C Programming and Software Engineering I
Application of Software Development Process
Group-66

Problem statement

Task

Our group has been employed to develop a seat-booking system with two loads which are respectively administer and customer. As an administer, the authorized person not only can regulate the basic information of seats and users, but also inspect the real-time income. If the person who logs in this system is a customer, that person needs to register for an account with legal characters first and starts reservation with specific seat types. For supplementary explanation, customer can cancel booking and recall the individual booking records if he wants.

General Requirements

To better meet the demand of the company, we need to build 3 files for users, seat, and administer which are respectively "users.txt", "seat.txt", admin.txt". In the main function of this software, it will examine whether all of these 3 files exists or not first, and the manager can initialize the system by setting the administer password less than 10 digits (0-2147483647). Next, the manager needs to customize the initial values of the numbers and prices of the seats. Notice that the range of all of these values are between 0 to 65535. In the end, the system menu will appear if these files exist.
Before elucidating the requirements of sub-functions, it is indispensable to define a function of input, for the design of a menu. In the software, apart from information collection, we are allowed to set the upper (max) and lower boundary (min) in choosing which step to go further. For example, in our system menu, we initially need to separate the identities by different inputs, where 1 stands for administer, 2 represents users, and 0 executes the command of exit.
Users are recommended to register their account by inputting id and password in legal ways (0-2147483647). After choosing the identity of user, the user can the choose to register or login. Every user needs to register before login the system. Then, in the User Menu page, the customer can choose to book a seat, view booking information, change booked seat type and booked days (1-1024), cancel the seat, and also return to previous page.
On the other side, managers can log in to the admin menu by typing password written in "admin.txt". In the admin menu, manager has the capability to check seats information, users' information and total income. In addition, the manager can modify total seat numbers, adjust the price of the seats, initialize all the data, and return back to the previous page.

Requirements for users

1. Log in

The user can choose 1 to log in with created count and password, then the user menu will appear. Notice that an unregistered account will not pass the log-in test. Therefore, it is strongly suggested that the users should register their account which is achieved by the next requirement first, and then log in to handle their seats.

a) <u>Book a seat</u>

The user type 1 to book a seat. In the beginning of this page, there will be a table indicating the information about seat-type, reservation, and price (all the table mentioned in problem statement all refers to the seat information table). After that the user can choose between 0-3 to continue the following steps to book the seat for different seat-types or not. Including in the booking steps, the user also needs to input the booking days, and pay for the seat. Additionally, if the user pays more than or less than the price, the system will realize refund and collection.

b) <u>View individual information</u>

The user type 2 to view individual information. The table recording seat information will be shown first, and the booking information with 3 indices which are seat-type, booked days, payment record will be performed after that.

c) <u>Change seat-type or booked days</u>

The user inputs 3 to change seat-type or booked days. The table and booking information will be given for users to refer. Next, the user can change seat-types and or booked days with valid number.

d) <u>Cancel seat</u>

The user inputs 4 to cancel seat. Also, the table and booking information will emerge for users to refer.

2. Create activation

The customer type 2 to create an account with id and password in legal ways (0-2147483647). Remember do it first and log in after that.

3. Go back to the previous page

Enter 0 to return to system menu.

Requirements for administers

1. Check information and total income

The manager can type 1 to enter this page. The table informing seat condition will appear. Considering the identity of the manager, the second information which appears on the screen is the booking record of users with the notation of their ids, seat-type, booking days, and payment record. After that, the total income will be shown on screen.

2. Add or delete total seat numbers

The manage can choose 2 to enter this page. There will be a table same as I mentioned in the proceeding paragraph. Then, the administer can choose between 0-3 to continue the following steps to edit the seat numbers for different seat-types or not. Moreover, if the manager inputs an integer that makes total number of one kind of seat numbers less than zero, the software will return "you cannot delete the number of seats in this type".

3. Adjust the price of seats

The administer can input 3 to enter this page. The table I covered before will be presented to the administer to display the condition of seats. Subsequently, the manager can choose between 0-3 to continue the following steps to edit the price for different seat-types or not. Having chosen a specific seat-type, the administer needs to modify the price by inputting numbers between 1 to 99999. The software will return "please enter a alid number" until the legal input is detected.

4. Initialize all the data

The administer can input 4 to enter this page. This page is designed for those who want to clear the existing data, and begin with the new system. The manager can choose to type 1 for sure and 0 for cancel.

5. Exit

If the administer type 0, the system will go back to the previous page which is system menu.

Analysis

Structure

1. User

Given that the number of users is not a fixed value, so we need to add or delete new terms of users by utilizing linked list with a structure pointer.

2. Seat

Considering the demand to deal with information like seat-number, seat-price, seat-type and number of booked seat, we set a structure for seat.

Function

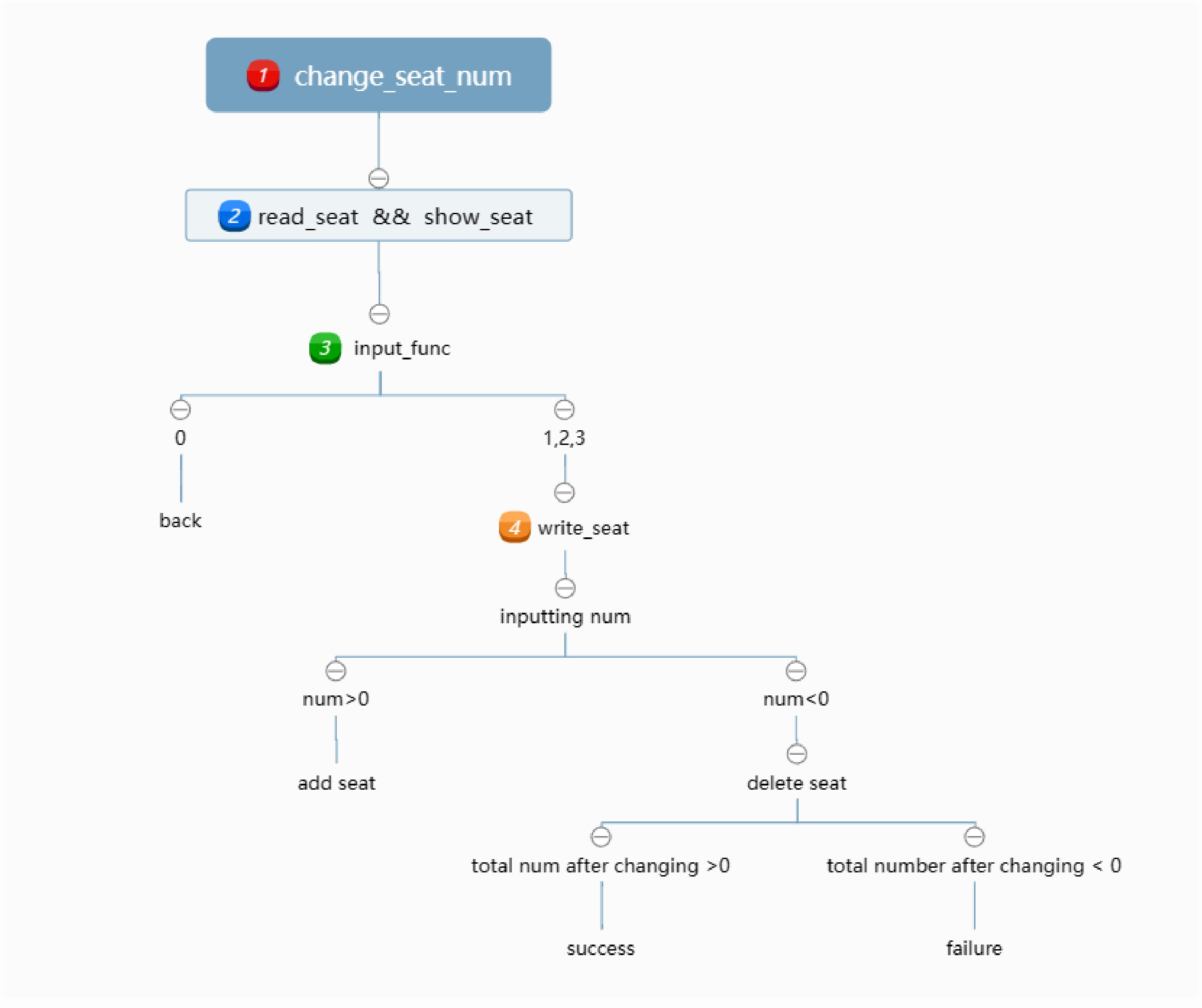
We only select the core functions we've written to explain.

