**Flight Management System**



Session: 2022 – 2026

**Submitted by:**

M.IJLAL 2022-CS-04

**Supervised by:**

MAIDA SHAHID

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

* **Description**

I want to make a flight management system in which user can books flights according to his/her needs. He can also see announcements, change food type, change his account info and more. With this system, user do not have to wait in lines and visit airports to see schedules, rather he can do it all from his home. It is a complete package for any airlines system, admins can handle database while users can easily buy tickets for himself.

* **Users of Application**

**Customers or clients:**

These are the users who are consumers and are using our system to find and book flights. They can book a flight, cancel flight, change food ( if available) , change timings of his flight, he can also issue a complain if necessary.

**Admins:**

He is the one who has the ultimate access, he can see which customers has booked a flight, total flights, complaints. He can also change contact info and make announcements.

* **Functional Requirements**

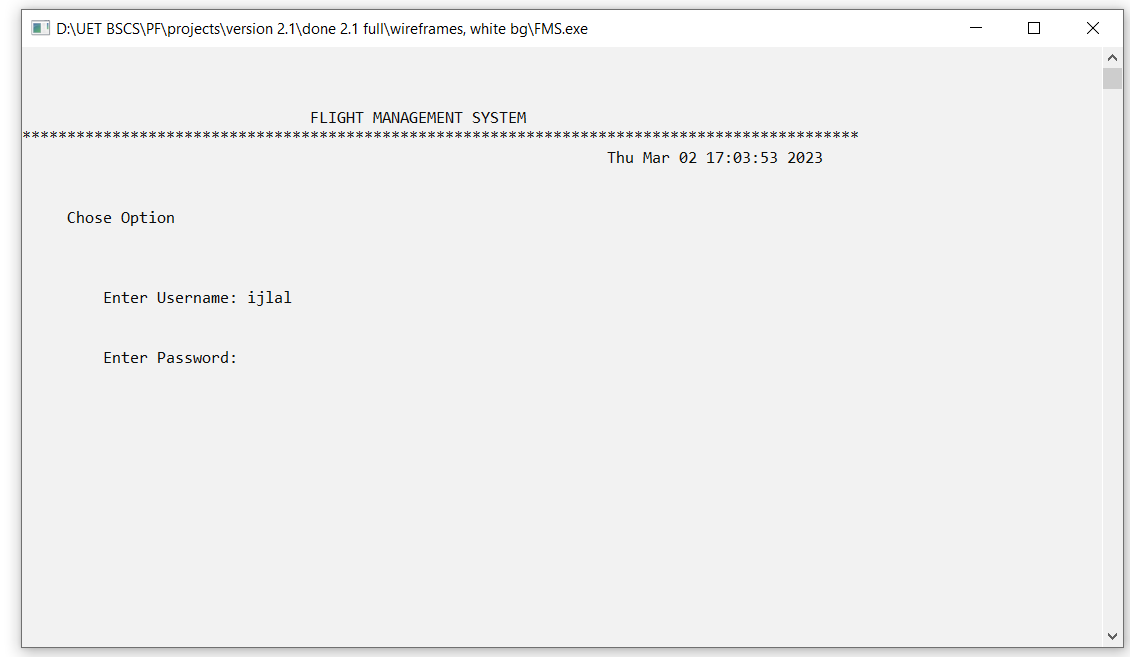
|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | *As a* | *I want to perform* | *So that I can* |
| **1** | User | Book flight | Add records in database so that seat can be reserved in real time |
| **1** | Cancel Flight | Remove record from database so that seat can be given to any other user |
| **2** | Check Ticket | User can see again his flight timings, date, pickup and destination point. |
| **3** | File Complaint | In Case user has any issues, he can contact help and contact section. |
| **4** | Change Timings | Move user registered data to another flight in database. |
| **5** | File Complaint | Add complain in complain box so that admin can see them. |
| **6** | Change seat | Change user seat from database to another (if possible) |
| **7** | See History | Show user which flights he has took in last period. |
| **8** | Help | Add contact info and help guide so that user never feels stuck. |
| **9** | User | See announcements | Show announcement added by admin to user and also notify when there are unread available. |
| **10** | Change Username | Change username form database (with instant logout after changing) |
| **11** | Change Password | Change password form database (with instant logout after changing) |
|  |
| **1** | Admin | Check Booked Flights | See all the flights available in database |
| **2** | Add flights | Add a flight in database which users can see |
| **3** | Remove Flights | Remove a flight from database |
| **4** | See Complaints | Show admin all the complains added by users |
| **5** | Change Appearance | Change color of text and background |
| **6** | Change Contact Info | Change contact number from database |
| **7** | Change Help Menu | Change contact number from database |
| **8** | Search Customers | Search customers from database by name, date number of flights took. |
| **9** | Remove Customers | Remove customers from database |
| **10** | Add Announcement | Add announcement in its section so that users can see them |
| **11** | Remove Announcement | Remove an announcement in case it gets outdated. |
| **12** | Modify Announcement | Modify announcement in case there was a mistake |
| **13** | Read Announcement | Admin can see announcements he has made. |

* + The password must be 8 characters long with a numeric value.
  + Passport number must be correct
  + Phone number must be correct
  + Username must not be taken
  + Seats are available in flights
  + Discounts are also available which are notified via announcements.
  + User cannot cancel flight after certain time.
  + And more.
* **Wireframes**

