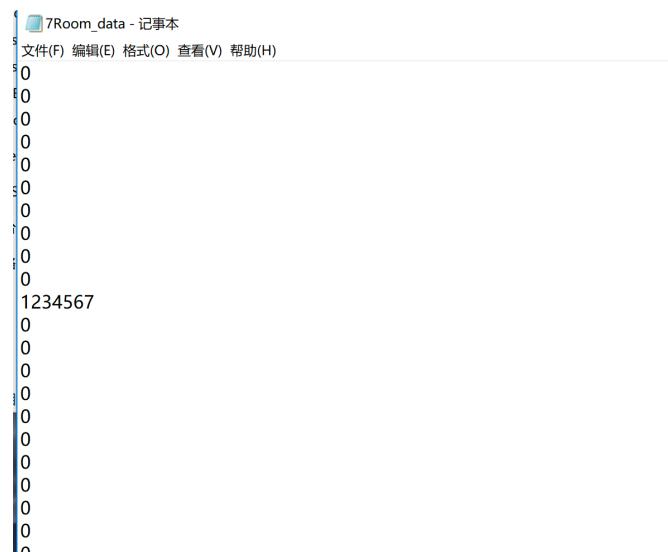
Please input the number of Room: (Start from 0)
Please input the number of movie:
```

We choose 7th room and No.1 movie.

```
Please input the number of Room: (Start from 0)
Please input the number of movie:
0 you can buy, 1 you can not buy
      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Please input row:
4
Please input column:
3
PURCHASE SUCCESS
请按任意键继续. . .
```

After selecting the row and column, the ticket has been bought successfully.

The data in the file updates:



A screenshot of a Windows Notepad window titled "7Room\_data - 记事本". The menu bar includes "文件(F)" (File), "编辑(E)" (Edit), "格式(O)" (Format), "查看(V)" (View), and "帮助(H)" (Help). The content area displays binary data starting with several zeros, followed by the sequence "1234567", and then more zeros. The cursor is positioned at the end of the data.

```
1234567
```

Then we choose 1 to show the user 1234567's book list.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
User Mode Activating=====
User: 1234567 Logged in.

1. Show book list
2. Book seat
3. Log Out
4. Change System Mode

*****
User 1234567      Room 7    The 1's movie
The seat Row 4 andColum 3
请按任意键继续. . .
```

To the update room function in the complex update, we choose Room 8 to change the size to large:

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Which room do you want to operate? (Input a 1 greater number to add room)
Number 0's room TimeTable
1. Movie Id 1   Movie Name Test Movie Start 2:00 Price 100

Number 1's room has no movie arranged.

Number 2's room has no movie arranged.

Number 3's room has no movie arranged.

Number 4's room TimeTable
1. Movie Id 4   Movie Name 456 Movie Start 4:00 Price 100

Number 5's room has no movie arranged.

Number 6's room has no movie arranged.

Number 7's room TimeTable
1. Movie Id 1   Movie Name Test Movie Start 8:00 Price 500

Number 8's room has no movie arranged.

Number 9's room has no movie arranged.

How do you want to modify?
1. Delete
2. Change room size
3. No change

Room size changed to LARGE
请按任意键继续. . .
```

For the function that update room's movie timetable , we choose No.8 room which has not been arranged movies yet. We choose the movie named lxx to add and the time is 8, price is 50.

```
=====
Number 8's room has no movie arranged.

=====
Number 9's room has no movie arranged.

Choose the room you want to operate

1. Arrange the venue
2. Delete the venue

The movie list:
1 Movie name: lxx
    Movie duration: 1206 minutes
    Movie ID: 112
2 Movie name: star
    Movie duration: 120 minutes
    Movie ID: 12345
Which movie would you want to add in room 8
Enter movie number to add
1

What time do you want to arrange this movie
Only even integer time is support. Eg: 8
8