1. Input function

The function allows the user to enter a number within a given range. It takes two parameters, min and max, representing the minimum and maximum values allowed for the input.
Inside the function, the variable input is initially set to an invalid value, which is "$\mathrm{min} - 1$". Then, it prompts the user to enter a number using the "$printf$" and "$scanf$" functions. The function then checks if the input is within the valid range using the condition while $(\mathrm{input} < \mathrm{min} \,||\, \mathrm{input} > \mathrm{max})$. If the input is outside the valid range, it clears the input buffer using while $(\mathrm{getchar}() != '\backslash n')$; and prompts the user again to enter a valid number. This process continues until a valid number is input. Finally, the function returns the valid input number using the return statement.
Overall, the function allows the user to enter a number within a given range and ensures that only valid inputs are accepted.

2. Change seat number--administer

The function change_seat_num allows the user to change the seat number in a given range. It performs several operations such as reading the seat type, displaying the seat type, prompting the user to choose a seat, and allowing the user to add or delete a seat number.
It is more direct to observe the flowchart shown as below.



Here's an illumination of the flowchart:

a) <u>Change seat number</u>

It is the full name of the function, also indicates the purpose of this function.

b) <u>Information gain</u>

The function starts by calling the read_seat function to read the seat-type. It then calls the show_seat function to display the seat type.

c) <u>Choose</u>

It prompts the user to choose a seat by displaying an options menu. The user enters a choice, and the function stores it in the choose variable. If the user chooses to go back, it clears the console and displays a message saying they've returned to the previous page.

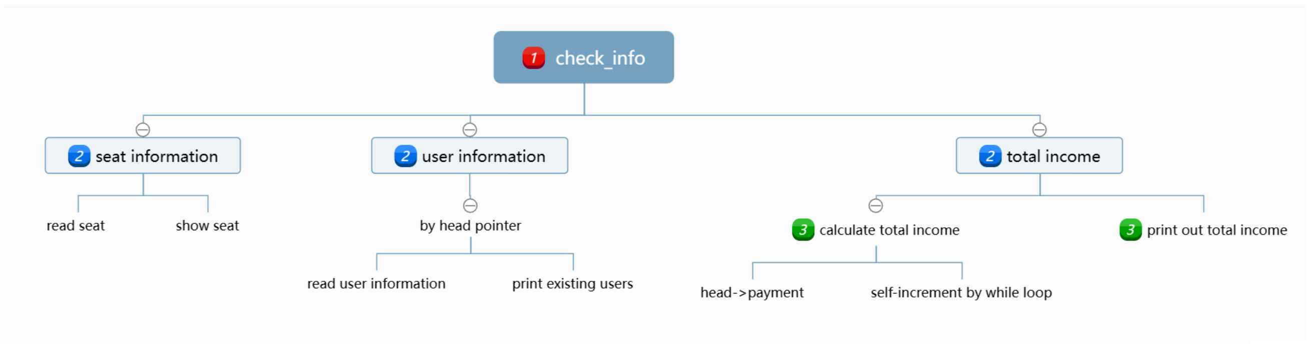
d) <u>Inputting changing value</u>

If the user chooses to add a seat, it prompts them to enter the number they want to add. The function checks if the entered number is positive and stores it in the num variable. If the number is positive, it adds it to the chosen seat number in the seat_type array and writes the changes to the file. It then clears the console and displays a success message. If the user chooses to delete a seat, it calculates the remaining seat number after subtracting the number already booked. It checks if the remaining number is non-zero and if the number the user wants to delete is less than or equal to the remaining number. If both conditions are met, it subtracts the number from the chosen seat number, writes the changes to the file, clears the console, and displays a success message. If the number they want to delete is greater than the remaining number, it displays a message saying they cannot delete that number of seats and prompts them to press enter to go back. It then clears the console. If the user enters zero, it displays a message saying they didn't do anything and prompts them to press enter to go back. It then clears the console.

Overall, this function allows users to modify seat numbers within a given range based on their choice of seat and input value.

3. Check information and total income—administer

The function check_info() performs several tasks related to checking and displaying information about users and their seat bookings. It reads the seat type and shows the seat type. It also reads the user information, prints the existing users, calculates the total income, and displays it.



Here's an illumination of the flowchart:

a) Check information

It is the full name of the function, also indicates the purpose of this function.

b) Realization of the function by sub-functions

i. The function first reads the seat type using the read_seat(seat_type) function. It then shows the seat type using the show_seat(seat_type) function.

ii. The function defines a pointer head to a struct users for user data and a pointer name for the name associated with the seat type. income. The function then reads the user data using the read_users(&head) function, which populates the linked list of user structures pointed to by head. The function then prints the existing users while traversing the linked list. For each user, it checks the seat type and assigns a name to the name pointer based on the seat type. It then prints the user's data, including their user ID, seat type, booked days, and payment record. It also calculates the total income by summing the payment for each user.