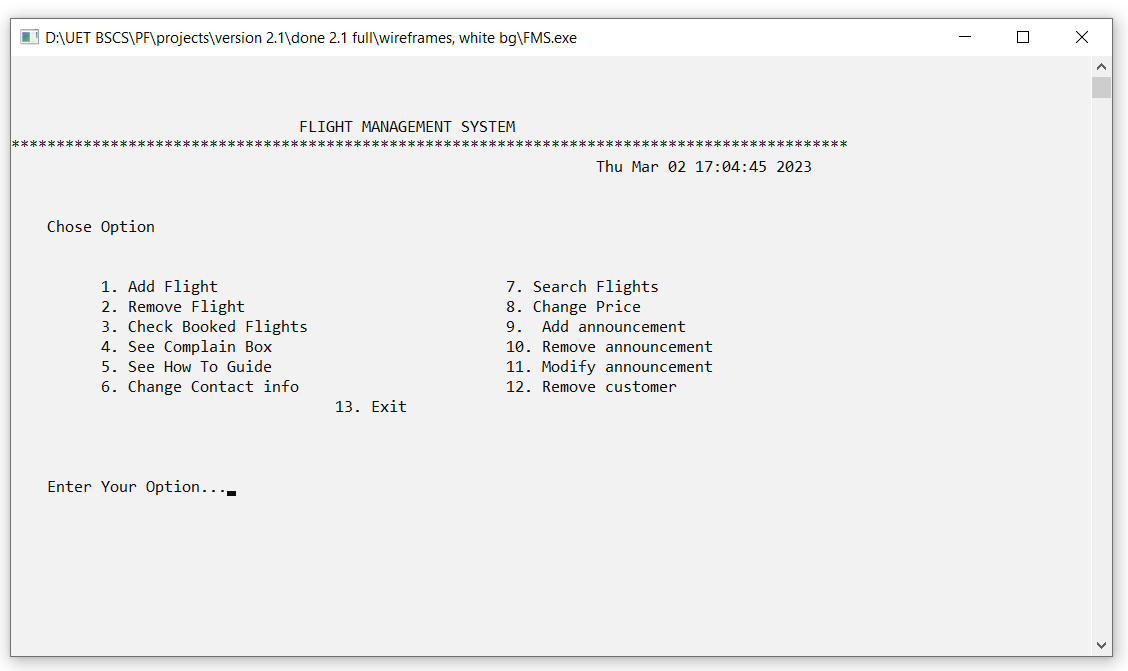
**Figure 1: Startup Screen**



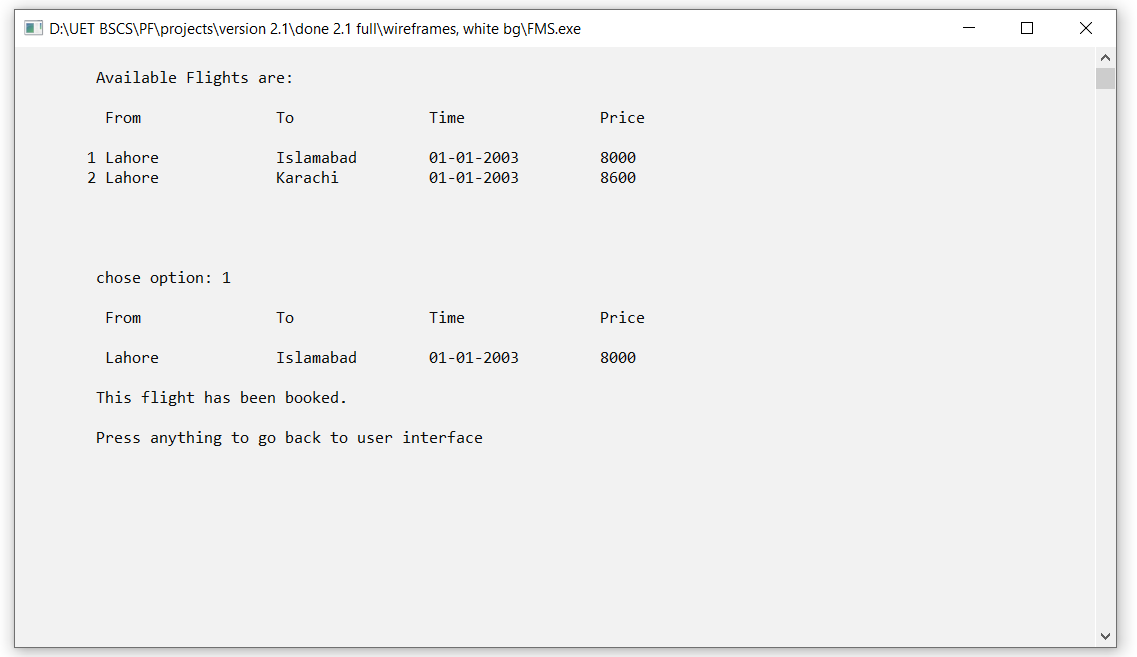
**Figure 2: Login Screen**



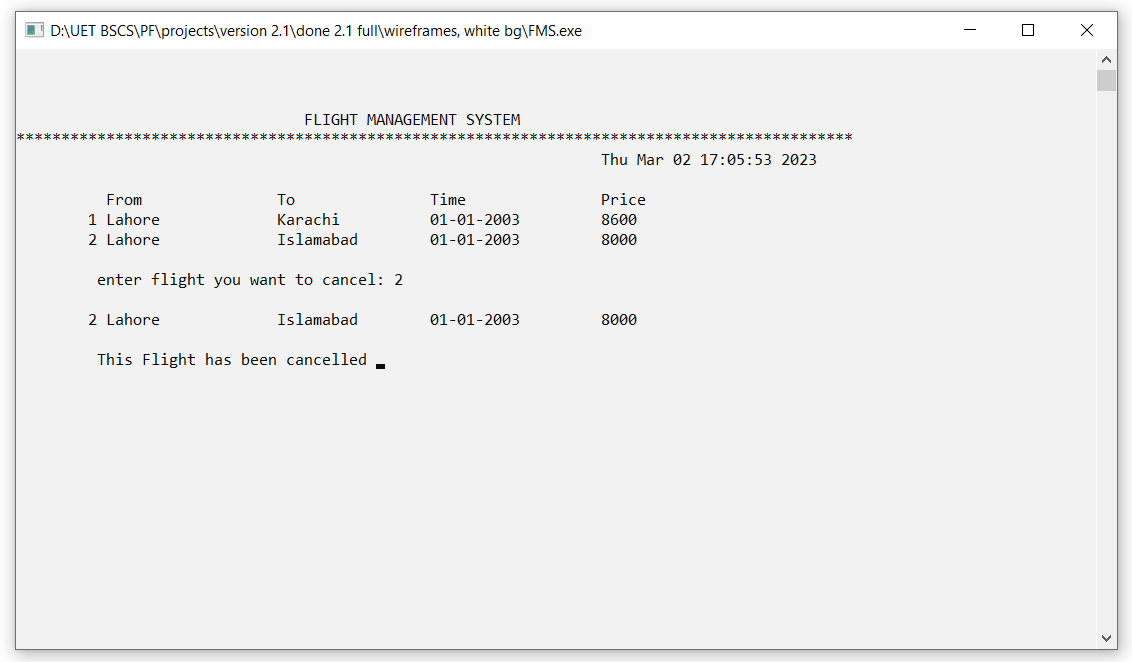
**Figure 3: Admin Interface**



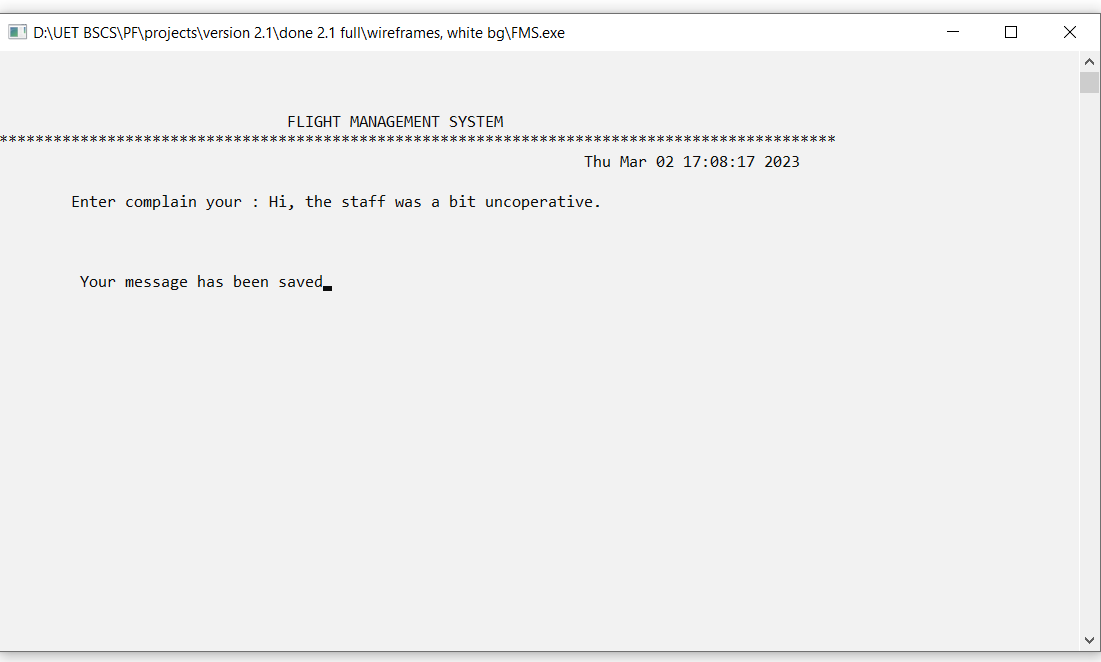
