Date: 22.04.2019

MINI PROJECT

Airline Reservation System

BRIEF DESCRIPTION OF THE PROJECT

The Airline Reservation System project is an implementation of a general Airline Ticketing website, which helps the customers to search the availability and prices of various airline tickets, along with the different packages available with the reservations. This project also covers various features like registration of the users, modifying the details of the system by the management staff or administrator of the system, by adding, deleting or modifying the customer details, flights or packages information. In general, this system is designed to perform all the operations that any other airline ticketing website available online does.

FILES, MODULES AND DATA STRUCTURES USED FOR IMPLEMENTING THE PROJECT

FILES:

user.bin

DATA STRUCTURES:

date: Structure to store the dates in the desired format.

```
struct date
{   int day;
    int month;
    int year;
};
```

location: Structure to store the address in the desired format.

```
struct location
{   char street[40];
    char city[25];
    char pincode[10];
    char state[25];
};
```

user_details: Structure to store the user account details.

```
struct user_details
{   char username[40];
    char password[40];
    char name[40];
    struct location address;
    char nationality[40];
    char mobile[12];
    char email[45];
    struct date dob;
    int age;
    char gender;
};
```

admin: Structure to store the admin login details.

```
typedef struct
{int code;
char pw[15];
} admin;
```

time: Structure to store the time in the desired format.

```
struct time
{int hh;
int mm;
};
```

flight_det: Structure to store the flight details.

```
struct flight_det
{int acode;
char fcode[10];
enum places source;
enum places destination;
struct time deptime;
```

struct time arrtime;

enum days day;

price adult_first;

price adult_business;

price adult_economy;

price child_first;

price child_business;

price child_economy;

};

MODULES:

1. **SYSTEM INTERFACE MODULES**

   1.1. **void intro():** Prints the introductory page with creator details.
   1.2. **void user_menu():** Prints the user menu options in a user-friendly format.
   1.3. **void user_login():** Takes in the username and password and checks validity. If the entered inut matches stored details, login is successful. Else, ERROR message is thrown.
   1.4. **user_terminal():** It displays a menu for the user to signup or login into their account.
   1.5. **void admin_menu():** Prints the admin menu options in a user-friendly format.
   1.6. **void add_admin():** It facilitates adding another login entry.
   1.7. **void admin_login():** Takes in the username and password and checks validity. If the entered input matches stored details, login is successful. Else, ERROR message is thrown.
   1.8. **void getPassword(char password[]):** It takes in the password with * display instead of the characters getting printed on the screen for security.
   1.9. **void check(char password[]):** Verification of the username and password is done here.
   1.10. **void user_det_input():** It takes in input for the various fields of the user_details structure variable and creates a single user account entry that is stored in a file.

2. **USER ACCOUNT MODULES**

   2.1. **book():** It alows the booking of the tickets for a given source and destination.
      2.1.1. **void DFS(enum places source, enum places dest, enum places day, int visited[], struct path trip,int *path_no, struct flight adj_list[][10], int nodes[], struct path paths[], struct date *dep_date, struct tm *curr):** The function under

book ticket that filters out all possible airlines based on the given date optimally.

2.2. **void user_det_update():** It takes in username as the input and searches for the required user account in the file. It then allows the desired changes and makes changes to the file.

2.3. **void user_det_delete():** It deletes the account of a particular user.

3. **ADMIN ACCOUNT MODULES**

3.1. **void add_flights(int n, int acode):** It facilitates entry of new available flying flights.

3.2. **void display(int i,struct flight_det allflight[]):** It displays details of all the available flying flights.

3.3. **void modify(int index,int n,struct flight_det allflight[]):** It facilitates updating flight details.

3.4. **void delete(int index,int n,struct flight_det allflight[]):** It deletes unavailable flights with their coressponding details.

3.5. **void user_det_view_all():** It displays the details of all the user accounts for administrative purposes.

3.6. **void user_det_view_particular():** It displays the details of a particular user account for administrative purposes.

CONCEPTS USED TO BUILD THE PROJECT

Modularity: The entire program is split into several functions and parts, each of which serves a specific purpose and can be repeatedly used.

Incremental Programming: Several versions were created prior to the final and were constantly modified to make it the best possible version there is.

Integration: Several parts of the code was designed by different people and was integrated into one program using the interface code as the base.

Enumerated data types: It was used to conveniently assign values to days and places.

Arrays: Used for storing collection of the same data like list of all the flights.

Structures: Used for storing collection of related data like user_details, flight_details.

Functions: For implementing specific operations.

Files: For storing data permanenty.