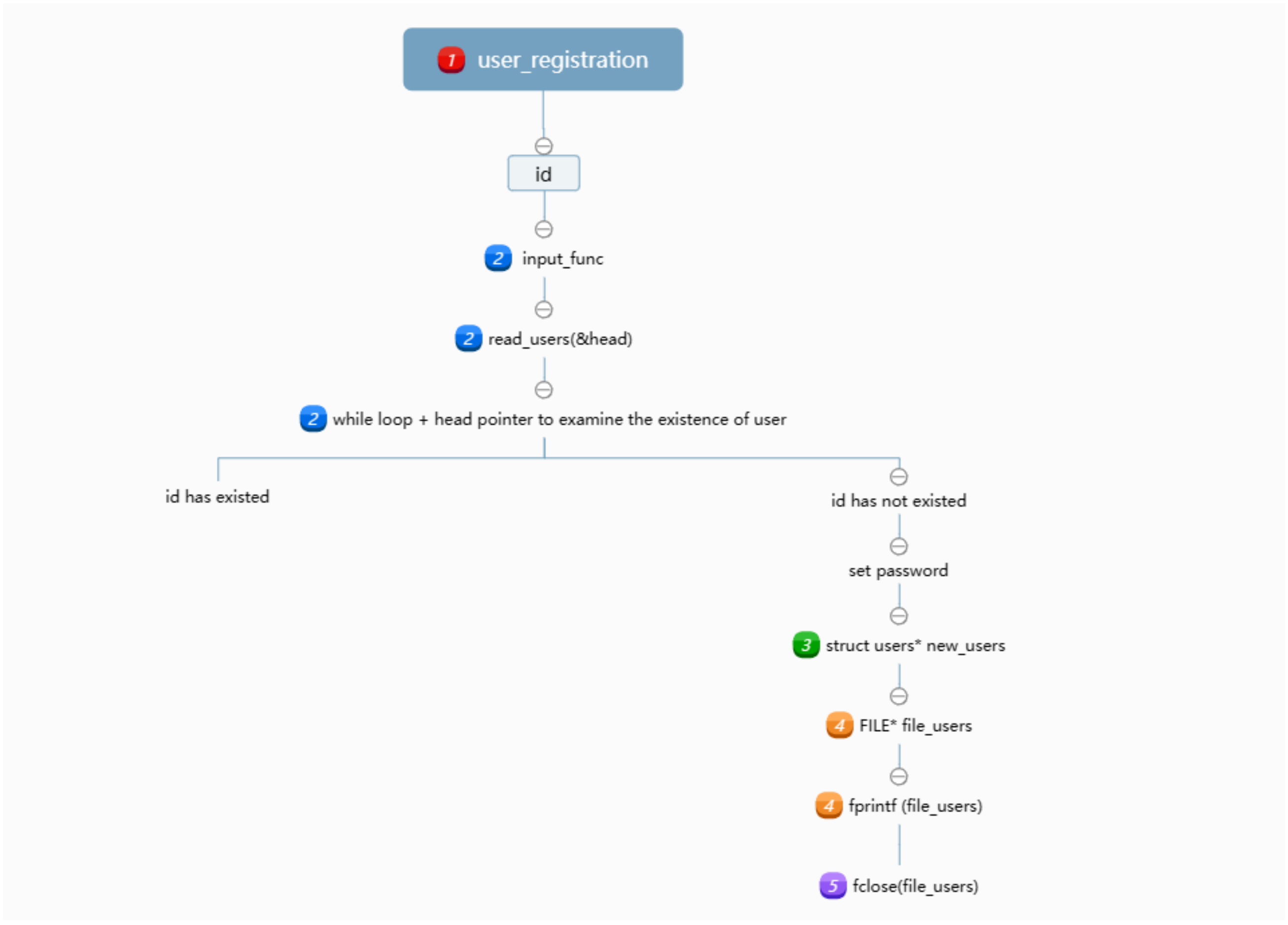
iii. The information collected above can be used to calculate the total income.

c) Total income

The function initializes a variable total_income to 0 and use a while loop to accumulate the payment by each user to calculate the total income. Finally, the function prints the total income.

Overall, this function provides a way to check and display user data, including seat bookings and payment records, and calculate the total income for all bookings.

4. User registration—user



The function "users_registration" is designed to handle user registration. It executes several tasks to check and register the accounts of users. It also uses the "fopen_s" function to open the "users.txt" file in append mode, ensuring that the file is created if it doesn't already exist. If any errors occur during the process, appropriate error messages are displayed.

Here's an illumination of the flowchart:

a) <u>User registration</u>

It is the name of the function, also indicates the purpose of this function.

b) <u>Choose and information receive</u>

The software uses the "input_func" function to get user input and performs range checking to ensure that the ID and password are within valid ranges. This function uses a linked list to store all registered user information. The "read_users()" function reads all user data from the file and adds them to the linked list. Then, the function uses a while loop to iterate through the linked list, looking for users that match the input ID.

c) <u>New users</u>

Next, the function asks the user to set a password and allocates memory space for the new user using the "malloc()" function.

d) <u>User information</u>

Then, the function uses the "fopen_s()" function to open the file "users.txt", uses the "fprintf()" function to write the user information to the file, and adds the new user information to the end of the file.