**Figure 4: Book Flight**



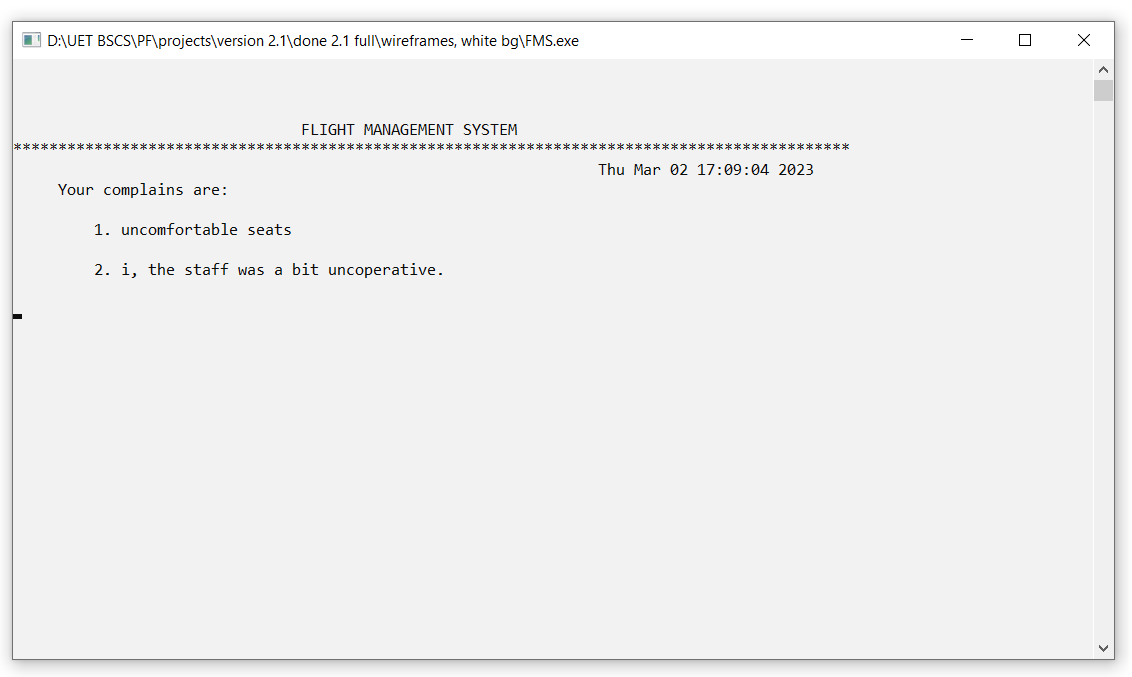
**Figure 5: Cancel Flight**



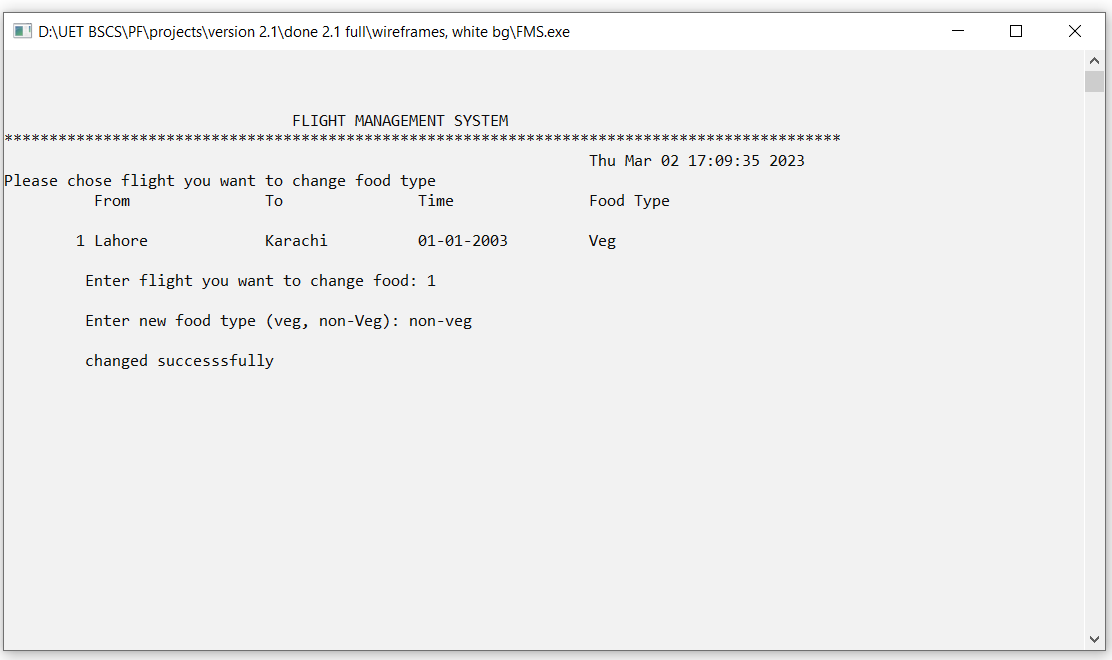
**Figure 6: File Complain**



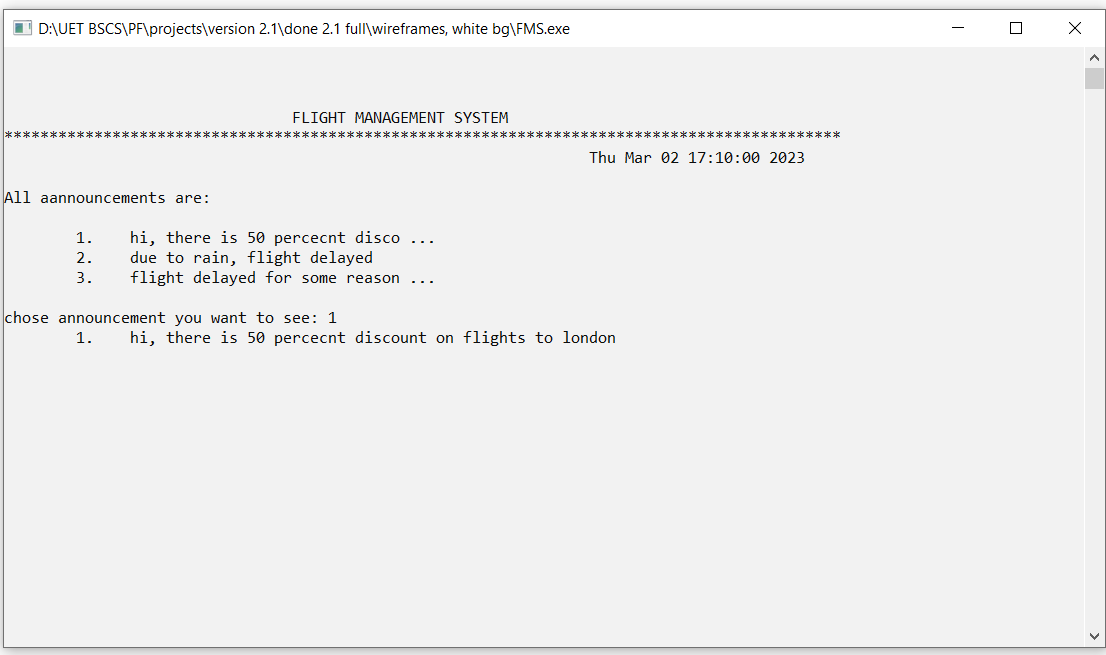
