Xi'an Jiaotong-Liverpool University

西交利物浦大學

EEE 102 Group Project

Author Yibo Shan (1717599)

Tianyu Cai (1717617)

Jianheng Yuan (1716258)

Dequn Teng (1717617)

Module EEE102 - Experiment Report

Teacher Dr. Qing Liu

Date 23^{th} / May / 2019

Abstract

In this group project, a movie ticket booking system was built. Firstly, the group analyze the given problem and drew the flow charts of each part in this project, including the room, administrator, movie and customer. Secondly, the functions of each identity (the administrator, customer and manager) were simulated successfully based on the drawn flow charts. Then, for the testing section, the group tested all the necessary contents in this project and found some limitations and bugs. After that, the group tried advance method to overcome the problems and some limitations still existed. Finally, the result is acceptable and satisfactory and a brief conclusion will be demonstrated at the end of this report.

Key Words: C++, movie ticket selling system.

Contents

Chapter 1

Introduction

1.1 Background

C++ is a general-purpose programming language developed by Bjarne Stroustrup and an extension of the C language. It has object-oriented, general-purpose programming features, while also supporting underlying memory operations [1]. Based on these two characteristics, c++ has an efficiency advantage over Java and is easy to understand compared with C language.
A computer booking system is a system whereby publicly accessible computers can be reserved for a period of time. These systems are commonly used in facilities such as public libraries to ensure equitable use of limited numbers of computers[2]. Bookings may be done over the internet or within the library itself using a separate computer set up as a booking terminal. Computer booking systems allow public service with reduced staff involvement[3].
C++ is known for its efficient, which is based on object-oriented features and the efficiency of no virtual machines. Therefore, c++ is a widely used language in system development.
Therefore, in this assessment, students are expected to develop movie tickets booking system based on the features learned in the EEE 102 lectures. These features include but not limit to the file operation, encapsulation. The final goal for the admin to browse, modify information for the customers, rooms, and movies and finally retrieve statics information of the movie. The final goal for the customer is to buy the ticket for the specific movie.

1.2 Thesis Statement

In this report, firstly, a brief introduction of the C++, and computer booking system is described above. Then the body paragraph is designed to follow the software development procedures, which contains the problem statement, analysis, design, implementation, and testing. Finally, the evaluation and final conclusion will be presented with the reference indicating the work of other people.

Chapter 2

Body Paragraph

2.1 Problem Statement

In this project, a movie ticket booking system of the theatre should be implemented with more details as follows:

2.1.1 Room

There are totally 5 small rooms $(15 * 20 seats)$ and 3 large rooms $(25 * 30 seats)$. The arrangement of seats is rectangular. In addition, the program should print out the map on the screen, indicating the seat number and seat state (occupied or not) of each seat.

2.1.2 Ticket

Each ticket has its price, corresponding movie and seat. The price of the ticket can be influenced by different movies, time as well as different types of room. Additionally, the program should print out all of the necessary information of this ticket on the screen, including the movie name, the show time of the movie, the corresponding room and seat number.

2.1.3 Administrator Function One

There are mainly two functions for the administrator specified with Administrator Function One and Administrator Function Two. Modify, view, add and delete the movie information, room information and customer information. However, the booking information of customers cannot be modified by the administrator. However, the booking information of customers cannot be modified by the administrator.

2.1.4 Administrator Function Two

Search and view specific the information of room, movie and customer if required. Another important rule is that the ID number of user and movie cannot be repeated when the administrator adds the information of customer or movie.

2.1.5 Customer

There are totally two functions for the customer. View the movie information. After choosing the movie, customers can select the room and seat and buy the corresponding ticket(s) (5 tickets at most). For these two functions, the program should print out the information that the customer wants to view on the screen. Besides, after the customer buy the ticket, the program should change the state of the corresponding seat, indicating that this customer occupies this seat.

2.1.6 Manager

The responsibility of manager is to retrieve necessary statistics about the operations of cinema such as total income for one movie, total amount of customers and other important information. For this function, after the administrator change the information of movie, room or customer, the manager should store the new information to the database for other users to visit.

2.1.7 Database

Since it is a system, which needs a place to store the history information. There are two kinds of approaches to store this information, which are file based approach and the database based approach. Even though this system does not require that mush standard of requirement. But for the future consideration, it is better to use the database based solution due to the limitations of the file based system, specified as follows.

- Separation and isolation of data
  - Each program maintains its own set of data.
  - Users of one program may be unaware of potentially useful data held by other programs.
- Duplication of data
  - Same data is held by different programs.
  - Wasted space and potentially different values and/or different formats for the same item.
- Incompatible file formats
  - Programs are written in different languages, and so cannot easily access each other s files.
  - Fixed Queries Proliferation of application programs
  - Programs are written to satisfy particular functions. Any new requirement needs a new program.

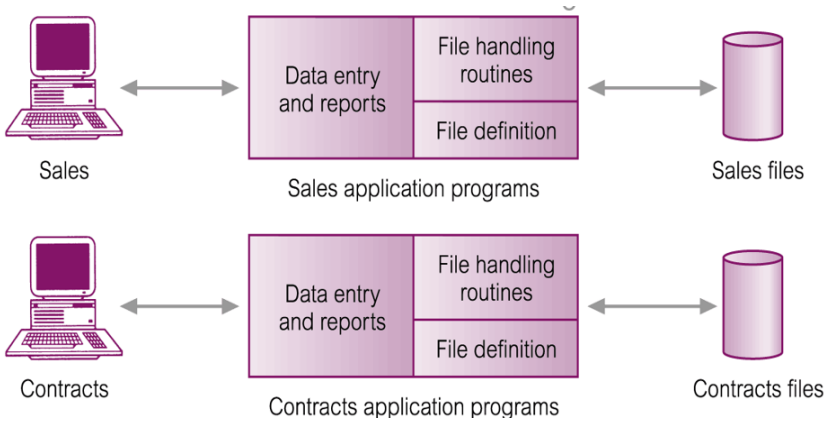

Fig. 2.1: File based system

Therefore, the database based approach is defined as follows.

- Shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization.
- System catalog (metadata) provides description of data to enable program–data independence.
- Logically related data comprises entities, attributes, and relationships of an organization's information.

Additionally, a database management system is defined as follows.

- A software system that enables users to define, create, maintain, and control access to the database.
- (Database) application program: a computer program that interacts with database by issuing an appropriate request (SQL statement) to the DBMS.

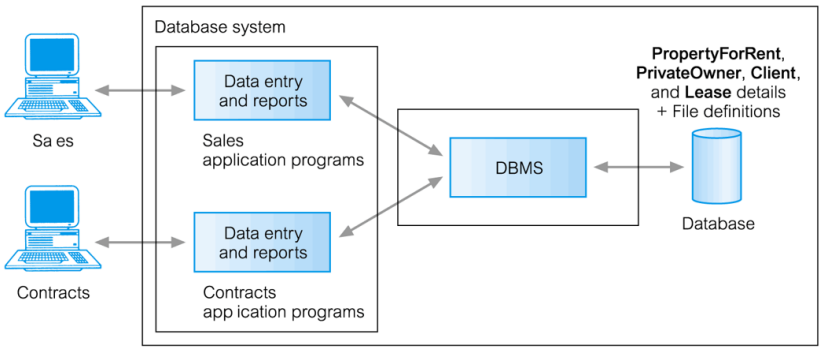

Fig. 2.2: File based system

2.2 Analysis

There are three parts of this assessment, which are rule simulation, class definition, and file operation specified as follows.

2.2.1 Room

Inputs

- When a customer chooses a seat, the user should input a seat number and the program should find the corresponding seat from the room class and the state of this seat should be changed.
- When the administrator sets a movie in a room, the user should enter a movie name corresponding to the room class. In addition, when the administrator sets a movie in a room, the user should input the show time of the movie corresponding to the movie.
- In addition, when the administrator sets a movie in a room, the user should input the show time of the movie corresponding to the movie.

outputs

- For outputs, there are totally four kinds of variable that should be outputted – the state of each seat, the number of each seat, the movie name and the show time of the movie. Therefore, the basic information of the room class can be displayed.

Additional requirements or constraints

- For additional requirements, if the user wants to view one of the specific information of the room, the program should have the function to output this information. For example, if the customer wants to view the movie of the room, the user should recall a member function to return the variable “movie” in the room class.

2.2.2 Ticket

Input

Let administrator to input the information of every movie, such as movie ID, movie name, room, starting time, lasting time and the price factor of movie.

- movie ID: input a movie ID and store in an int variable.
- movie name: input a movie name and store in a string variable.
- room: input the room number and store in an int variable.
- starting time: input the starting time of a movie and store in an int variable.
- lasting time: input the lasting time of a movie and store in an int variable.

- price factor of movie: input the price factor of movie and store in a float variable.

Output

Write the information of every movie input by administrator into the database, such as movie ID, movie name, room, starting time, lasting time and the price.

- movie ID: store the movie ID in an int variable.
- movie name: store a movie name in a string variable.
- room: store the room name in an int variable.
- starting time: store the starting time of a movie in an int variable.
- lasting time: store the lasting time of a movie in an int variable.
- price: store the price in a float variable.

Additional requirements or constraints

- When write the room information into the database, the program will covert the input room number to the corresponding room name. Room number 1 to 5 are small room; room number 6 to 8 are big room.
- When write the price of the movie into the database, the program will generate the price according to the factors of room, time and price. If the room is the small room, the room price factor will be 1.2; if the room is the big room, the room price factor will be 1.0. If the starting time is 0 to 12, the time price factor will be 1.0; if the starting time is 12 to 24, the time price factor will be 1.2. The formula of the movie price is:

$$Movieprice = timepricefactor * roompricefactor * moviepricefactor$$

2.2.3 Administrator Function One

Inputs

- When user input the username and password, program will verify identity, afterwards, user get into the administrator interface. At this interface, there are four choices for user to select, and user can input the corresponding number to overtake the function that they want:
- When administrator want to surf all the movie that stored in the object array they can input 1, the whole information about a movie will display on the screen. Moreover, all the stored movie information will be displayed included movie ID, movie name, location of movie, date and lasting time. Then, user can go back to the administrator interface.