e) <u>End</u>

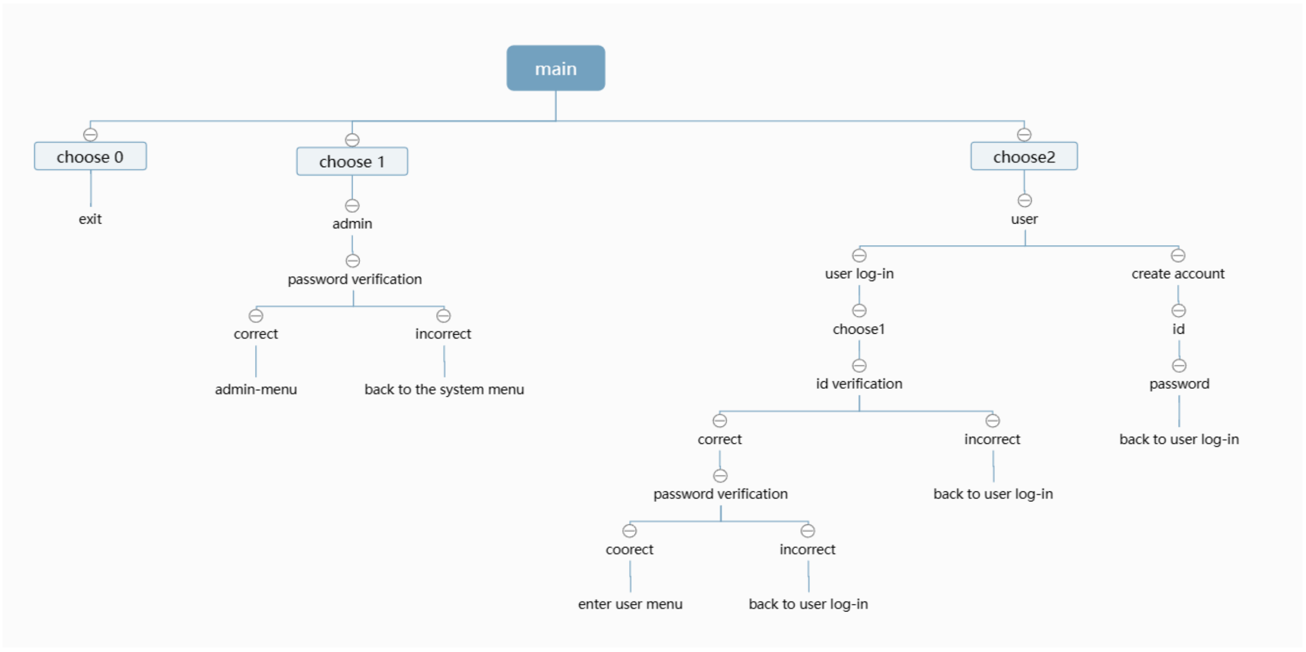
Finally, the function closes the file and displays a message indicating successful registration.

Overall, this function allows a user to register with a unique ID and password, ensuring that duplicate IDs are not used. The information is stored in a text file for future reference.

Design

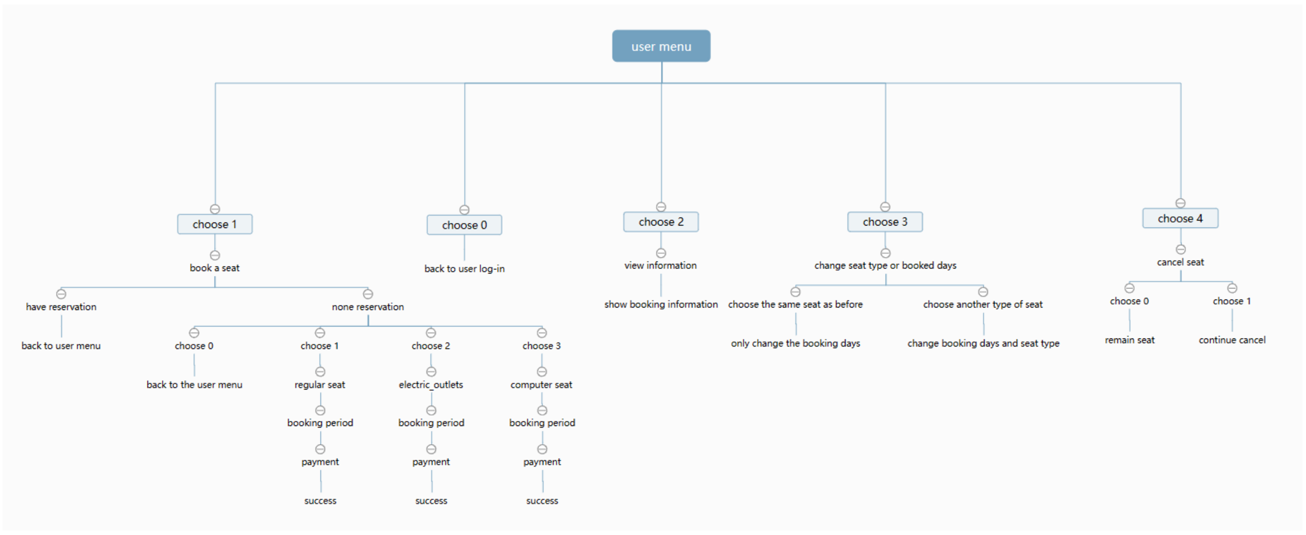
In this part, we would like to give an overview of our software by a combination of flowcharts and description with the sequence of menu nesting.

System menu – identity selection



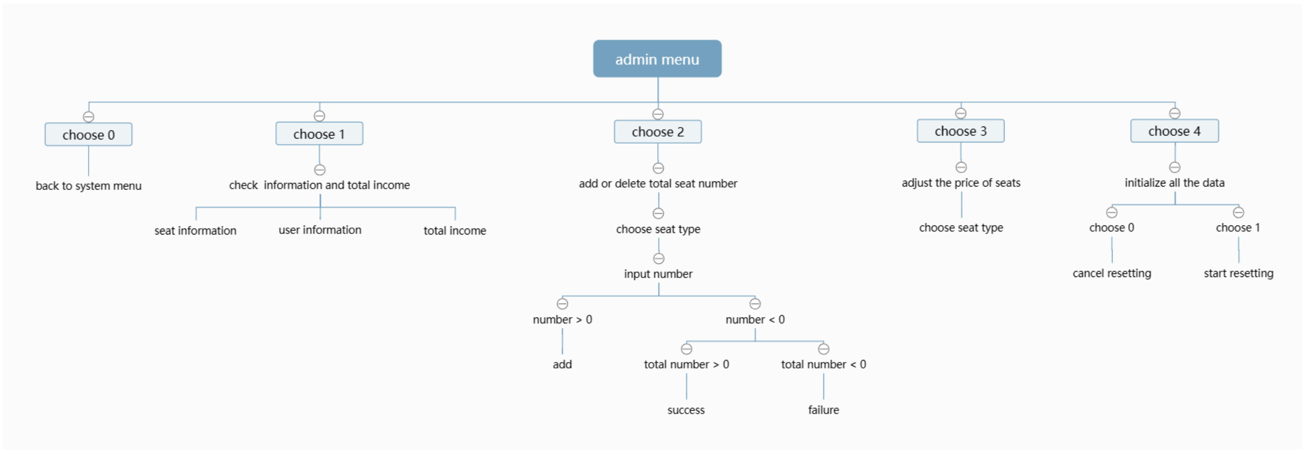
In the system menu, the individual can choose different numbers to log-in as different identities. If the logger is a administer, that person can input the password to enter admin-menu. If the customer who has registered account before, he or she only needs to enter the correct account and password to get into the user menu. If the user has not possessed an account, he or she needs to activate their account first, and logs in with the newly created account.

User menu



In the user menu, user can input various numbers to realize different functions. If they want to book a seat, we consider whether they have reservation before as a test. If they would like to change seat types, we also divide the input into two situations, which are same seat type and other seat type.

Administer menu



In administer menu, the administer can choose to check and modify the business strategy of this software. What needs to be accentuated is that if the delete number makes the total number of the seats negative, the software will detect the error and set the administer to the previous page.