**Figure 7: See Complains**



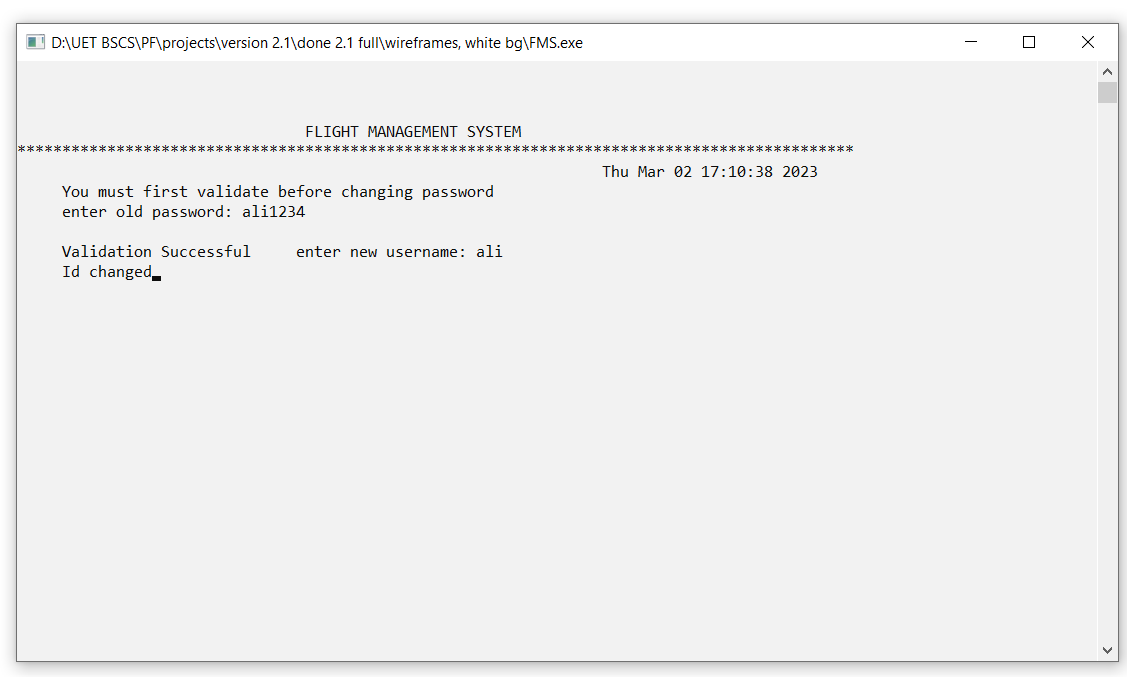
**Figure 8: Change Food Type**



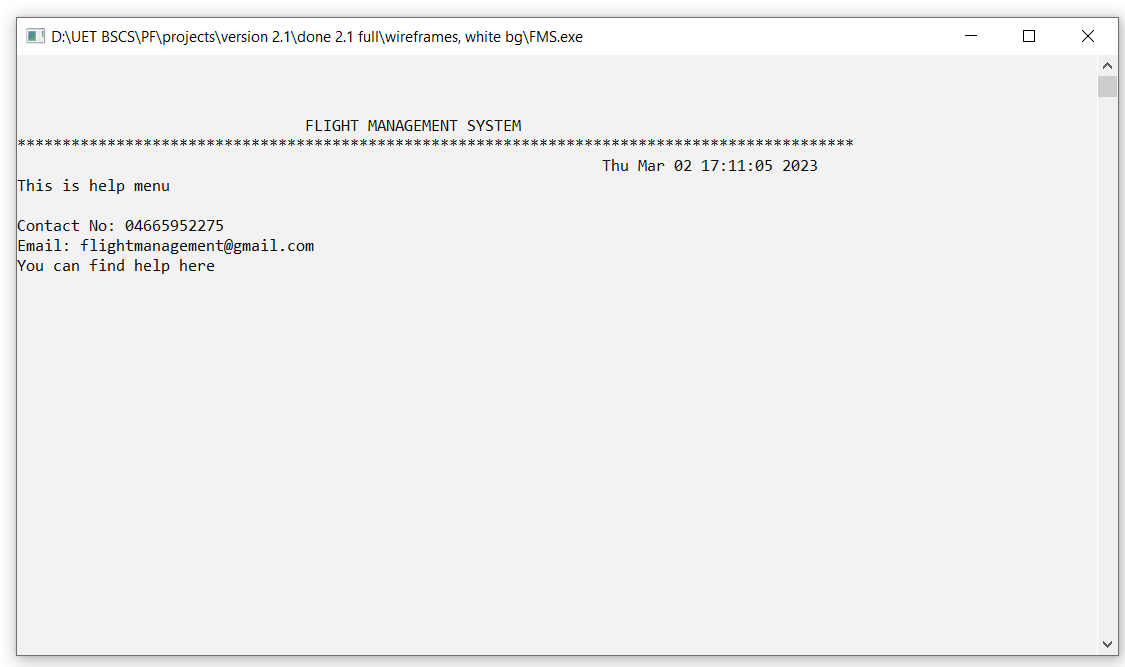
**Figure 9: See Announcements**



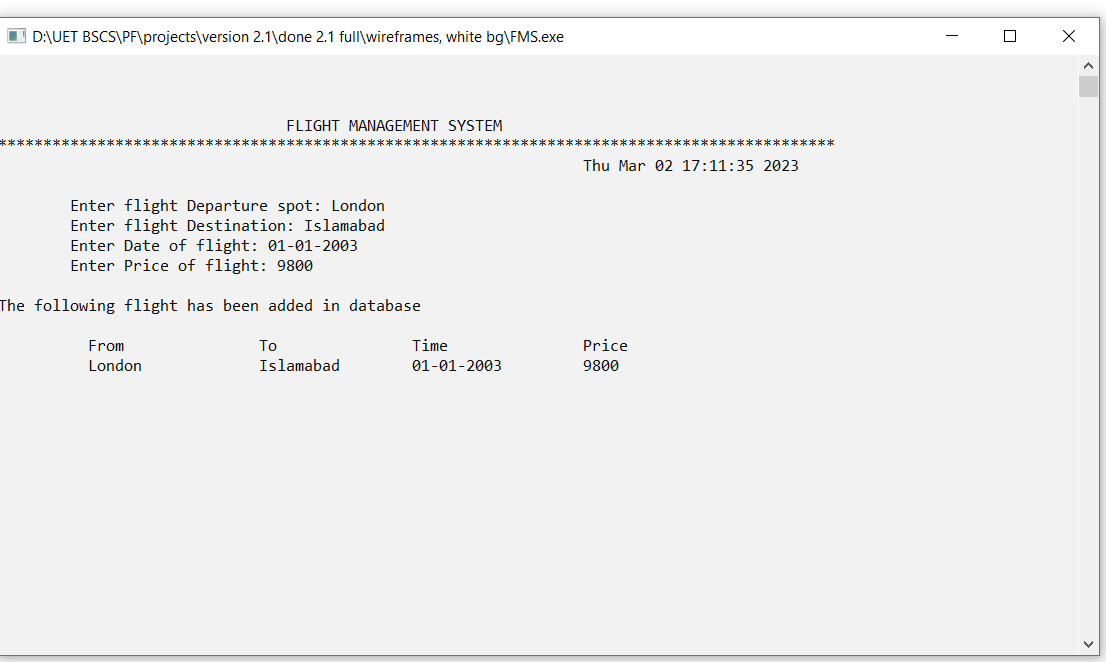
**Figure 10: Change ID**



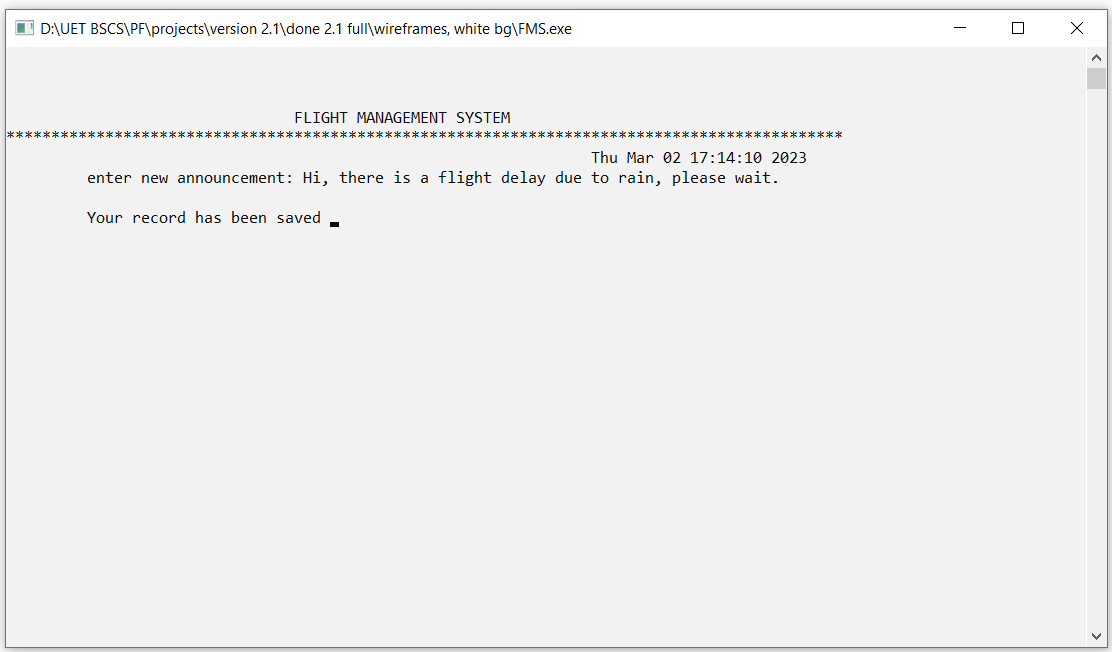
