Flight Reservation System

CS3101

Documentation

by

Pranav Ganesh Chandratre (17MS072)
Samya Roychowdhury (17MS050)
Ankit Anand (17MS069)
Himanshu (17MS042)

(Group No. 12)

Course Instructor
Dr. Kripabandhu Ghosh
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata

November 15, 2021

Contents

The whole code was compiled and executed using Dev C++ 5.11

1 Login System

1.1 Registration

There are two types of registrations possible in the system. User and Admin registration. Details of both the user and the admin are saved in the structure user.

1.1.1 Structure User

This structure is used for storing data User and admin information. It includes name, userid, email and password, all as strings. The User data is stored in a file User.dat using functions fopen()(in append mode) and fwrite() for simple file-handling. Similarly Admin data is stored in a file Admins.dat.
One major difference in user and admin registration is that User registration can be done by anyone with the option present on the home screen. As opposed to this, a new admin can only be registered by an existing admin. Some functions were defined to take input from the user/admin for registration.

1.1.2 p_input()

This function takes a char variable as an argument. We use function $fgets()$ to store information on terminal into the argument.

1.1.3 p_pass()

This function takes a char variable and also masks the password as it is typed on the terminal. It displays '*' instead of character typed in the terminal. The function ends when enter is pressed and stores the password in the string argument taken.

1.1.4 checkpass()

This function checks whether the password contains maximum 16 character contains a special character(!isalnum()), a number(isdigit()) and an alphabet(isalpha()). It also displays an error message and goes back to entering password again.

1.1.5 check_email()

This function checks if the taken string contains @ or not. We use ASCII value for @, 64 , for this.

1.2 Main

The home screen shows an option to go into User registration, User Login and Admin login. We use switch statement go through to various options. Specifically when we select user registration, we go to a terminal screen where you enter details of the user. Details are stored in the form of stucture user. We check if the userid is unique using basic file-handling, check is email is valid(check-email()), check if password is valid(checkpass()) and also confirm if password entered is correct by saving another string(cpass) where password has to be re-entered. We use function strcmp() to check if password and re-entered password match. Every condition has to individually satisfied to successfully register. Every condition has an appropriate error message associated with it which is displayed when condition is violated. If all conditions are satisfied we save the data in User.dat using fopen() in append mode and fwrite() functions.

1.3 Login

When you select Login option on the Home screen, it takes you to a new blank screen where it asks the user to enter it userid and password(p_input is used to take input). We open Users.dat in read mode using fopen() and use strcmp() to check whether the userid is registered and the password matches for the given userid. If either userid is not registered or password does not match it's corresponding userid, an error message associated with userid mismatch or password mismatch is displayed respectively. If you log in successfully, we proceed to Flight Booking.

1.4 Admin

When you select the Admin option on the Home screen, it takes you to a new blank screen where it asks the user to enter it adminid and password(p_input is used to take input). It proceeds similar to Login part just the file used for reference is Admins.dat instead of User.dat. Once you have succesfully logged in, it gives you an option to add a new admin or modify flight schedule.

1.4.1 Add new admin

This works similar to user registration,just the file used for storing admin information is Admins.dat.

1.4.2 Modify Flight Schedule

Here we presented with three options,

1. Add new flight.
2. Delete a flight.
3. View all flights.

We have stored the flight information in a structure called flights.

1.4.3 Structure flights

This structure is used to store information on various flights available for booking in the system. It includes flight number(also called ID), origin, destination as strings and day the flight is operational as an integer(Note: Sunday is 0, Monday is 1, and so on till Saturday is 6.) and distance travelled by the flight as float. In our system we have divided the flights based on weekdays i.e if a flight is operational on Monday this week it will operational on Monday of any future week. All this information is stored in a file called flights.txt.

1.4.4 addflight()

This function takes the name of where the flight information has to be stored as argument. For our case, it is flights.txt.This function is similar to user or admin registration. We take user input for flightid, origin, destination using p_input. For day and distance input we use scanf(). We store all input in the form structure flights. Then we use fopen() in append mode and fwrite to save the flight information in flights.txt.

1.4.5 deleteflight()

This function takes the name of where the flight information has been stored and the flight ID of the flight that is to be deleted as arguments. Basic idea of the function is that we create a new temporary file where we copy all the flights from flight.txt except the one we want to delete. Then we remove flights.txt and rename the temporary file as flights.txt.If flight with given flight ID is found and deleted a message is printed on the terminal acknowleding the same.If given flight ID is not present in flights.txt we display an appropriate error message.