TESTING

Function name	Aim	Input	Expected	Got
init_system	Set admin password.	1. abcde 2. 1234567899 3. 123	1. please enter a valid number 2. please enter a valid number 3. Pass	1. Correct 2. Correct 3. Correct
	Set seats number (0-65535)	1. 0 2. 600 3. 65535 4. 65536	1. Pass 2. Pass 3. Pass 4. please enter a valid number	1. Correct 2. Correct 3. Correct 4. Correct
	Set price number (0-65535)	1. 0 2. 600 3. 65535 4. 65536	1. Pass 2. Pass 3. Pass 4. please enter a valid number	1. Correct 2. Correct 3. Correct 4. Correct
Admin login	Check whether the input password is equal to set password or not.	1. Correct password 2. Wrong password 3. Non-digit	1. Pass 2. back to previous page. 3. please enter a valid number	1. Correct 2. Correct 3. Correct
Check_info	Check information and total income.	Void	Void	Information shown
Change seat number	Choose on type of seat and change its number	1. Valid number 2. More than 65536 3. Delete number larger than remain number	1. Adding success 2. please enter a valid number 3. you cannot delete this amount of seat	1. Correct 2. Correct 3. Correct
Change seat price	Choose on type of seat and change its price	1. Valid number 2. More than 65536	1. Adding success 2. please enter a valid number	1. Correct 2. Correct
Create account	Set valid id and password	1. No more than 9 valid digits 2. More than 9 numbers 3. Non-numeric	1. Pass 2. please enter a valid number 3. please enter a valid number	1. Correct 2. Correct 3. Correct
Book seats	Book one type of seats	1. Valid number 2. More than 1024 (the upper limitation)	1. Pass 2. please enter a valid number	1. Correct 2. Correct
Check information	Check the seat quantity	Void	Void	Information shown
Book seats	Book one type of seats	1. Valid number 2. More than 1024 (the upper limitation)	1. Pass 2. please enter a valid number	1. Correct 2. Correct

change seats	1. change the type of booked seats 2. Change the quantity of booked seats	1. Same type but different days 2. Different type of seats	1. Success and check out 2. Success and check out	1. Correct 2. Correct
Cancel seats	Cancel the booked seats	1. 1 2. 0	1. Successfully Cancel the information 2. Return back	1. Correct 2. Correct