**Figure 11: See Help Menu**



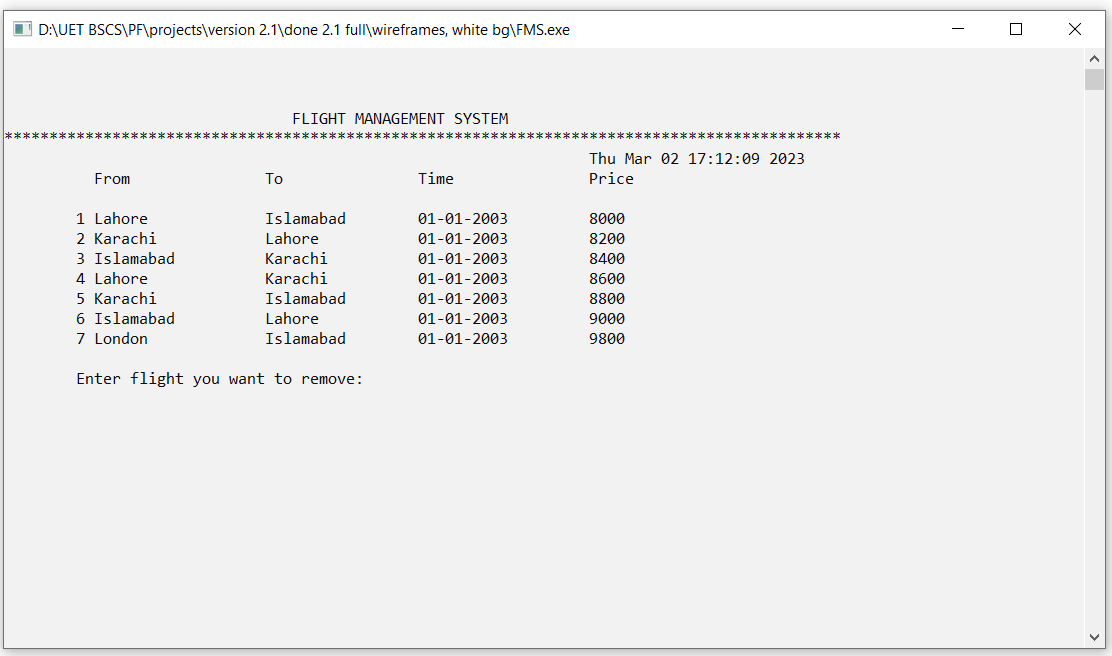
**Figure 12: Add Flight**



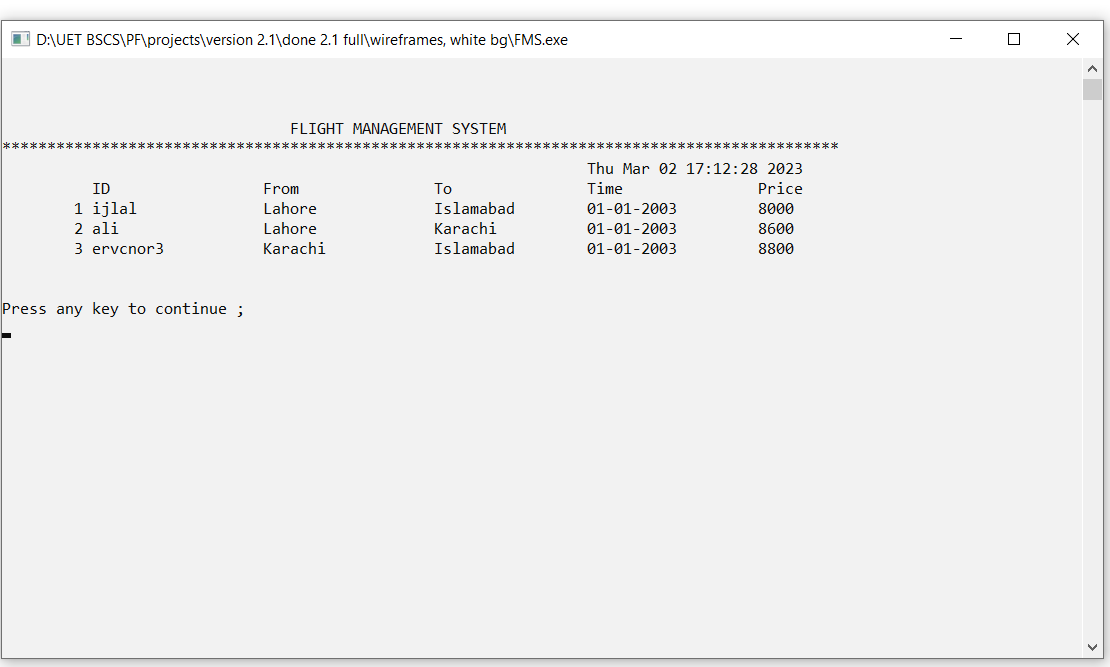
**Figure 13: Add Announcements**

****

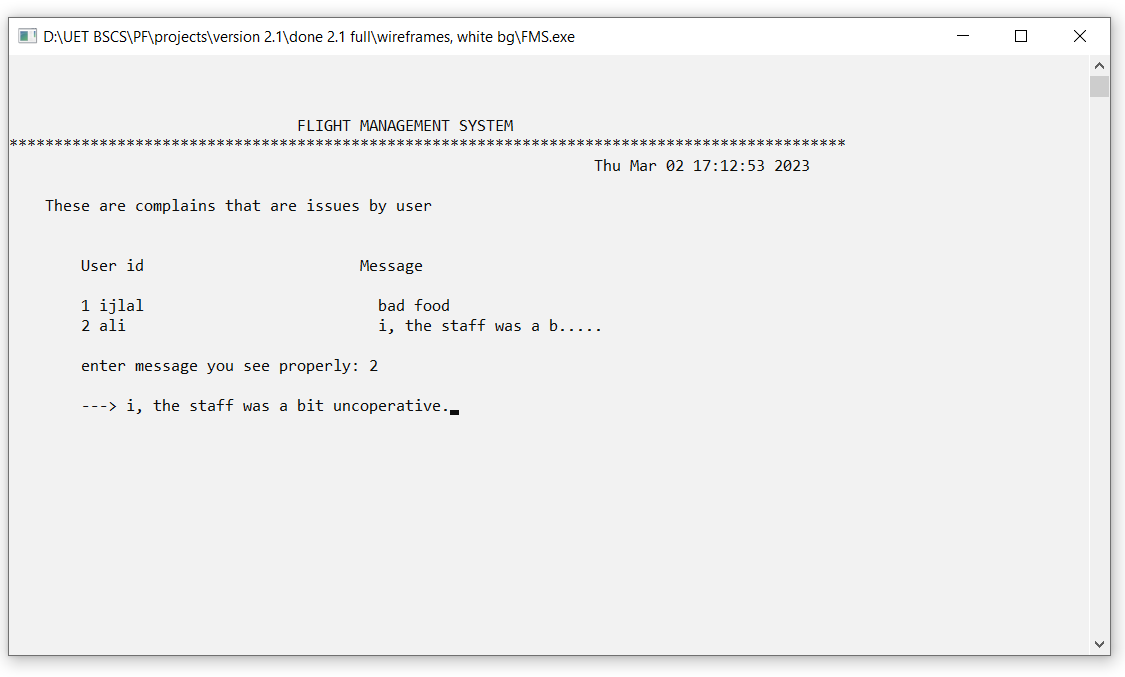