- When administrator want to add new movie, they can input number 2 and then they will enter the interface of add new movie. However, in this process, user will input the movie name, ID, location, price parameter, date (month, day, hour and minute) and lasting time of a movie. And these information will be stored in a object array with the guideline of movie date.

- When people want to delete a movie, they can input number 3 and input the number of movie. Afterwards, when user go back to surf the movie stored in the object array, they will find that movie is disappear.

- The function of modify is the combination of delete and add new movie, under this circumstance, if user want to modify the information of a movie, they have to delete it and create a new one.

Outputs

- When user get into the administrator surface, program will require user input number which represents the function of this program: 1. Surf all the movie 2. Add new movie 3. Delete movie 4. Exit.

- When user input number 1, program displays all the information of a movie: name, ID, location, date and lasting time. However, if there is no movie stored in the object array, screen will display nothing except let user to press any key.

- When user input number 2, the screen shows the number of this movie which starts from 1. Afterwards, program requires user input the name, ID, location, date (month, day, hour, minute) and lasting time consequently. Finally, program inquiries user whether he or she want to continue add movie, if user input 1, which means yes, the number of movie will plus one and go back to the beginning of this function, if user input 2, which means no, program will go back to the administrator interface.

- When user input number 3, program will require user input the number of the movie that he want to delete, user input the number and they will find that movie disappears from the surf list. However, the number which input by user is bigger than the maximum number of existed movie, program will say: “sorry, there is not so much movie.”

- When user input 4, the program will go back to the identity surface.

Additional requirements or constraints

- This program will handle with the invalid input by itself. For example, there are only 4 selections in the interface of administrator, if user input number bigger than 4 or less than 1, the program will inform user that it is an invalid input and require a re-input. This function also works in the process of create a new movie: if the user input the ID out of range,

program will require user input a valid one; the number of cinema room out of range (5 small rooms and 3 big rooms, the maximum number is 8), program will require another one; the date of movie is not valid (month bigger than 12, or day bigger than 31, hour bigger than 23 or minute bigger than 59), user will input an other one number.

2.2.4 Administrator Function Two

Inputs

- To view the information of one movie, room or customer, the user should input the ID number of the corresponding movie, room or customer. Therefore, the program can search the database and obtain the corresponding movie, room or customer according to the input ID number.

Outputs

- After entering the correct ID number, the program should obtain the corresponding movie, room or customer and printed out the detailed information on the screen with more details as follows.
- Movie the name, ID number, price parameter, position, release date and release time;
- Room the room number, map, states of each seat and movie in this room; Customer – the ID number, name, password, telephone number and bought movie.

Additional requirements or constraints

- Since this function is just to view the information of one movie, room or customer, this function does not have additional requirements or constraints.

2.2.5 Customer

Inputs

- In the browsing movie information function
- In the login function
  - Let customer to input the telephone number and the password.
  - For creating new account, let customer to input the name, telephone number and password. Then, store them into the database.
- In the buying tickets
  - Let customer to input the command to select the movie name.
  - Let customer to input the command to select the starting time and room of the movie.
  - Let customer to input the command to select the seat.

Outputs

- In the browsing movie information function
  - Display all of the recent films.
- In the login function
  - Display the relevant prompt information to let customer know what need to be input.
  - Display the relevant prompt information to let customer know whether login or creating a new account successfully.
- In the buying tickets
  - After customer selecting the movie name, display all of the available movies.
  - After customer selecting the certain movie, display the relevant prompt information to let customer to select the seat.
  - Display the map of the room selected by customer.
  - After customer buying a ticket, display the movie name, starting time, room, seat and price of that ticket.

Additional requirements or constraints

- When customer has selected the movie name, all of the available movie information will be displayed and the program will not let customer to select the room and time separately.
- During selecting the seat, customer should choose the number of the seat from 1 to 300 for small room and from 1 to 750 for big room.

2.2.6 Manager

Inputs

- Firstly, once finishing the display of one movie, the program should have the function to obtain the price of each ticket of this movie. After that, the manager is responsible to do the statistics of the prices of all tickets of this movie.

Outputs

- After doing the statistics of the prices of all tickets of this movie, the program should calculate the total income of this movie and print out the result on the screen for the administrator to view.

Additional requirements or constraints

- For the manager, another requirement is that the value of the total income of one movie should be stored in the database for the administrator to view if necessary.

2.2.7 Database

On inputs

- Select: $columnExpression, TableName, conditions$
  - column expression means the name of the columns
  - Table name means the name of the table you want to query
  - conditions are the constraints you want your selected outcome to have.
- Update: $TableName, columnName1, searchCondition$
  - TableName is the operated table
  - columnName is the column to be operated
  - searchCondition is the satisfied condition to be operated
- Delete: $TableName, searchCondition$
  - TableName is the operated tab;e
  - searchCondition is the satisfied condition to delete
- Insert: $columnList, dataValueList$
  - columnList is the column you want to insert
  - dataValueList is the value to be inserted

outputs

- Select: $selectedtuples$
- Modify: $void$
- Delete: $void$
- Insert: $void$

Additional requirements or constraints

- Select:
  - columnList is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.
  - Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column. dataValueList must match columnList as follows:
    - number of items in each list must be same;

    - must be direct correspondence in position of items in two lists;
    - data type of each item in dataValueList must be compatible with data type of corresponding column.

- Update
  - TableName can be name of a base table or an updatable view.
  - SET clause specifies names of one or more columns that are to be updated.
  - WHERE clause is optional: if omitted, named columns are updated for all rows in table; if specified, only those rows that satisfy searchCondition are updated.
  - New dataValue(s) must be compatible with data type for corresponding column.

- Delete
  - TableName can be name of a base table or an updatable view.
  - searchCondition is optional; if omitted, all rows are deleted from table. This does not delete table. If search condition is specified, only those rows that satisfy condition are deleted.

- Insert
  - columnList is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.
  - Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.

2.3 Design

This section contains the specific algorithm in solving the required components mentioned in the analysis part, which are the rule simulation, class definition, and the file operation. Finally, the user input working flow is specified in the end.

2.3.1 Room

For the design of rooms, define two class namely “bigRoom” and “smallRoom” respectively, representing the small rooms and the large rooms. Each class has the same members as follows.

- Member variables
  - Define an array of type bool namely “seatState”, which has 750 members for the class “bigRoom” and 300 members for the class “smallRoom”, representing the state of each seat in the room. That is, whether this state is occupied or not.

  - Define a variable of type string namely “roomNumber”, representing the room number of the room.
  - Define a variable of type string namely “movie”, representing the name of the movie which will be shown in this room.
  - Define an array of type integer namely “seatNumber”, which has 750 members for the class “bigRoom” and 300 members for the class “smallRoom”, representing the seat number of each seat.
  - Define a variable of type string namely “time”, representing the show time of the movie in this room.

- Member functions
  - Define a member function namely “iniSeatState” of type void, which is used to initialize the state of each seat in this room in the beginning of the program. That is, change the state that all the seats are unoccupied.
  - Define a member function namely “returnSeatState” of type bool, which is used to return the value of the variable “seatState” of one specific seat if necessary.
  - Define a member function namely “returnRoomNumber” of type string, which is used to return the value of the variable “roomNumber” of the class “room” if necessary.
  - Define a member function namely “returnSeatNumber” of type int, which is used to return the value of the variable “seatNumber” of one specific seat if necessary.
  - Define a member function namely “printOutState” of type void, which is used to print out the states of all seats in a room (the variable “seatState”) on the screen.
  - Define a member function namely “setSeatState0” of type void, which is used to change the state of one specific seat (the variable “seatState” of type bool) to 0, indicating that this state is unoccupied.
  - Define a member function namely “setSeatState1” of type void, which is used to change the state of one specific seat (the variable “seatState” of type bool) to 1, indicating that this state is occupied.
  - Define a member function namely “setMovie” of type void, which is used to set the movie to one class “room” (change the value of the variable “movie”, indicating that this movie will be shown in this room.
  - Define a member function namely “returnMovie” of type string, which is used to return the value of the variable “movie” if necessary.
  - Define a member function namely “setTime” of type void, which is used to set the show time of the movie in the class “room” (change the value of the variable “time”), indicating that the movie will be shown in this time.

– Define a member function namely “returnTime” of type string, which is used to return the value of the variable “time” if necessary.

– Define a member function namely “viewRoomInformation” of type void, which is used to view the information of the room class. That is, to print out all the necessary information of the room class, including the name of the movie in this room, the show time of the movie and the state of each seat in this room (the variable “seatState”, “movie” and “time”).

– Define a member function namely “deleteRoomInformation” of type void, which is used to delete the information of the room class. That is, to delete all the necessary information of the room class, including the name of the movie in this room, the show time of the movie and the state of each seat in this room (the variable “seatState”, “movie” and “time”).