C code

```c
#define NO_BOOK 0;
#define REGULAR 1;
#define ELECTRICAL_OUTLETS 2;
#define COMPUTER 3;
#include<stdio.h>
#include<stdlib.h>
#include <string.h>

/*-------------prototype--------------*/

void main();   /*  start from main function  */

void admin_login();  /*  manager functions  */
void admin();
void check_info();
void init_system();
void change_seat_num();
void change_seat_price();

void users();     /*  user functions  */
void users_registration();
void users_login();
void users_menu(int);
void change_seat(int, int);
void booking_seat(int);


/*------------structure-------------*/

struct seat {
    int seat_num;   /*  total seat munber  */
    int price;   /*  present price  */
    int type;   /*  three types  */
    int Booked;   /*  Booked number  */
};
struct seat seat_type[3];  /*  global structure variable for seat  */

struct users {
    int users_id;
    int users_password;
    int payment;   /*  payment record  */
    int seat_type;   /*  booked seat type  */
    int days;   /*  booked days  */
    struct users* next;   /*  link list  */
};

/*-------------Tool function--------------*/

int input_func(int min, int max)   /*  input function, called by pass the range  */
{
    int input = min - 1;   /*  initialize the input to a invalid value  */
    printf("input:");
    scanf_s("%d", &input);

    while (input < min || input > max)   /*  if input is invalid  */
    {
        while (getchar() != '\n');   /*  clear input buffer  */
        printf("\nplease enter a valid number\ninput:");
        scanf_s("%d", &input);
    }
    return input;   /*  return a valid value  */
}

int check_file_exist()   /*  check if three files all exist  */
{
    FILE* file_seat;
    FILE* file_users;
    FILE* file_admin;
    int count = 3;   /*  once a file exist, count--  */
    if (fopen_s(&file_users, "users.txt", "r") == 0) {
        fclose(file_users);
        count--;
    }
    if (fopen_s(&file_seat, "seat.txt", "r") == 0) {
        fclose(file_seat);
        count--;
    }
    if (fopen_s(&file_admin, "admin.txt", "r") == 0){
        fclose(file_admin);
        count--;
    }
    if (count == 0)   /*  if all exist  */
    {
        return 0;
    }
    else   /*  at least one file not exist  */
    {
        return 1;
    }
}

void read_seat(struct seat* seat_type)   /*  read seat file  */
{
    FILE* file_seat;
    int err = fopen_s(&file_seat, "seat.txt", "r");
    if (err != 0)   /*  exit if file not exist  */
    {
        printf("Can't find <seat.txt> file\n");
        exit(1);
    }
    for (int i = 0; i < 3; i++)   /*  read from txt  */
    {
        fscanf_s(file_seat, "%d %d %d %d",
            &seat_type[i].seat_num,
            &seat_type[i].price,
            &seat_type[i].type,
            &seat_type[i].Booked);
    }
    fclose(file_seat);
}

void show_seat(struct seat* seat_type)   /*  show present seat condition  */
{
    printf("---------------------------------------------------------------\n");
    printf("SEAT TYPE          TOTAL NUMBER     REMAIN NUMBER    PRICE/day\n");
    printf("regular                 %d                 %d            %d \n",
        seat_type[0].seat_num,
        seat_type[0].seat_num - seat_type[0].Booked,
        seat_type[0].price);
    printf("electric_outlets        %d                  %d            %d \n",
        seat_type[1].seat_num,
        seat_type[1].seat_num - seat_type[1].Booked,
        seat_type[1].price);
    printf("computer                %d                  %d            %d \n\n",
        seat_type[2].seat_num,
        seat_type[2].seat_num - seat_type[2].Booked,
        seat_type[2].price);
}


void write_seat(struct seat* seat_type)   /*  write in new data to seat file  */
{
    FILE* file_seat;
    int err = fopen_s(&file_seat, "seat.txt", "w");
    if (err != 0)   /*  exit if create file fail  */
    {
        printf("Can't create file\n");
        exit(1);
    }
    for (int i = 0; i < 3; i++)   /*  cover new data into txt  */
    {
        fprintf(file_seat, "%d %d %d %d\n",
            seat_type[i].seat_num,
            seat_type[i].price,
            seat_type[i].type,
            seat_type[i].Booked);
    }
    fclose(file_seat);
}

void read_users(struct users** head)   /*  read user data and return  */
{
    FILE* file_users;
    int err = fopen_s(&file_users, "users.txt", "r");
    if (err != 0)   /*  if file not exsit  */
    {
        printf("File <users.txt> not exist\n");
        exit(1);
    }
    *head = NULL;   /*  dereference pointer of pointer */
    struct users* previous = NULL;
    while (!feof(file_users))   /*  read while not reach the end  */
    {
        struct users* new_users = (struct users*)malloc(sizeof(struct users));
        if (new_users == NULL)   /*  check if malloc success  */
        {
            printf("Memory allocation failed\n");
            exit(1);
        }
        if (fscanf_s(file_users, "%d %d %d %d %d",
            &new_users->users_id,
            &new_users->users_password,
            &new_users->payment,
            &new_users->seat_type,
            &new_users->days) == 5)   /*  five data all exsit  */
        {
            if (*head == NULL && previous == NULL)   /*  link list  */
            {
                *head = new_users;
                previous = new_users;
                previous->next = NULL;
            }
            else
            {
                *head = new_users;
                new_users->next = previous;
                previous = new_users;
            }
        }
    }
    fclose(file_users);
}

void write_users(int id, int days, int choose, int payment)   /*  change specific user data  */
{
    struct users* head = NULL;
    read_users(&head);
    struct users* current = head;
    while (current != NULL)   /*  change data of specific uesr  */
    {
        if (current->users_id == id) {
            current->days = days;
            current->seat_type = choose;
            current->payment = payment;
            break;
        }
        current = current->next;
    }

    FILE* file_users;
    int err = fopen_s(&file_users, "users.txt", "w");
    if (err != 0)   /*  create file fail  */
    {
        printf("Can't create <user.txt> file\n");
        exit(1);
    }
    current = head;
    while (current != NULL)   /*  cover link list into txt  */
    {
        fprintf(file_users, "%d %d %d %d %d\n",
            current->users_id,
            current->users_password,
            current->payment,
            current->seat_type,
            current->days);
        current = current->next;
    }
    fclose(file_users);
    free(head);
}

/*-------------------User Menu--------------------*/

void change_seat(int id, int menu_choose)   /*  view booked record, change booked days and seat type,
cancel booking  */
{
    struct users* users_list = NULL;
    read_users(&users_list);   /*  get all users' information  */

    int user_seat_type = 0;
    int user_days = 0;
    int record_payment = 0;

    while (users_list != NULL)   /*  find specific user  */
    {
        if (users_list->users_id == id)   /*  copy condition  */
        {
            user_seat_type = users_list->seat_type;
            user_days = users_list->days;
            record_payment = users_list->payment;
            break;
        }
        users_list = users_list->next;
    }

    if (user_seat_type == 0)   /*  if the user haven't booked a seat, go back  */
    {
        system("cls");
        printf("you haven't booked a seat\n");
    }
    else   /*  the user have booked a seat  */
    {
        char* name = NULL;   /*  transform number to string  */
        if (user_seat_type == 1) name = "regular";
        if (user_seat_type == 2) name = "electrical outlets";
        if (user_seat_type == 3) name = "computer";

        read_seat(seat_type);
        show_seat(seat_type);

        printf("\nyour information\n");   /*  show user information  */
        printf("seat type: %s\n", name);
        printf("booked days: %d\n", user_days);
        printf("payment record: %d$\n\n", record_payment);

        if (menu_choose == 2)   /*  show information  */
        {
            while (getchar() != '\n');
            printf("press enter for back:");
            while (getchar() != '\n');
            system("cls");
        }
        else if (menu_choose == 3)   /*  change seat  */
        {
            printf("\nwhich seat type do you want to change to?\n(choose the same seat for change
days)\n");   /*  choose 0-4 */
            printf("--enter 1 for regular\n--enter 2 for electric_outlets\n");
            printf("--enter 3 for computer\n--enter 0 for back\n");
            int choose = input_func(0, 3);
            if (choose == 1) name = "regular";   /*  transform number to string again  */
            if (choose == 2) name = "electrical outlets";
            if (choose == 3) name = "computer";
            if (choose != 0)
            {