**Figure 14: Remove Flight**



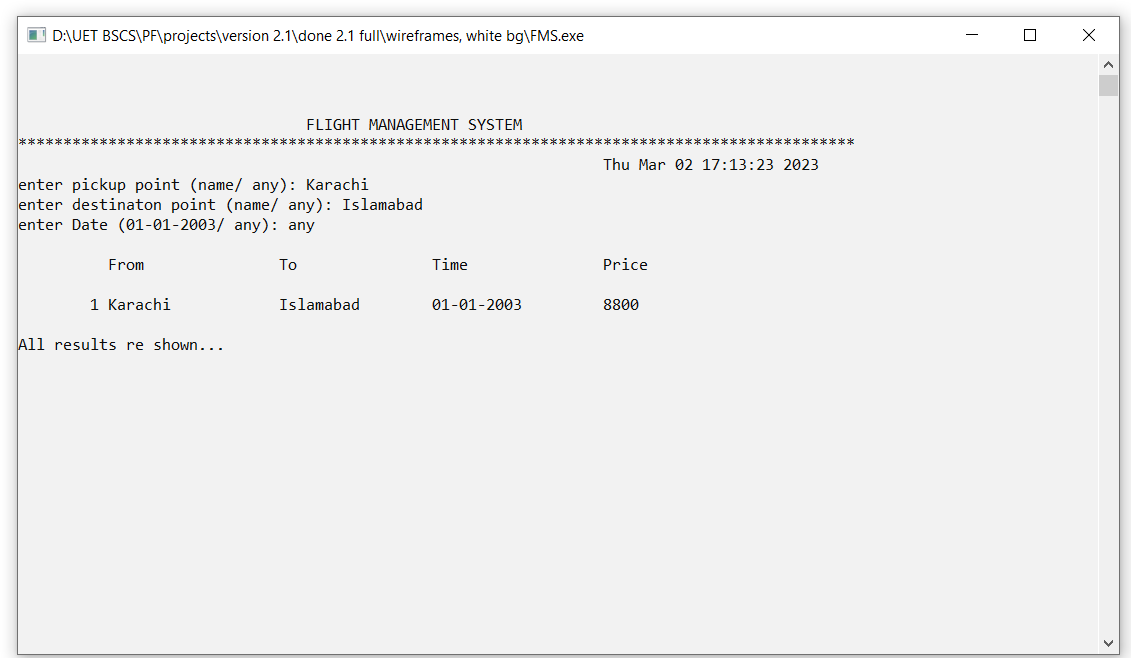
**Figure 15: See Booked Flights**



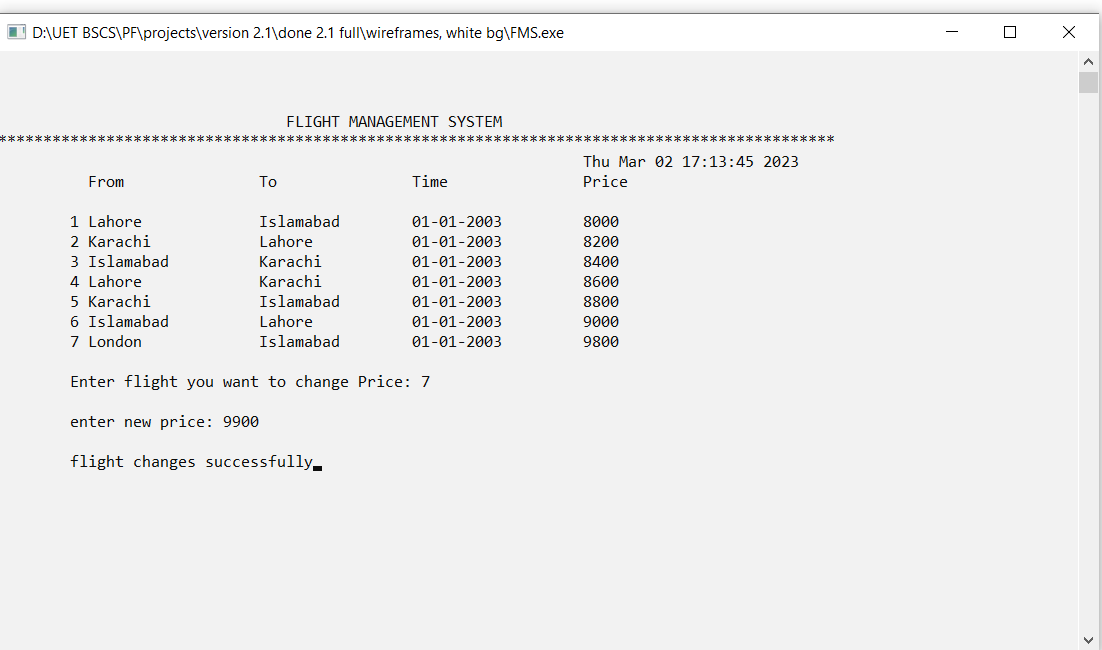
**Figure 16: See Complains**



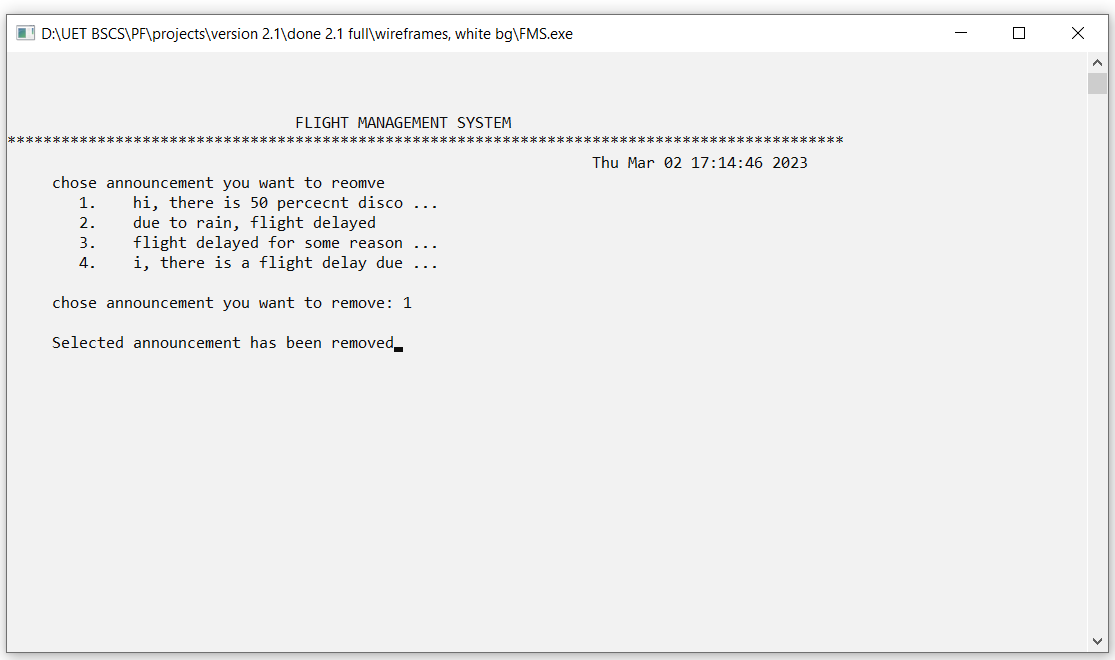
**Figure 17: Search Flight**



**Figure 18: Change Flight Price**



**Figure 19: Remove Announcements**



* **Complete Code of the Business Application**

#include <iostream>

#include<conio.h>

#include<windows.h>

#include<fstream>

#include<string>

#include <ctime>

#include<time.h>

using namespace std;