2.3.2 Ticket

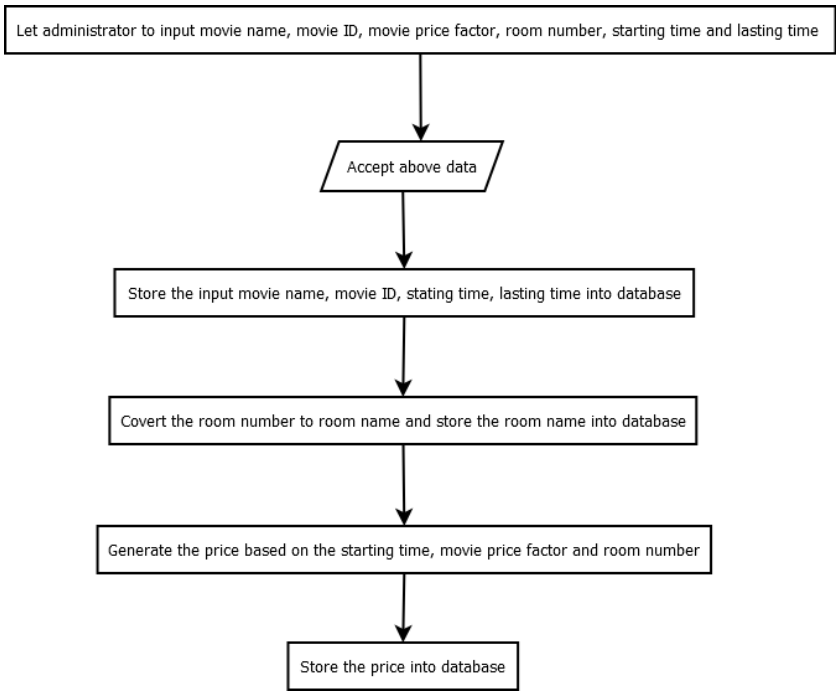

Fig. 2.3: Administrator Function One part One

2.3.3 Administrator Function One

The flow chart in this part is shown below:

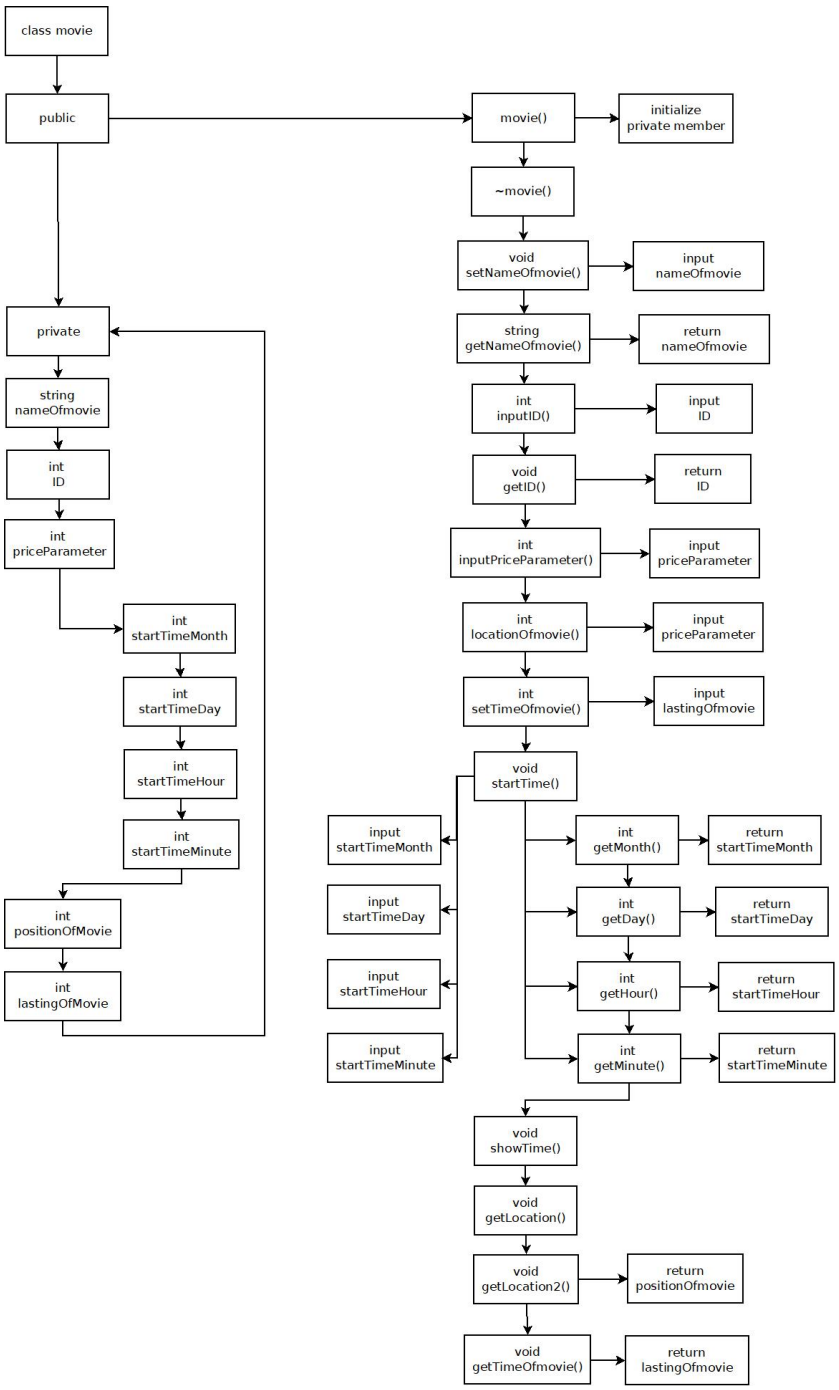

Fig. 2.4: Administrator Function One part One

This flow chart shows private and public part of class movie. There are 9 private variables in this class: name of movie, movie ID, price parameter, time (month, day, hour and minute), position and last time of movie. There are 22 member functions in public part, except constructor (initialize all private variables) and destructor. The rest member functions have two kinds: one is entry number or string to correspond variables, the other one is get the value of each variables. Program claims these member function and give them definition consequently which makes them

available in following function.

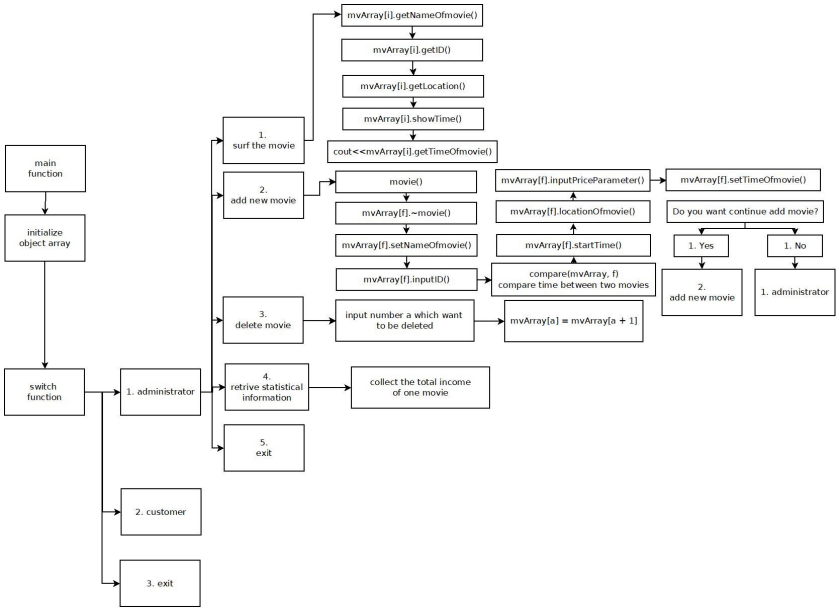

Fig. 2.5: Administrator Function One part two

- This is the principle of administrator. For convenience, flow chart uses the form of object array.
- In the first choice: surf the movie. Program prints movie information from the first in data base one by one until the last information displays on the screen.
- In the second choice: add new movie. There exists a variable represents the number of movie in the data base. Once user adds a new movie into data base, this variable plus one, and the number of object array plus one. Under this circumstance, program avoids add movie from beginning and store the former movie information. Moreover, there exists a flow in terms of add new movie as shown in flow chart. Finally, user can determine whether they continue to add movie.
- In the third choice: delete new movie. Program assigns the later object array to the former one to keep the continuous of object array. Moreover, user can determine the movie that they want to delete.
- In the fourth choice: retrieve statistical information. Administrator employs this program to monitor the income of each movie because the information of each movie is collected in this function.
- In the fifth choice: exit. User exist administrator surface.

2.3.4 Administrator Function Two

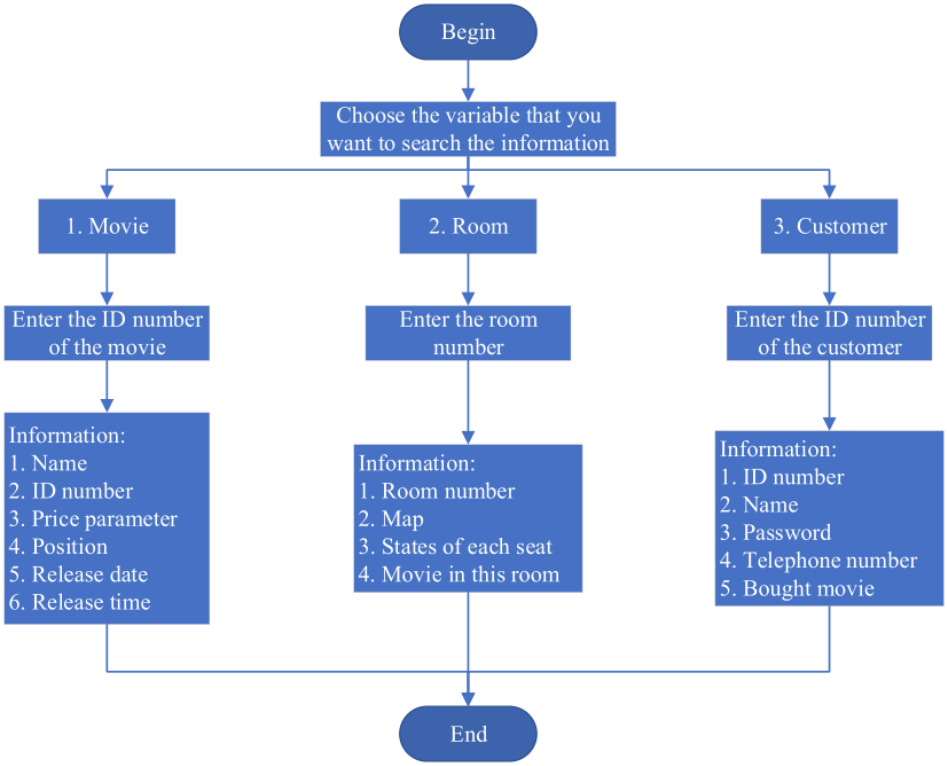

Fig. 2.6: Administrator Function Two

- Ask the user to choose one variable from the movie, room and customer (if the user is inclined to view a movie, room or customer).
- Ask the user to enter a number, which is the ID number of the movie, room or customer that the user is inclined to view its information.
- If this input ID number is correct, then the program will search for the corresponding movie, room or customer in the database.
- If the program obtains the correct movie, room or customer successfully, then the program will print out its necessary information on the screen with details as follows.
- Movie the name, ID number, price parameter, position, release date and release time;
- Room the room number, map, states of each seat and movie in this room; Customer the ID number, name, password, telephone number and bought movie.