1.4.6 displayflights()

This function takes the name of where the flight information has to be stored as argument.For our case, it is flights.txt. This function then opens the file in read mode and the prints the flight details of flights on the terminal using a loop.

1.4.7 Add a new flight

When you select this option it takes you to a blank screen and then addflight function is executed on flights.txt.

1.4.8 Delete a flight

When you select this option first the displayflights() is executed and all flights are displayed on the screen. The user is asked to Enter the flight ID of the flight they want to delete(p_input is used to store this in a variable). Then function deleteflight() is executed using flight ID and flights.txt as arguments. Finally displayflights() is executed so user can confirm if the flight was deleted.

1.4.9 View all flights

Function displayflights() is run with flights.txt as argument and all flights with their corresponding information is displayed on the terminal screen.

2 Flight Booking by User

In this section we will focus on describing the functions and structure used to obtain data from passenger, store that data and finally display the ticket. The age category is automatically decided once the user enters his/her age.

2.1 Structure pass_data

This structure is used for storing the data supplied by passenger. It stores first name of the passenger, last name of the passenger, age of the passenger and ticket fare (the detailed calculation used for ticket fare will be discussed in the next section).

2.2 s_information()

This function takes character as input and returns integer. This is used to update the seat information. It basically checks a file named "FlightIDdatemonth.txt" and opens it in read mode. If the file is not present, it opens the file in append mode and write the available seat as 30 that is the maximum capacity. If the file is present, it seeks the last two digits of the file(by "fseek" command) which is the updated available seats in that particle flight at that particular day in the variable s_n.

2.3 information_ticket()

This function takes argument char*f, int s_n and int s. It returns void. The purpose of this function is to open the file named "FlightIDdatemonth.txt" in append mode and print the available seats after booking in the same file (using "fprintf"). Then the file is closed. Here the variable s_n denotes the number of seats user wish to book at once (which cannot be greater than 4). The variable s is the maximum seats available in that flight at that day.

2.4 ticket_disp()

The argument of this function is **int** m, **float** tot_fare, **char** fn[50], **int** d, **int** mon, **char** o_final[50], **char** d_final[50], **char** n_final[50],**char** g[4]. Inside this function structure flights is stored in a local variable f. Then the file "flights.txt" is opened using "fopen" in read mode. Then it compares the details of the flight name entered by the user and print corresponding flight details along with passenger details (e.g. first name, last name, age, gender, age category, individual fare) and total fare in the terminal.

2.5 info_collect

The argument of this function is **char** *f, **int** m, **int** s, **char** fn[50], **int** d, **int** mon,**char** o_final[50], **char** d_final[50], **char** n_final[50], **char** g[4], **float** dist. It basically collect the information from the passenger from the terminal and stores them in relevant variables. This step is done using "printf" and "scanf" function.It also calculates fare for individual passenger by calling the function p_fare_c (discussed in Payment part of the document). Finally it

calculates total fare for the corresponding ticket and stores them in a variable. At the end of this function, it calls function ticket_disp().

2.6 pnr()

This is used to generate unique PNR of 7 digits. Here the function "srand()" is used to take a seed value , in this case the tome of booking in seconds, for generation of random number (which is done by function"rand()").

2.7 weekday()

It takes date , year and month as argument and returns numerical equivalent of weekday. For example, 0 for Sunday, 1 for Monday and so on. The reference can be found here.