                printf("\nplease enter the days you want to book (1-1024)\n");
                int days = input_func(1, 1024);

                if ((seat_type[choose - 1].seat_num - seat_type[choose - 1].Booked) != 0 || choose ==
user_seat_type)   /*  chosen seat have remain or choose original seat*/
                {
                    (seat_type[user_seat_type - 1].Booked)--;   /*  original booked number -1  */
                    (seat_type[choose - 1].Booked)++;   /*  new booked number +1  */
                    int total_price = days * seat_type[choose - 1].price;
                    write_users(id, days, choose, total_price);   /*  change user data  */
                    write_seat(seat_type);   /*  change seat data  */

                    int diff_price = record_payment - total_price;   /*  calculate the difference of
record payment and present price  */
                    if (diff_price > 0)   /*  Refund  */
                    {
                        while (getchar() != '\n');
                        printf("\nRefund %d$\npress enter for sure:", diff_price);
                        while (getchar() != '\n');
                    }
                    else if (diff_price < 0)   /*  Pay for difference  */
                    {
                        while (getchar() != '\n');
                        printf("\nyou need to pay %d$ more\npress enter for sure:", -diff_price);
                        while (getchar() != '\n');
                    }
                    else   /*  same price  */
                    {
                        while (getchar() != '\n');
                        printf("\nSame price, you don't need to pay anymore\npress enter for sure:");
                        while (getchar() != '\n');
                    }
                    system("cls");
                    printf("you have changed your seat!\n\nseat type: %s\n", name);   /*  show changed
information  */
                    printf("booked days: %d\npayment: %d$\n\n", days, total_price);

                }
                else   /*  seat no remain  */
                {
                    system("cls");
                    printf("seat type has no remain, please choose another seat\n\n");
                }
            }
```

```c
            else
            {
                system("cls");
                printf("back to previous page\n\n");
            }

        }
        else if (menu_choose == 4)   /*  cancel seat  */
        {
            printf("\nAre you sure you want to cancel your seat?\n");
            printf("enter 1 for sure\nenter 0 for back\n");
            int sure = input_func(0, 1);
            if (sure == 1)   /*  sure again  */
            {
                (seat_type[user_seat_type - 1].Booked)--;   /*  record booked seat +1  */
                write_seat(seat_type);   /*  change seat data  */
                write_users(id, 0, 0, 0);   /*  user data change  */
                system("cls");
                printf("you have cancel your seat\nrefund %d$\n\n", record_payment);   /*  Refund
recorded payment  */
            }
            else if (sure == 0)   /*  not sure  */
            {
                system("cls");
            }
        }
    }
}


void booking_seat(int id)   /*  user book seat  */
{
    struct users* users_list = NULL;
    int seat_type_condition = 0;

    read_users(&users_list);
    while (users_list != NULL)   /*  find specific user  */
    {
        if (users_list->users_id == id) {
            seat_type_condition = users_list->seat_type;
        }
        users_list = users_list->next;
    }
    if (seat_type_condition != 0)   /*  user have booked a seat  */
    {
        system("cls");
        printf("you have booked a seat\n\n");
    }
    else   /*  haven't booked  */
    {
        read_seat(seat_type);
        show_seat(seat_type);

        printf("enter 1 for regular\nenter 2 for electric_outlets\nenter 3 for computer\nenter 0 for
cancel\n");
        int choose = input_func(0, 3);
        if (choose == 0)   /*  back  */
        {
            system("cls");
            printf("back to previous page\n\n");
        }
        else
        {
            if ((seat_type[choose - 1].seat_num - seat_type[choose - 1].Booked) > 0)   /*  seat have
remain number  */
            {
                printf("\nplease enter the days you want to book (1-1024)\n");
                int days = input_func(1, 1024);
                int total_price = days * seat_type[choose - 1].price;
                write_users(id, days, choose, total_price);   /*  change user data  */
                seat_type[choose - 1].Booked++;   /*  booked seat +1  */
                write_seat(seat_type);   /*  change seat data  */
                while (getchar() != '\n');
                printf("\nplease pay %d$ for booking\nenter to pay:", total_price);   /*  calculate
the price  */
                while (getchar() != '\n');
                system("cls");
                printf("book success!\n\n");
            }
            else  /*  no seat remain  */
            {
                system("cls");
                printf("seat no remain\n\n");
            }
        }
    }
}
```

```c
/*------------------before user login--------------------*/


void users_menu(int id)   /*  show user menu  */
{
    while (1)   /*  show menu repetitive  */
    {
        printf("-------------\n");
        printf("| User menu |\n");
        printf("-------------\n\n");
        printf("input 1 for book a seat\n");
        printf("input 2 for view your information\n");
        printf("input 3 for change seat type / change booked days\n");
        printf("input 4 for cancel seat\n");
        printf("input 0 for previous page\n");
        int choose = input_func(0, 4);
        if (choose == 0)   /*  stop repeat  */
        {
            system("cls");
            printf("back to previous page\n\n");
            break;
        }
        else if (choose == 1)   /*  called func  */
        {
            system("cls");
            booking_seat(id);
        }
        else if (choose == 2)   /*  use same function with different choose  */
        {
            system("cls");
            change_seat(id, choose);
        }
        else if (choose == 3)   /*  use same function with different choose  */
        {
            system("cls");
            change_seat(id, choose);
        }
        else if (choose == 4)   /*  use same function with different choose  */
        {
            system("cls");
            change_seat(id, choose);
        }
    }
}
```

```c
void users_login()   /*  user login  */
{
    printf("\nplease enter your id\n");
    int id = input_func(0, 2147483647);
    struct users* head = NULL;
    read_users(&head);   /*  read all users data  */
    int condition = 1;
    int written_password = 0;
    while (head != NULL)   /*  find user  */
    {
        if (head->users_id == id)
        {
            condition = 0;
            written_password = head->users_password;
            break;
        }
        head = head->next;
    }
    if (condition == 1)   /*  no id found  */
    {
        system("cls");
        printf("ID do not exist,please go to registration\n");
    }
    else if (condition == 0)   /*  found the id  */
    {
        printf("\nHello! users %d\nplease enter your password\n", id);
        int password = input_func(0, 2147483647);
        if (password == written_password)   /*  check password  */
        {
            system("cls");
            users_menu(id);
        }
        else
        {
            system("cls");
            printf("Password wrong\n");
        }
    }

}

void users_registration()   /*  user registration  */
{
```

```c
printf("\nplease enter your id (no more than 9 numbers): ");
int id = input_func(0, 999999999);
int condition = 0;
struct users* head = NULL;
read_users(&head);   /*  read all users data  */
while (head != NULL)   /*  find user  */
{
    if (head->users_id == id)
    {
        condition = 1;
        break;
    }
    head = head->next;
}
if (condition == 1)   /*  found same id  */
{
    system("cls");
    printf("This id has been registrated\n\n");
}
else   /*  valid id for registration  */
{
    printf("\nHello! user: %d\nplease set your password (no more than 9 numbers): ", id);
    int password = input_func(0, 999999999);
    struct users* new_users = (struct users*)malloc(sizeof(struct users));
    if (new_users == NULL)
    {
        printf("Memory allocation failed\n");
        exit(1);
    }
    FILE* file_users;
    int err = fopen_s(&file_users, "users.txt", "a");   /*  add user to the end of file  */
    if (err != 0)
    {
        printf("Adding data file\n");
        exit(1);
    }
    fprintf(file_users, "%d %d %d %d %d\n",
        new_users->users_id = id,
        new_users->users_password = password,
        new_users->payment = 0,
        new_users->seat_type = 0,
        new_users->days = 0);   /*  print data to txt  */