What price do you want to set for one seat?
50

Are you finish adding? Yes for 1 No for 2
1
```

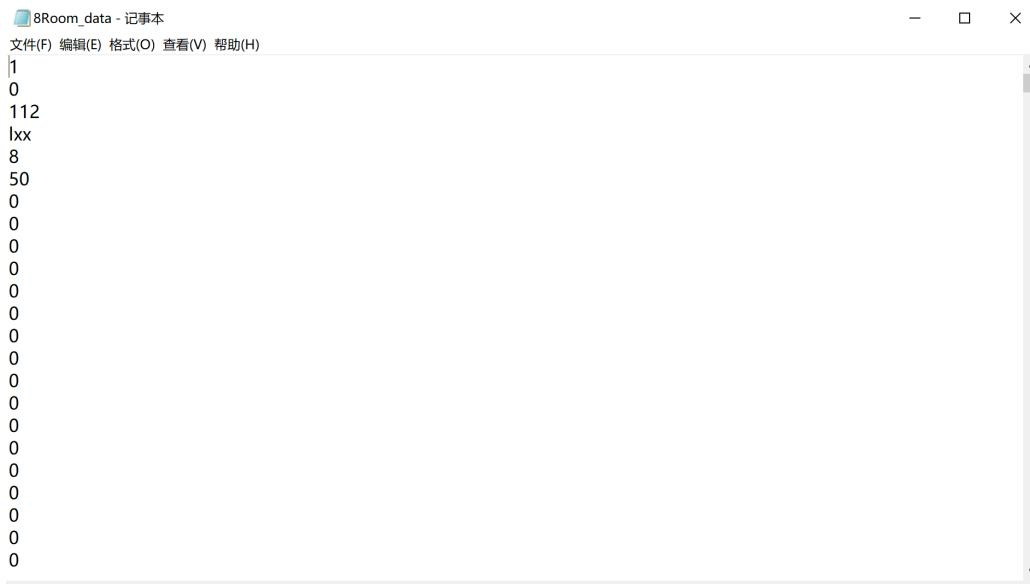
Then we search the No.8 Room, it can been seen that the movie lxx has been added successfully.

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Administrator mode:
1.      Complex Update.
2.      Search Movie.
3.      Search Rooms.
4.      Check Customers.
5.      Change System Mode.
6.      Exit.
=====
Please input your command:

Input Room Number:
=====
Number 8's room TimeTable
-----
1. Movie Id 112      Movie Name lxx  Movie Start 8:00 Price 50

请按任意键继续. . .
```

The data in '8Room\_data.txt' has also been updated.



## 4. Bugs report

When testing the administer mode's check customers mode, when we want to check the specific customer. After inputting the name, there is a display error:

```
C:\Users\LXX\Desktop\EEE102\Debug\Group_Project_user.exe
=====
Administrator mode:
1.      Complex Update.
2.      Search Movie.
3.      Search Rooms.
4.      Check Customers.
5.      Change System Mode.
6.      Exit.
=====
Please input your command:
=====
1. Show all customers.
2. Search specific customer.
=====
Input the name for searching:
Maple
No result in this name.
User name: Maple User password: lxx12345 User ID: 12345
请按任意键继续. . .
```

In the room change part, the statement messages are not clear enough, it is suggested that the previous room sizes are printed out to let the administer makes sure that the room size has been change successfully.

# User manual

## Content

1. Administer mode.....	34
2. Manager mode.....	35
3. Customer mode.....	36

## 1. Administer mode

The manipulations that you can select:

- 1.Complex Update;
- 2.Search Movie;
- 3.Search Rooms;
- 4.Check Customers;
- 5.Change system mode;
- 6.Exit.

### 1. Complex Update:

-Update movies: You should choose a specific movie that you want to modify, then you can

1. Add a movie;
2. Delete a movie;
3. View movie list.

-Update rooms: You can delete a room or change room size to small or to large.

-Update customers' information: You can show all the customers or search the specific customer by name.

-Update room's movie timetable: Firstly, you should choose a specific room, then you can

1. Arrange the venue: You can choose some movies to be added into the room and set their time and seat price.
2. Delete the movies in the venue.

## 2. Manager mode

The manager can only view today's income.

### 3. Customer mode

Firstly, you should log in by inputting your Id and password. If your Id not exists in the system, the system will create a new one for you if you want.

After log in, you can:

1. View book list;
2. Book seat;
3. Log out.

You should select the room and choose a movie.

The ownership of seats represents as 0 means that the seats have not been occupied yet and you can choose through inputting the column and row's number.

Hint: you can buy maximum 5 tickets in total.