2.3.5 Customer

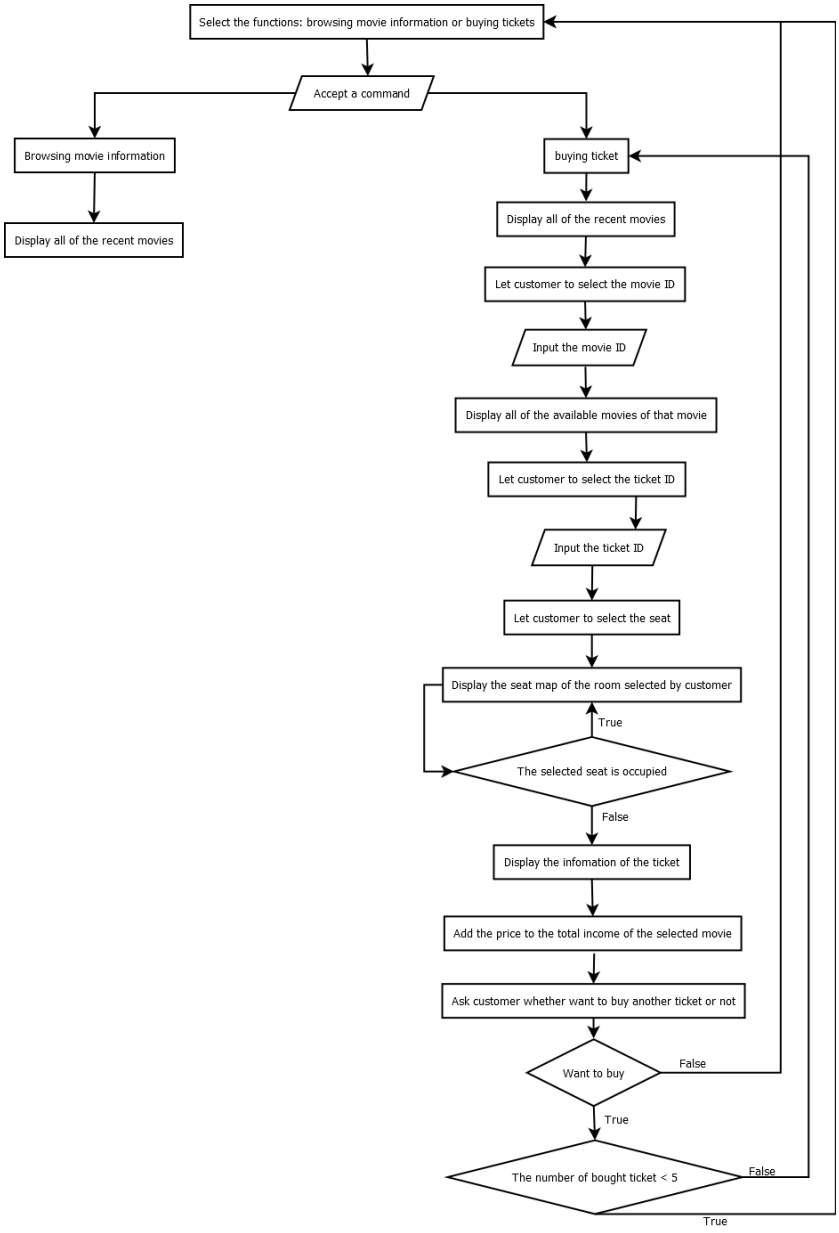

Fig. 2.7: Customer

Customer mainly has two functions, which are browsing the recent movies and buying the ticket. For the browsing the recent movies function, the program will display all of the recent movie from the database. For the buying ticket function, the objective of it is to generate a ticket with the movie name, room, starting time, lasting time and price. Then, add the price of this ticket to the total income of that movie, which can implement the function of total income statistics of a movie. The generally procedures of buying ticket function are listed below.

- Let customer to select the movie name

- Let customer to select the ticket ID, which represents a kind of ticket information of the selected movie
- Let customer to select the seat
- If the selected seat has been occupied, let the customer to select the seat again
- Generate the ticket based on the selection of the customer and display the information of the ticket.
- Add the ticket price to the total price of that movie.

2.3.6 Manager

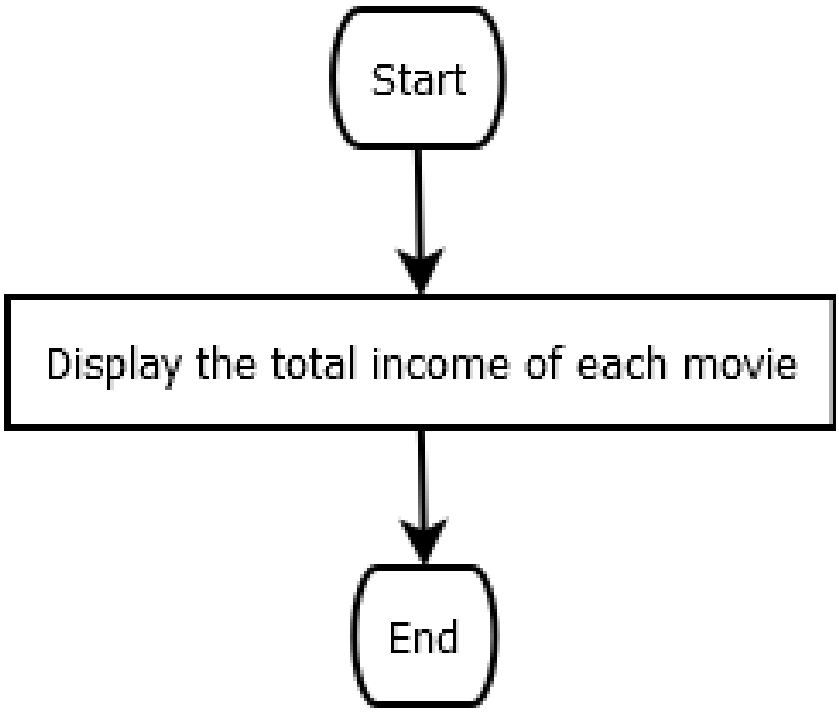

Fig. 2.8: Manager

The manager just need to browse the information of all the statical information.

2.3.7 Database

- Select

$$const char * select_query = "SELECT DISTINCT movieId, movieName from movieTimeSlot";$$

- Delete

$$stringdeleteStr = "deletefrommovieTimeSlotwheremovieId = ("+to_string(a)+")";$$

- Insert

$$insertintomovieTimeSlot(movieId, moviename, room, startingtime, lastingtime, seatstatus, price,$$

totalPrice)

$$values("+to_string(Temp.returnID())+",'"+stest+"','"+Temp.returnRoom()+"',"+$$

$$to_string(Temp.getHour())+","+to_string(Temp.getTimeOfmovie())+",'"+Temp.seatStatus()+$$

$$"'," + to_string(Temp.generatePrice()) + "," + to_string(0) + ")"$$

2.3.8 User Input Working Flow

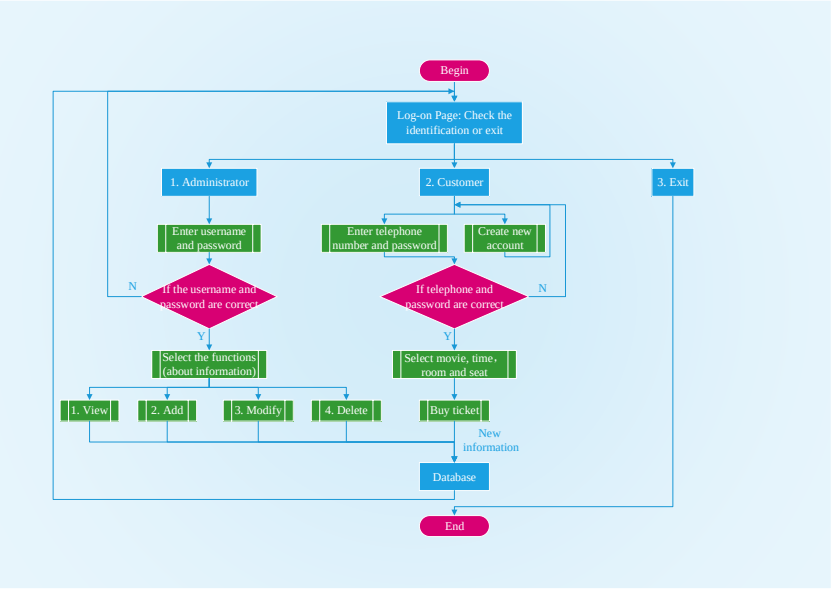

Fig. 2.9: User Input Working Flow

- First, the user is asked to choose one selection from the movie, room and customer to determine which kind of variable that the user is inclined to search.
- Secondly, the user is asked to enter the ID number of the movie, room or customer that the user is inclined to search.
- Then, the program will search the corresponding movie, room or customer in the database based on the given ID number.