2.8 Main function

Now let's discuss the booking part of **int** main(). The booking portal at first asks the user to put five details- origin of the flight, destination of the flight, date of travel, month of travel and year of travel. Then using the function weekday the date, month and year as given by the user is converted into number between 0 t0 6 which denotes the weekday (e.g. 0 for Sunday, 1 for Monday etc). Inside main structure flights is called as flight. Then a file named c_file is created where "flights.txt" is opened in read mode. Then the program search for the relevant details in the file "flights.txt" and print the flight details accordingly in terminal. To do this we have used 'while' loop , "fread" function and if statement. If statement checks three things- number equivalent of weekday, destination and origin of the flights. If all these three matches with any flight, flight id , destination and origin of the concerned flight is printed. If no such flights are found, the program goes back to the starting of booking portal i.e. asks to re-enter the five details as discussed at the beginning of this paragraph.
Once the flight is found then the program asks the user to write the flight id , re-enter the date and month of travel. Interesting thing to note is that we have applied checks at every step of these information. Say, for example, if the user mistakenly writes the wrong Fight id , the program will show a message "Incorrect flight ID. Press any key to re-enter flight Id". If user presses any key , the program will go back to ask the user about flight id and other details as discussed earlier. Similar checks are also associated with date and month entered by user.
If every details written by the user are correct, the program will move to print the available seats in that flight on that particular day. Basically, what the program does is to concatenates the strings flight id, date , month and ".txt" in the variable "flight_no". Then it uses the function "s_information" with "flight_no" as argument to update the available seat information and finally prints that information. Then it asks for the number of passengers the user wish to book a ticket for. Now there are several checks. If the number of passenger is more than 4, the system will show proper error message and will ask the user to re-enter this particular details. In case the seats user wish to book is more than the available seats, the program will take the user back to the very first part of booking portal with the message shown as "Currently Unavailable".
If the user passes all these checks, the program will call the function "info_collect" with proper arguments(discussed earlier). Now, there is an important thing to note. The system will now ask the user whether he or she will like to do payment or not. Note that, if the user decides not to pay, the available seat information will remain same. Once the user enters the payment portal, program calls the function "booking" with proper arguments(discussed in Payment section). Once the payment is done the ticket will be displayed on terminal with the flight details, passenger details and PNR generated by the function "pnr()".

3 Payment

This section describes the method of payment for the booking. After we collected all the required information of the passengers and the total amount to be paid is shown on the screen.

3.1 p_fare_c()

This is **passenger fare calculator**. This function is used to **calcuate the passenger fare** which includes base fare along with booking charges and the distance fare. We have different base fares increasing according to the age

and special reduced base fare for the senior citizens. Distance fare is Rs. 8 per Km. We have set different distances travelled by the flights which is the actual air distance between the origin and the destination. Booking charges is 16% of the total of the distance fare and the base fare.

- The return type of this function is **float**.
- The arguments required are **int age**, **float d_fare**, **float dist**

The argument 'age' is used for different base fare, 'd_fare' is constant which is Rs. 8 per Km, and 'dist' is the distance between the origin and the destination.

3.2 total_fare()

This module is used to calculate the sum total fare of the all the passengers travelling together. It is just the loop run of the function "**p_fare_c**()" for all the passengers.

- The return type is **float**
- Arguments used is **int n**, **float dist**

Arguments 'n' is the number of passengers travelling together and 'dist' is the distance between the origin and the destination which is being read by the 'flights.txt' file.

3.3 booking()

The module used is **booking**(). This module is made for the **final payment** after the passenger done checking its flight and passenger details.

- The return type of this fucntion is **void**.
- Arguments taken in this function are: **float payable_amount**, **char o_final[50]**, **char d_final[50]**, **char n_final[50]**, **int d**, **int mon**, **int m**, **char *f**, **int s**.

It facilitates the user with two options for the payment:

- Card payment
- UPI

The module uses "switch" to take the required inputs for the respective fields to choose between Card payment and UPI payment. This function has three associated structs in our header file "st.h".

- Card_payment: The variables along with the argument types are:
  - unsigned long int card_no. : stores card number.
  - unsigned int cvv : stores cvv of the card
  - unsigned int exp_date : stores expiry date of the card
  - unsigned int exp_mon : stores expiry month of the card
- UPI_payment: The variables along with the argument types are:
  - char UPI_id[50] : stores UPI ID
  - int UPI_pin : stores UPI PIN
- pay_details: The variables along with the argument types are:
  - char pay_type
  - struct card_payment card_pay
  - struct UPI_payment UPI_pay

The arguments 'payable amount' is used for displaying the total fare for the travel along with the tickets. Similarly, 'o_final', 'd_final', 'd' and 'mon' are origin, destination, date of travel and month of travel respectively are used for displaying the flight details on the terminal. The other arguments 'f', 'm', 's' are used for the function "**information_ticket**()" for displaying the final ticket along with the PNR on the terminal after the successful completion of the payment.
A good feature is in the 'UPI' section, we have checked for **'@'** symbol in the UPI ID. It shows 'invalid ID' and again ask for the UPI ID. We have used ASCII value of **'@'** is 64 for checking the UPI ID.

4 Contributions

1. **Pranav Ganesh Chandratre(17MS072):** Admin system full, Login Portal
2. **Samya Roychowdhury(17MS050):** Flight Booking Portal full
3. **Ankit Anand(17MS069):** Payment Portal, User registration
4. **Himanshu(17MS042):** Fare calculation