    fclose(file_users);
    system("cls");
```

```c
        printf("registration sueecss!\n\n");
    }
}

void users()   /*  login or registration  */
{
    while (1)   /*  end until choose = 0  */
    {
        printf("----------------\n");
        printf("|  User login  |\n");
        printf("----------------\n");
        printf("\ninput 1 for login\n");
        printf("input 2 for create your account\n");
        printf("input 0 for pervious page\n");
        int choose = input_func(0, 2);
        if (choose == 0)
        {
            system("cls");
            printf("back to previous page\n\n");
            break;
        }
        else if (choose == 1)   /*  called func  */
        {
            users_login();
        }
        else if (choose == 2) /*  called func  */
        {
            users_registration();
        }
    }
}


/*------------------admin operation--------------------*/


void change_seat_price()
{
    read_seat(seat_type);
    show_seat(seat_type);
    printf("\n\nWhich seat price do you want to change?\n");
    printf("- 1 for regular\n- 2 for electrical outlets\n- 3 for computer\n- 0 for back\n");
    int choose = input_func(0, 3);
    if (choose == 0)
    {
        system("cls");
        printf("back to previous page\n\n");
    }
    else
    {
        printf("\nInput the price you want (0 - 65535)\n");
        int price = input_func(0, 65535);   /*  limit the range of price  */
        seat_type[choose - 1].price = price;   /*  change the chosen seat price  */
        write_seat(seat_type);   /*  write in txt  */
        system("cls");
        printf("Price changing success\n");
    }
}

void change_seat_num()
{
    read_seat(seat_type);
    show_seat(seat_type);
    printf("\n\nChoose the seat you want to change\n");
    printf("- 1 for regular\n- 2 for electrical outlets\n- 3 for computer\n- 0 for back\n");
    int choose = input_func(0, 3);
    if (choose == 0)
    {
        system("cls");
        printf("back to previous page\n\n");
    }
    else
    {
        printf("\ninput the number you want to add(less than 65535) / delete (e.g: 10 / -10)\n");
        int num = input_func(-65535, 65535);
        if (num > 0)   /*  for adding seat  */
        {
            seat_type[choose - 1].seat_num += num;   /*  add number to chosen seat  */
            write_seat(seat_type);
            system("cls");
            printf("Adding success\n");
        }
        else if (num < 0)   /*  for delete seat  */
        {
            int seat_remain_num = seat_type[choose - 1].seat_num - seat_type[choose - 1].Booked;   /*
calculate remain seat number  */
            if (seat_remain_num != 0 && -num <= seat_remain_num)   /*  delete less than remain  */
            {
                seat_type[choose - 1].seat_num -= (-num);   /*  delete chosen seat number  */
                write_seat(seat_type);
                system("cls");
                printf("Deleting success\n");
            }
            else   /*  delete number bigger than remain  */
            {
                while (getchar() != '\n');
                printf("you cannot delete this amount of seat\npress enter for back:");
                while (getchar() != '\n');
                system("cls");
            }
        }
        else   /*  input zero  */
        {
            while (getchar() != '\n');
            printf("\nSeems like you did nothing...\npress enter for back:\n");
            while (getchar() != '\n');
            system("cls");
        }
    }
}

void init_system()
{
    /*  some data for init system  */
    printf("please set the admin password(no more than 9 digits)\n");
    int password = input_func(0, 999999999);
    printf("\nplease set the REGULAR_SEATS number(0-65535)\n");
    int num_r = input_func(0, 65535);
    printf("please set the REGULAR_SEATS price(0-65535)\n");
    int price_r = input_func(0, 65535);

    printf("\nplease set the ELECTRICAL_OUTLETS_SEATS number(0-65535)\n");
    int num_e = input_func(0, 65535);
    printf("please set the ELECTRICAL_OUTLETS_SEATS price(0-65535)\n");
    int price_e = input_func(0, 65535);

    printf("\nplease set the COMPUTER_SEATS number(0-65535)\n");
    int num_c = input_func(0, 65535);
    printf("please set the COMPUTER_SEATS price(0-65535)n\n");
    int price_c = input_func(0, 65535);

    seat_type[0].seat_num = num_r;  seat_type[1].seat_num = num_e;           seat_type[2].seat_num =
num_c;
    seat_type[0].price = price_r;   seat_type[1].price = price_e;           seat_type[2].price =
price_c;
    seat_type[0].type = REGULAR;    seat_type[1].type = ELECTRICAL_OUTLETS;  seat_type[2].type =
COMPUTER;
    seat_type[0].Booked = 0;        seat_type[1].Booked = 0;                seat_type[2].Booked = 0;

    write_seat(seat_type);   /*  write seat data  */
    FILE* file_users;
    FILE* file_admin;
    int err1 = fopen_s(&file_admin, "admin.txt", "w");   /*  delete and create new file  */
    if (err1 != 0)
    {
        printf("Can't create <admin.txt> file\n");
        exit(1);
    }
    int err2 = fopen_s(&file_users, "users.txt", "w");   /*  delete and create new file  */
    if (err2 != 0)
    {
        printf("Can't create <users.txt> file\n");
        exit(1);
    }
    fprintf(file_admin, "%d", password);  /*  write admin password in txt  */
    fclose(file_users);
    fclose(file_admin);
    system("cls");
    printf("------------------------------------\n");
    printf("|    system initialize success!    |\n");
    printf("------------------------------------\n");
    printf("press enter back to menu: ");
    while (getchar() != '\n');
    system("cls");
}


void check_info()
{
    read_seat(seat_type);
    show_seat(seat_type);
    while (getchar() != '\n');
    struct users* head = NULL;   /*  pointer for users  */
    char* name = NULL;   /*  pointer for name  */
    int total_income = 0;
    read_users(&head);   /*  read user  */
```

```c
    printf("Exist users:\n");
    while (head != NULL)   /*  print users data  */
    {
        if (head->seat_type == 0) name = "No Booked";
        if (head->seat_type == 1) name = "Regular";
        if (head->seat_type == 2) name = "Elec_outlets";
        if (head->seat_type == 3) name = "Computer";
        printf("User_id:%11d, seat type:%15s, booked days:%5d, payment record:%10d$\n",
head->users_id, name, head->days, head->payment);
        total_income += (head->payment);
        head = head->next;
    }
    printf("\nTotal income is: %d$\n",total_income);   /*  calculate total income  */
}

/*------------------admin menu--------------------*/

void admin()
{
    while (1)
    {
        printf("----------------\n");
        printf("|  admin menu  |\n");
        printf("----------------\n\n");
        printf("Enter 1 for check information and total income\n");
        printf("Enter 2 for add / delete total seat numbers\n");
        printf("Enter 3 for adjust the price of seats\n");
        printf("Enter 4 for initialize all the data\n");
        printf("Enter 0 for previous page\n");
        int choose = input_func(0, 4);
        if (choose == 0)
        {
            system("cls");
            printf("back to previous page\n\n");
            break;
        }
        else if (choose == 1)   /*  check info  */
        {
            system("cls");
            check_info();
            printf("\npress enter for back:");
            while (getchar() != '\n');
            system("cls");
        }
        else if (choose == 2)   /*  change seat number  */
        {
            system("cls");
            change_seat_num();
        }
        else if (choose == 3)   /*  change seat price  */
        {
            system("cls");
            change_seat_price();
        }
        else if (choose == 4)   /*  initialize the system  */
        {
            system("cls");
            printf("sure to initialize the system? (all txt data will lose)\n");
            printf("input 1 for sure\ninput 0 for cancel\n");
            int sure = input_func(0, 1);
            if (sure == 1)
            {
                system("cls");
                init_system();
            }
            else if (sure == 0)
            {
                system("cls");
                printf("back to previous page\n\n");
            }
        }
    }
}

void admin_login()
{
    printf("please enter admin password\n");

    FILE* file_admin;
    int err = fopen_s(&file_admin, "admin.txt", "r");
    if (err != 0)
    {
        printf("Can't open admin.txt\n");
        exit(0);
    }
    int written_password;
    int input_password = input_func(0, 999999999);
    fscanf_s(file_admin, "%d", &written_password);   /*  read password from txt  */
    fclose(file_admin);
    if (input_password == written_password)   /*  password correct  */
    {
        system("cls");
        admin();
    }
    else
    {
        system("cls");
        printf("wrong password\n");
    }
}
```