- Finally, if the program obtains the corresponding movie, room or customer successfully, then the program should print out its necessary information on the screen with details as follows. Movie the name, ID number, price parameter, position, release date and release time; Room – the room number, map, states of each seat and movie in this room; Customer – the ID number, name, password, telephone number and bought movie.

2.4 Implementation

There are nine files in the code file.

- The main.cpp file contains the processes for making the processing flow available
- the NEW.db file has contained the previous movie ticket records, which the admin saved.
- customer.h contains the predefined variable name and the function declaration used in the customer.cpp file.
- The customer.cpp file contains the definition of behavior of the customer.
- movie.h contains the predefined variable name and the function declaration used in the movie.cpp file.
- movie.cpp defines the attributes such as movieID, movie name, which is unique for a movie.
- Furthermore, the sqlite3.c, sqlite3.dll and sqlite3.h files contains the information used for sqlite3 database.

2.5 Testing

2.5.1 Normal Test

Introductory Instruction

When user get into this program:

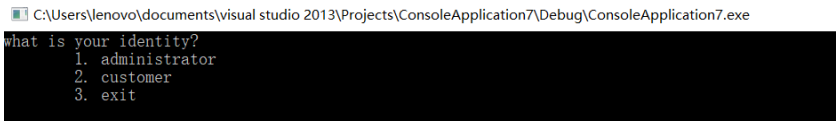

Fig. 2.10: Beginning of program

Admin

When user input 1, the ticketID, movieID, movieName, room, startingtime, lasting time, seatstatus, price, and total price are displayed in the table.
It is worth noting that there are many zeros in the seat status indicating whether the seat in this

case has been taken or not. When the customer surf the information, this will be printed in the selectable table.

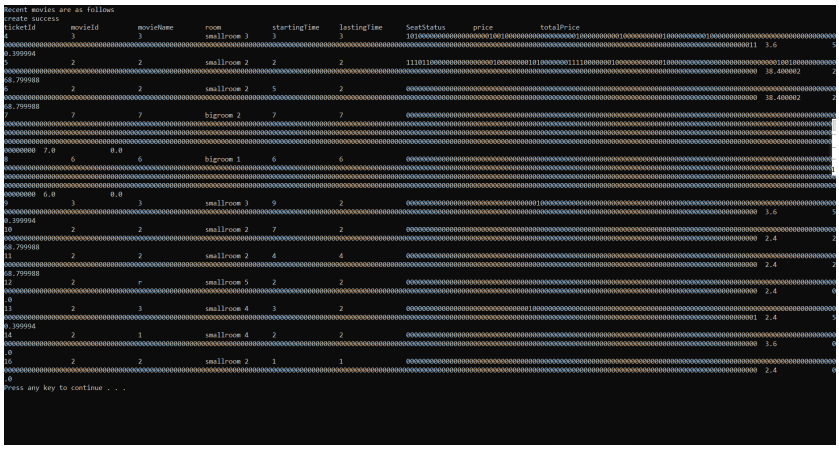

Fig. 2.11: Surfing the movie information

When user input 2,

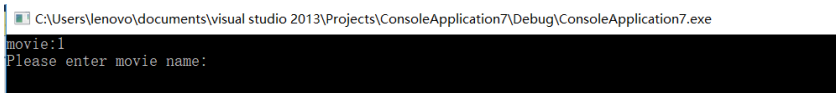

Fig. 2.12: Entering information

Afterwards

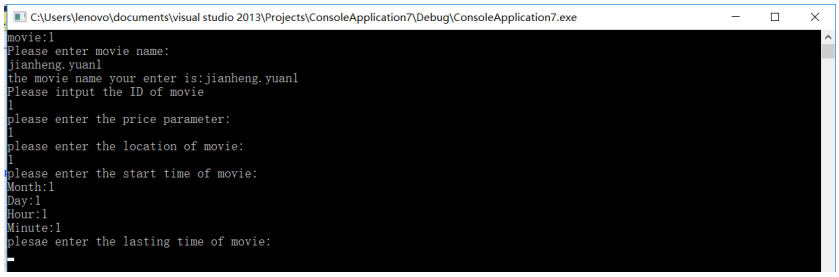

Fig. 2.13: Afterwards

Finally, program will ask you “do you want to add more movie?”

Fig. 2.14: Confirmation

Then user can input the next movie:

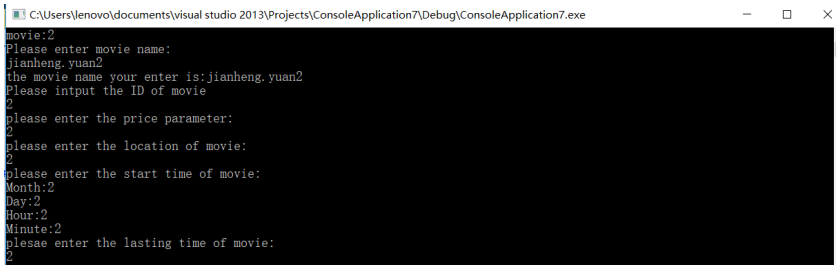

Fig. 2.15: Next Movie

Afterwards:

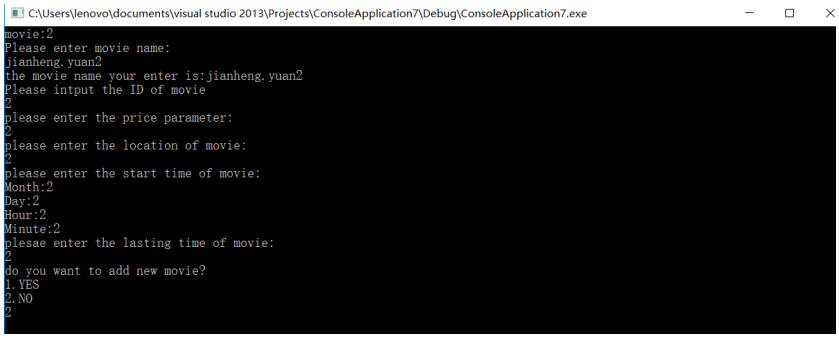

Fig. 2.16: Afterwards

User return to administrator surface:

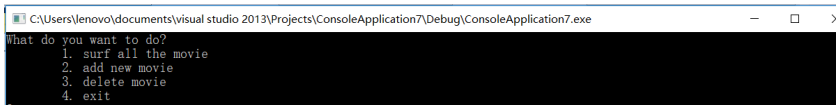

Fig. 2.17: Administrator surface

User can surf the movie information:

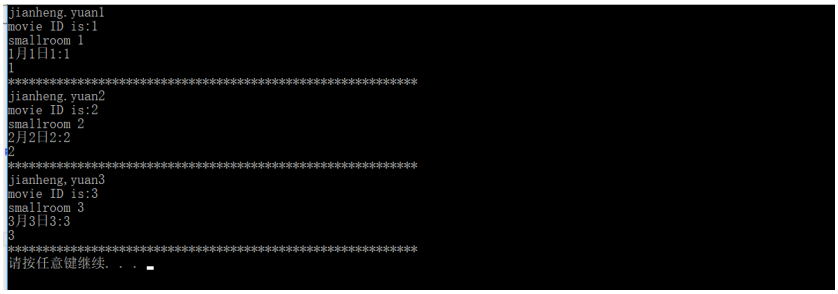

Fig. 2.18: surf the movie information

They can also delete movie information:

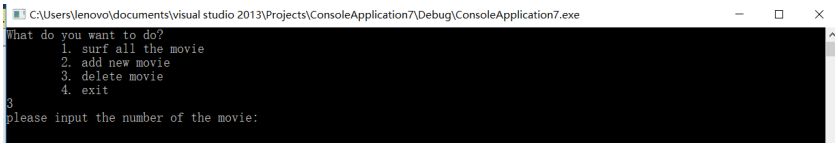

Fig. 2.19: delete movie information

User input 1:

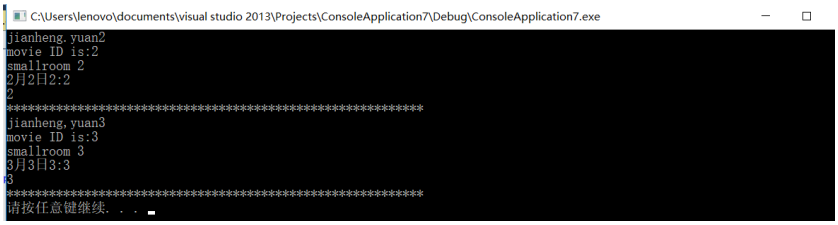

Fig. 2.20: User input 1

However, if the number that user input is bigger than the maximum number of movie:

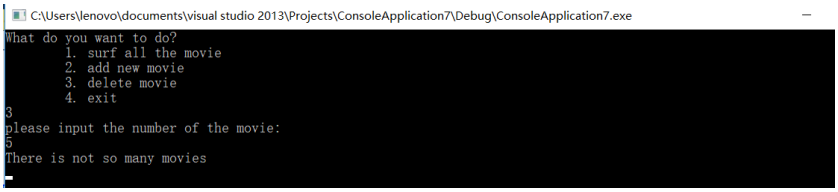

Fig. 2.21: number that user input is bigger than the maximum number of movie:

From the view of user:

Fig. 2.22: From the view of user:

Customer

browse recent movies

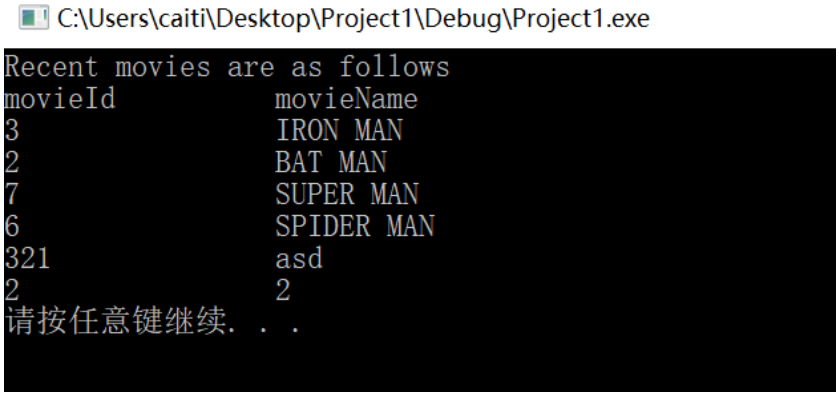

Fig. 2.23: From the view of user:

display the ticket

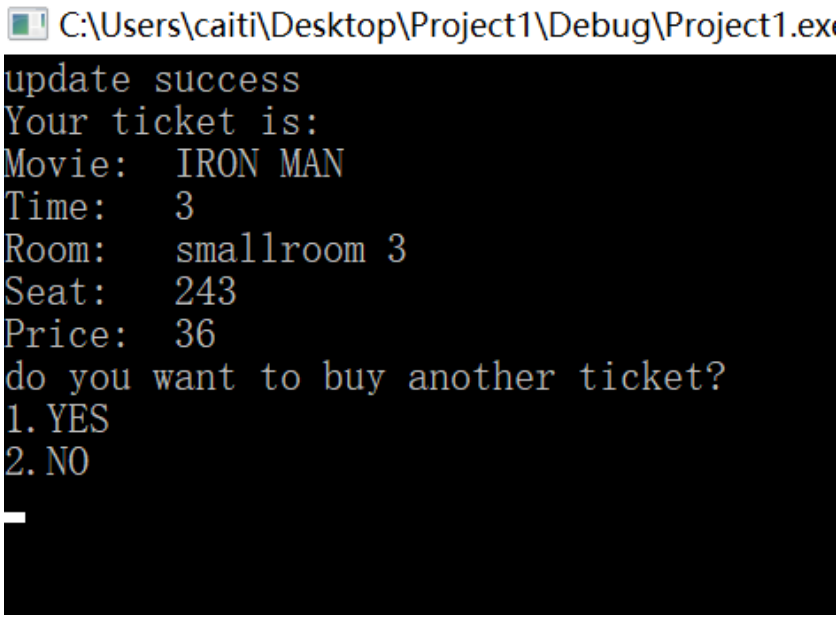

Fig. 2.24: From the view of user:

if the selected seat is occupied, let customer to select again

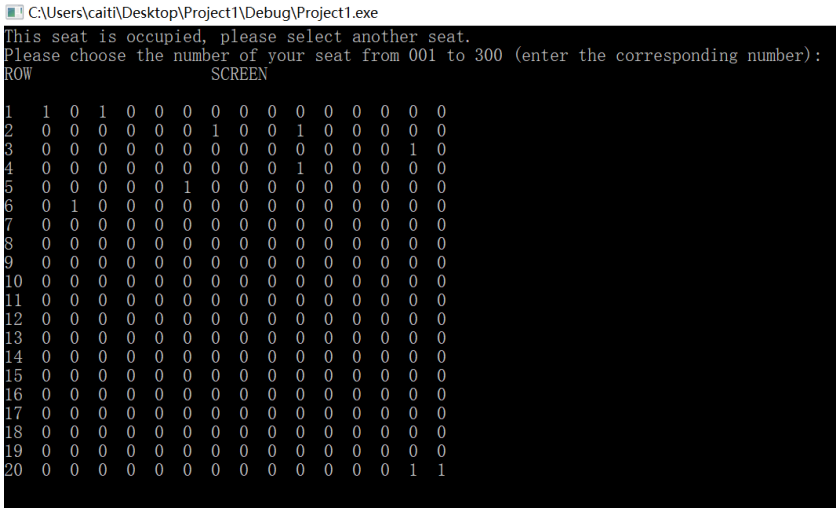

Fig. 2.25: From the view of user:

initial interface

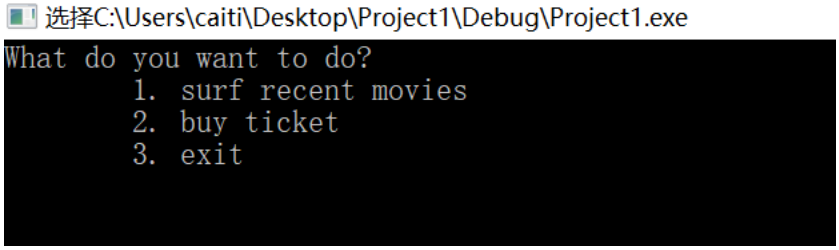

Fig. 2.26: From the view of user:

select movie ID

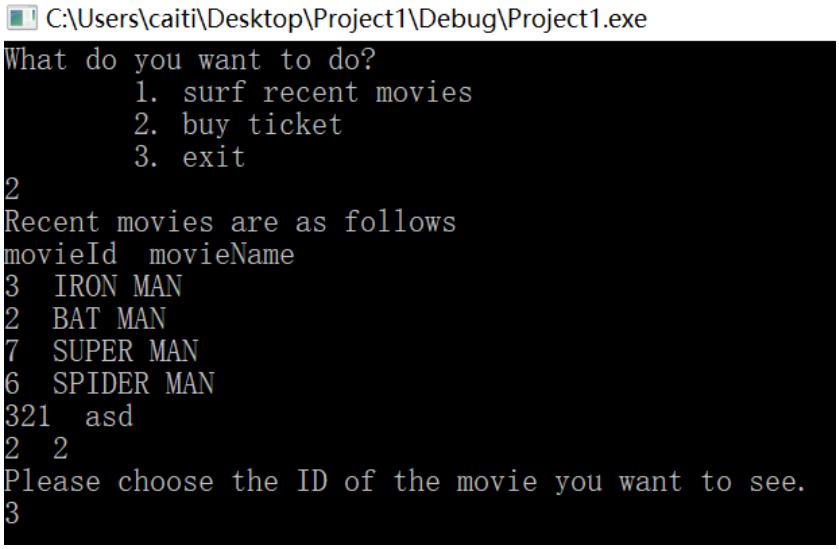

Fig. 2.27: From the view of user:

select the seat

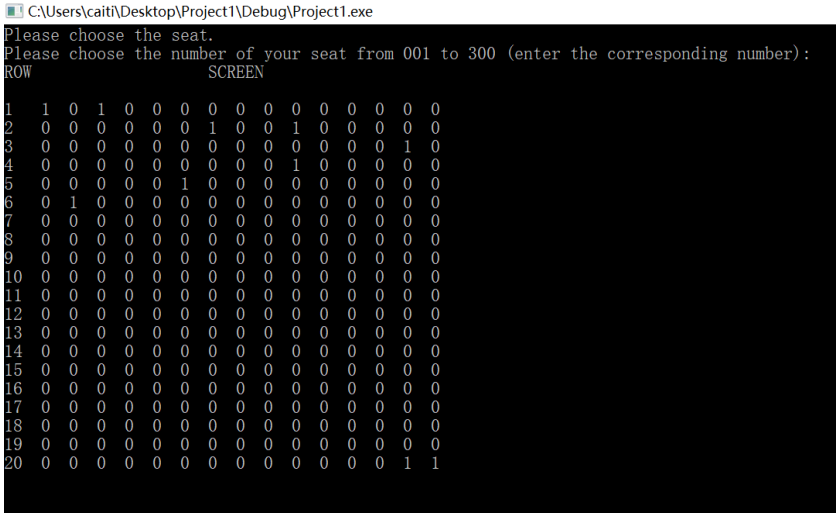

Fig. 2.28: From the view of user:

select ticket ID

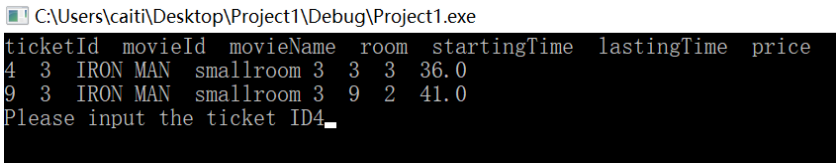

Fig. 2.29: From the view of user:

2.5.2 Robustness Test

robustness of initial interface1

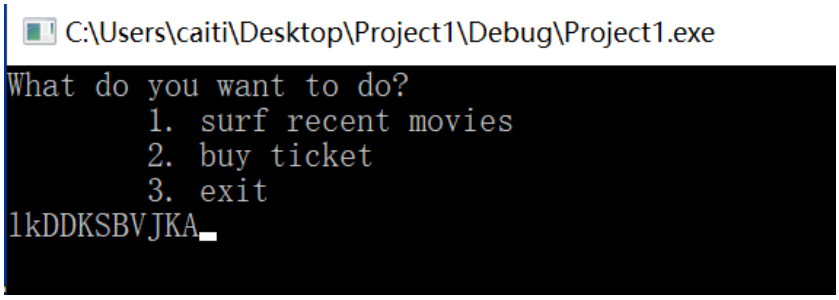

Fig. 2.30: From the view of user:

robustness of initial interface2

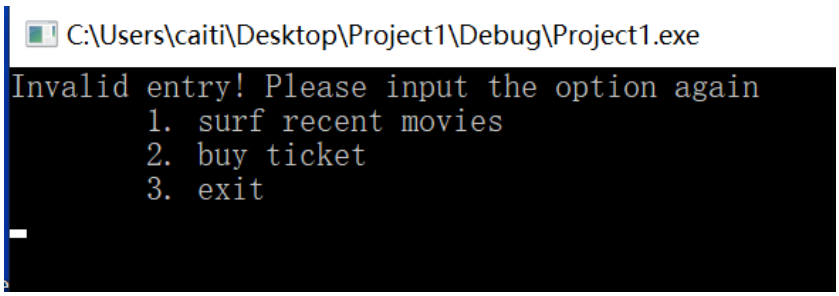

Fig. 2.31: From the view of user:

robustness of select movie ID

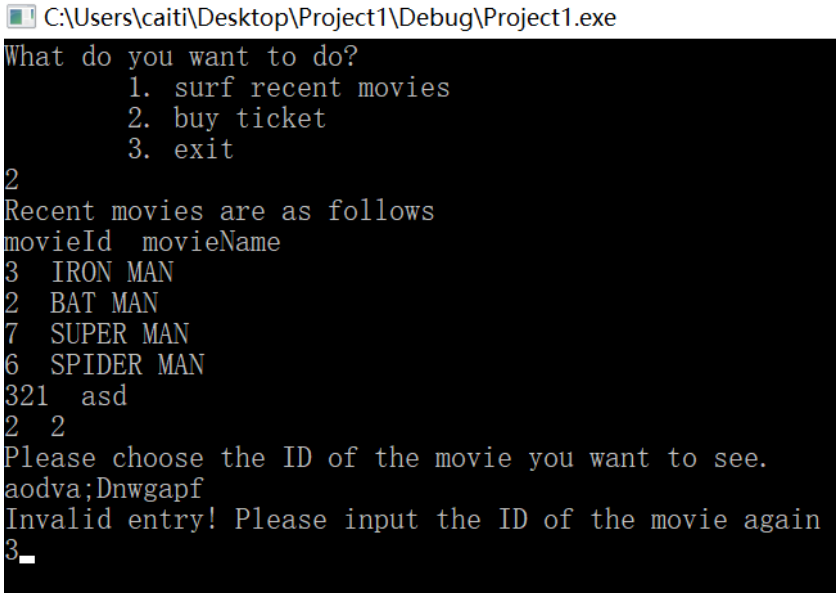

Fig. 2.32: From the view of user:

robustness of select ticket ID

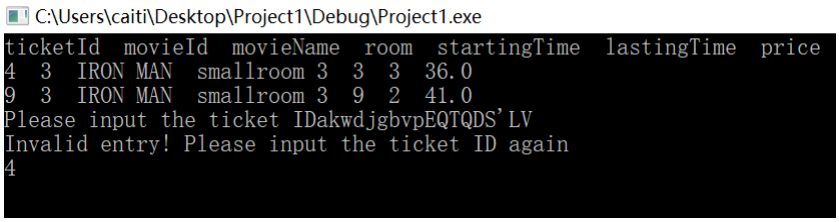

Fig. 2.33: From the view of user:

2.6 Debug and Correct

2.6.1 Bug

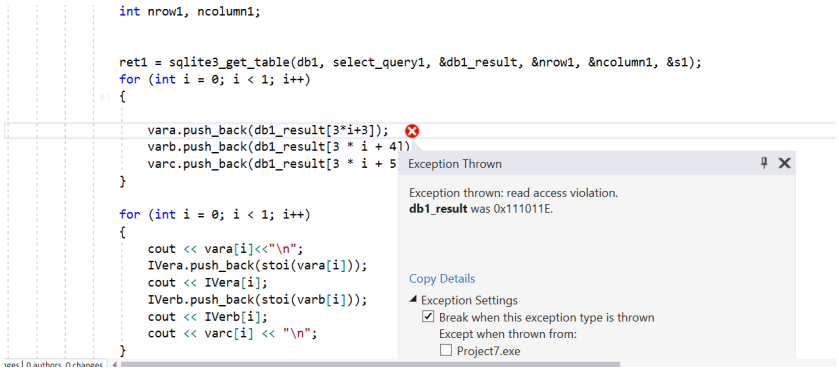

Fig. 2.34: read access violation bug

2.6.2 Correct

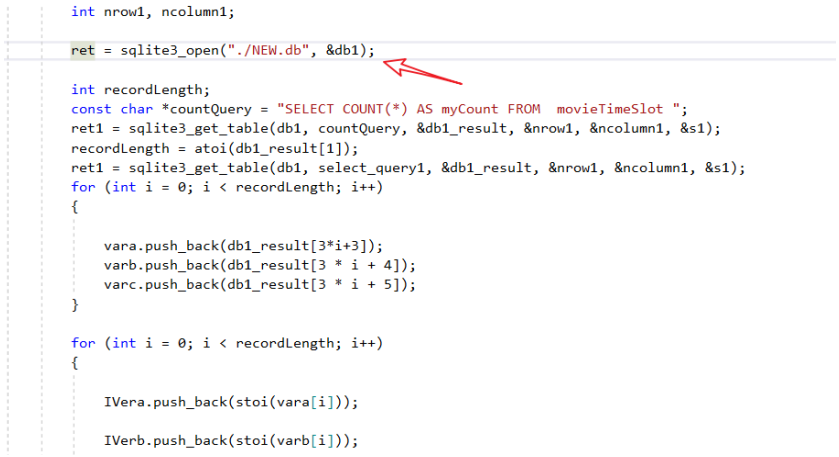

Fig. 2.35: bug correction

Chapter 3

Group Cooperation

3.1 Group Meetings

In this group project, we have arranged six group meetings with the pictures as follows.

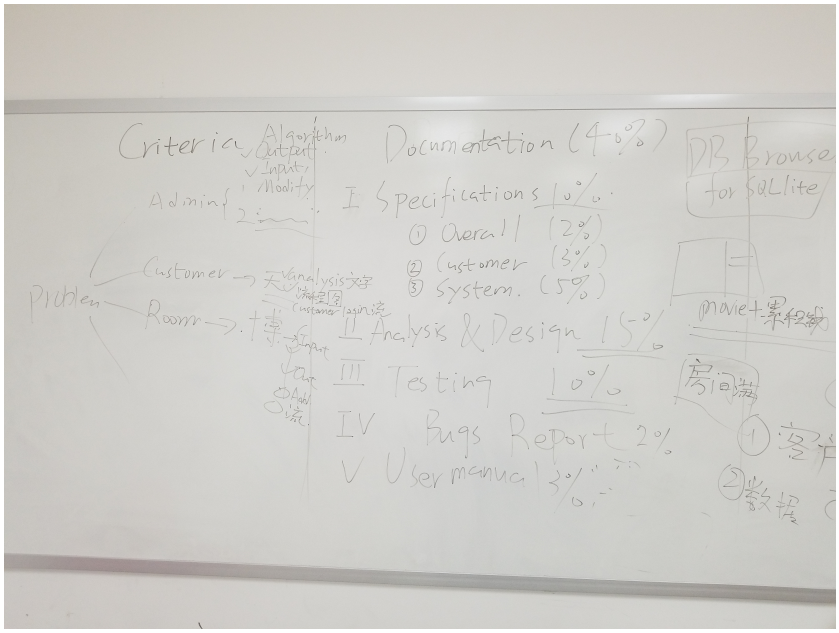

Fig. 3.1: Group Meeting White Board1

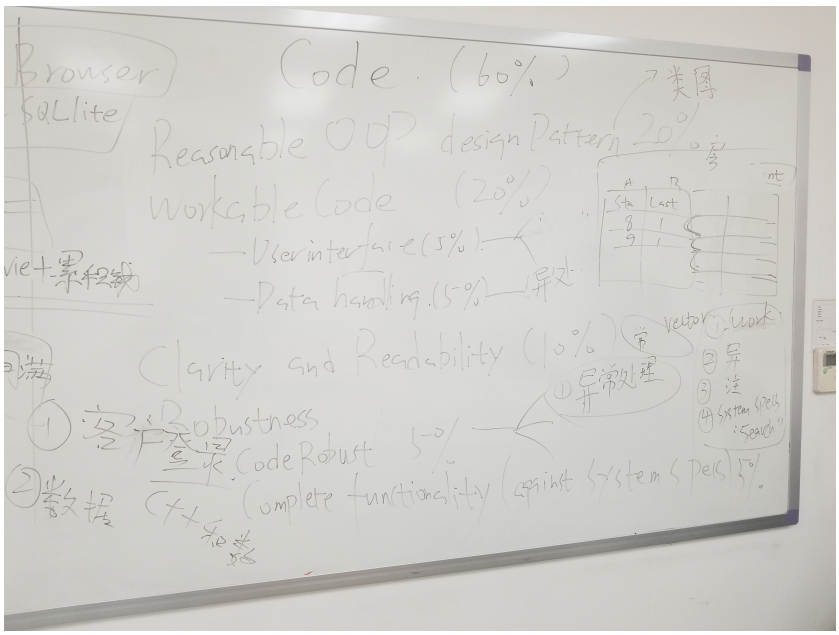

Fig. 3.2: Group Meeting White Board2

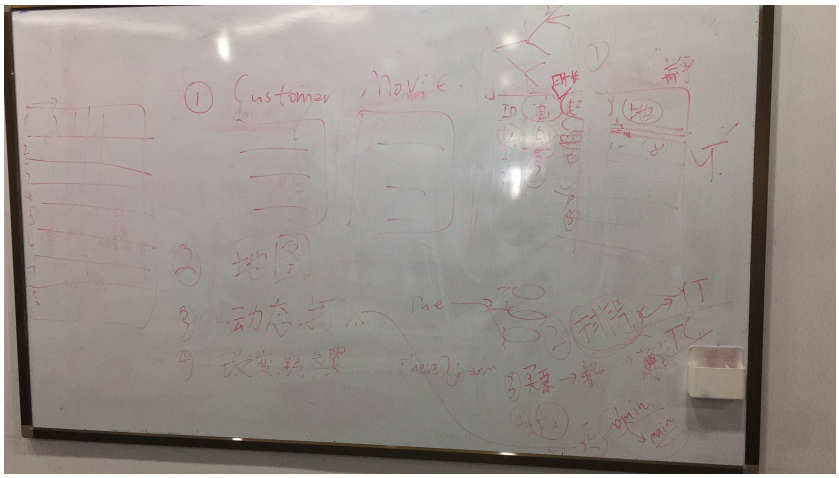

Fig. 3.3: Group Meeting White Board3

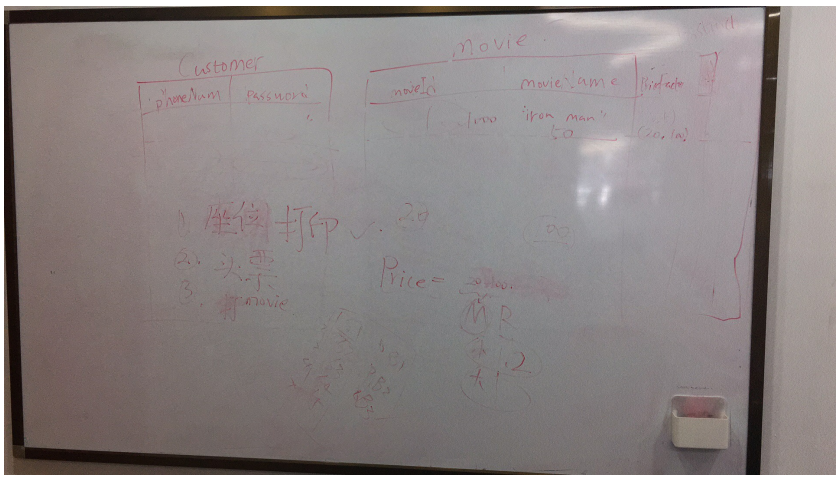

Fig. 3.4: Group Meeting White Board 4

3.2 Source Control With Github

Development Log

date 5.2 by the team

Question confirmation

date 5.4 weekly plan

Join in the github project
yibo: room class with encapsulation, basic main's input procedures
tianyu: customer class with encapsulation, buy tickets
jianheng: movie class with encapsulation
dequn: admin class with encapsulation, build the sql, build manage the movie

date 5.11 by the team

Display the seat
customer 's database: phone number, pw
movie's database: movieId, movieName, PriceFactorM, lastTime
movieTimeSlot: movieId, movieName, lastingTime,startingTime,Room

date 5.11 by dequn and tiaoyu

The test demo of the front end and the back end 1.

date 5.12 by yibo

Fig. 3.5: Github Development Log in READ.ME

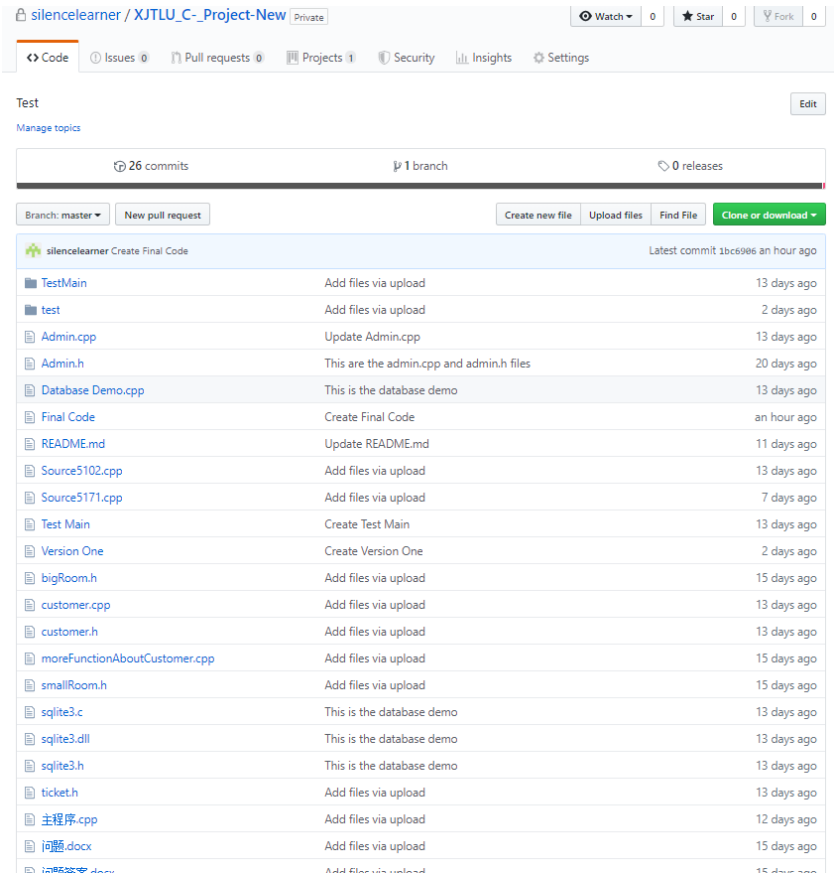

Fig. 3.6: Overview of github responsory

3.3 Progress Control With Worktile

There are four tasks that I am working with which are responsible project, assigned project, created project, and participated project.

- responsible project
- assigned project
- created project
- participated project

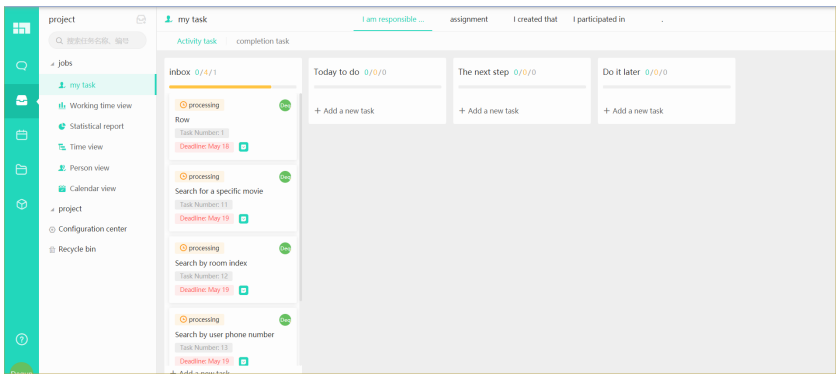

Fig. 3.7: My responsible project

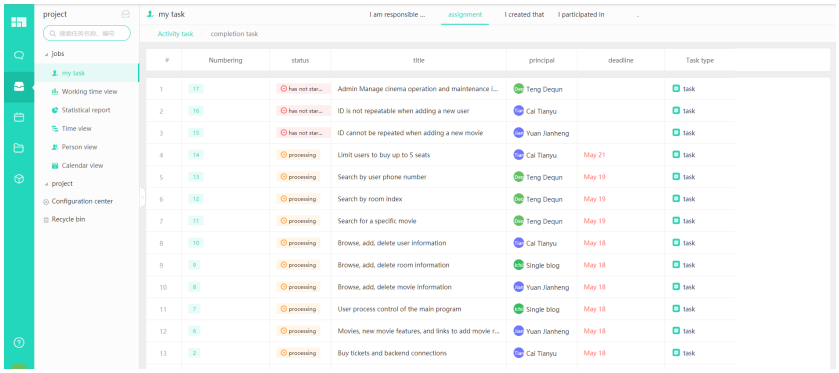

Fig. 3.8: I assigned

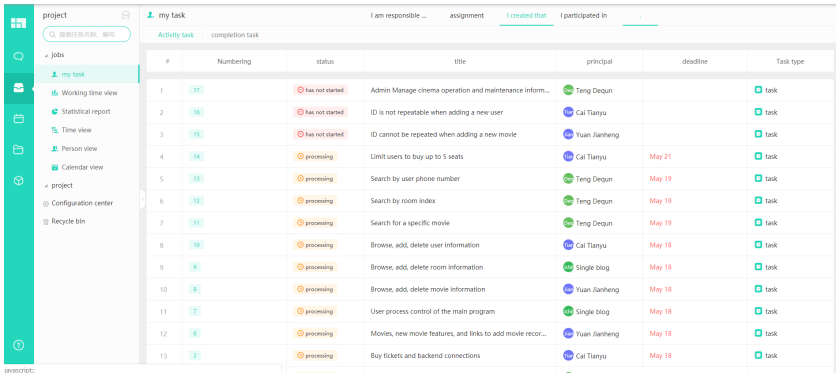

Fig. 3.9: I created

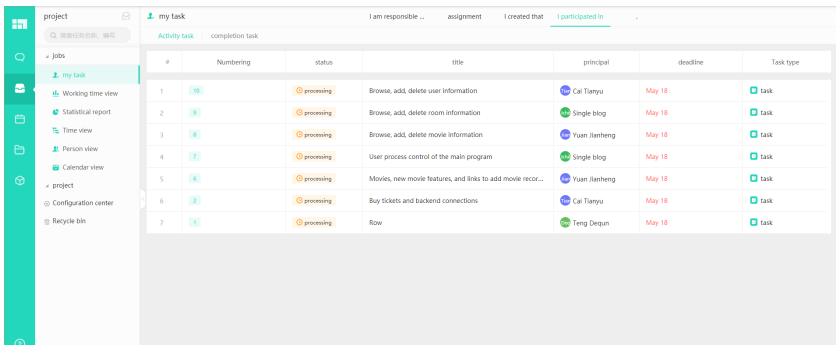

Fig. 3.10: I participated

Chapter 4

Conclusion

To summarize this group project, a movie ticket booking system was built successfully. Firstly, the functions of each identity were simulated successfully, including the administrators, customers and managers, which is the main part of this group project. The administrator is responsible to add, modify and delete the information of room and customer. The customer can view the information of movie and buy tickets according to the chosen room and seat. The function of the manager is to calculate and view the total income of each movie. Secondly, the information of each room can be observed at any time if necessary, which is a significant project in this assessment. In addition, in this group project, many necessary contents were tested and some bugs were occurred, indicating that this program exists limitations, which should be improved next time. Although some limitations existed, the students had made their greatest effort to fulfill this project and the result was acceptable.

Chapter 5

Reference

[1] S. Rabin, Introduction to game development (no. Sirsi) i9781584503774). Charles River Media Boston, 2005.
[2] Stroustrup, Bjarne (1997). ”1”. The C++ Programming Language (Third ed.). ISBN 0-201-88954-4. OCLC 59193992.
[3] Q. Liu , EEE102, Topic: ”Class and objects.”, Xi’an Jiaotong-Liverpool University/Electrical and Electronic Engineering, University, Suzhou. March, 18, 2019.