string currentrole = "none"; // this shows what role is currently now

string currentId = "none"; // this shows what role is currently now

string Username[20] ; // usernames are storeed in it

string Password[20] ; // password are storeed in it

string Role[20] ; // admin/user are storeed in it

int nr\_users = 0 - 1; // stored in such a way, if there are 1 users, nr\_users are 0, so indexing is easy

string flightFrom[20] ; // where flight will leave

string flightDestination[20]; // where flight will reach

string flightDate[20] ; // self explanatory

string flightPrice[20] ; // self explanatory

int nr\_flights = 0 - 1; // stored in such a way, if there are 1 flights, nr\_fligths are 0, so indexing is easy

string bookedUsersId[20] ;

int bookedUsersFlight[20] ;

string bookedUsersFood[20] ;

int nr\_booking = 0-1;

void changeFood(); // user can chose to change food of a flight

void everyEnd(); // what is to be printed at the end of every screen

void printHeader();

void printStartupScreen();

void printEnterOption();

void printLoginScreen();

void printSignupScreen();

void printUserInterface();

void printAdminInterface();

void addFileComplain();

int enterIdPassword(string, string, string); //check for validity of id and password and return 0 or 1

int checkIdPassword(string, string, string);

int newPasswordCheck(string); // check if a new password meets required criteria

void bookFlight(); // user option 1

int displayFlight(string, string, string, int); // shows all the flights available

void modifyFlight(); // not used anywhere as of now

int changePassword(); // user option

int changeId(); // user option

void exit(); // used to move back to startup screen

void cancelFlight(); // user option

void displaySpecificUserFlight(); // display booked flights of current user

int adminAddFlight(); // admin option, adds flight to a database

int adminSeeBookedFlight(); // see flights booked by user

int adminRemoveFlight(); // remove flights fromdatabase

void adminRemoveCustomer(); // delete a user

string announcmentMessage[20] ;

int nr\_announcements = 0-1;

void adminAddAccnouncement(); // admin options

void adminRemoveAccnouncement(); // admin options

void adminModifyAccnouncement(); // admin options

void adminSearchFlight(); // admin options

int subShowAllAnnouncement(string, int, string);

void seeAnnouncements();

string complainId[20];

string complainMessages[20];

int nr\_complains = 0-1;

void removeComplain();

void adminPrintComplain();

string CheckAlphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

string CheckNumber = "123456789";

string contactNo = "04665952275";

string email = "flightmanagement@gmail.com";

string helpMessage = "You can find help here";

string resetContactNo = "04665952275";

string resetEmail = "flightmanagement@gmail.com";

string resetHelpMessage = "You can find help here";

void printHelpMenu();

void adminEditHelpMenu();

void clearWithoutHeader();

void exitToUserInterface(string, int); // move back to user interface

void exitToAdminInterface(string, int); // move back to admin interface

void printSky();

void printPlane(int);

void clearSpace();

void oneTimeStartup();

int SeachWord(string, string);

// ================================== //

// anim prototypes //

void animprintRobo(); // print robo used in help menu

void animprintAdminInterface(); // print animationused in startup

void animprintHowToLogo(); // print animationused in startup

void animhowToUseFull(); // print animationused in startup

void animprintRoboInterface(); // print animationused in startup

void animStartupOneTimeAnim(); // print animationused in startup

bool isAnimShown = false; // so that inly 1 time is shows

// ================================== //

void adminchangeAppearance();` // removed in final version

// VERSION 2.0 CHANGES //

void storeIdPass(); // store all data form txt

void loadIdPass(); // load all data form txt

void storeAnnouncments(); // store all data form txt

void loadAnnouncments(); // load all data form txt

void storeAvailableFlights(); // store all data form txt

void loadAvailableFlights(); // load all data form txt

void storeBookedFlights(); // store all data form txt

void loadBookedFlights(); // load all data form txt

void storeComplains(); // store all data form txt

void loadComplains(); // load all data form txt

void loadAllFiles(); // load all data form txt

string getFieldFromRecord(string record, int option);

// VERSION 2.1 CHANGES //

void GeneralShowSingleFlight(string, string, string, string, int); // show flights booked by a user

int subReturnCountAnnouncement(); // return nr. of announcemtns available without none

int subReturnCountComplains(); // return nr. of complains available without none

bool isLocationAvailable(string, string); // if location to go input by user available ??

string enterToLocation();

string enterFromLocation();

string enterDateCorrectly(); // validations

string enterfoodCorrectly(); // validations

void gotostartupscreen(string);

void changePrice();

string printTime(); // time at top left

void printUniqueLines();

int main()

{

loadAllFiles();

oneTimeStartup();

// Start up screen

printStartupScreen();

printEnterOption();

getch();

return 0;

}

// ===================================================================

void gotoxy(int x, int y)

{

COORD coordinates;

coordinates.X = x;

coordinates.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);

}

void oneTimeStartup()

{

system("Color 03");

gotoxy(28, 12);

cout << "===== Enter full screen for best experience =====";

gotoxy(0,0);

getch();

system("Color 3F");

// system("color \*blue\*\*white\*");

// getch();

printSky();

gotoxy(90,30);

cout << "LOADING...";

for (int i =1; i <= 30; i++) // 17x5 = 85 \\ 300

{

printPlane(i);

Sleep(120);

clearSpace();

}

printPlane(30);

gotoxy(90,30);

cout << "LOADED SUCCESSFULLY...";

Sleep(1000);

// system("Color 03");

// getch();

}

// ==================================================================//

int SeachWord(string word, string sentence) // this will tell if a word is present in a string

{

int word\_index = 0;

int matching\_Word = 0;

int wordCheckSize = word.length();

for(int i = 0; i <sentence.length(); i++)

{

if (sentence[i] == ' ')

{

word\_index = 0;

matching\_Word = 0;

continue;

}

if (sentence[i] == word[word\_index])

{

matching\_Word+=1;

if (matching\_Word == wordCheckSize )

return 1;

}

word\_index+=1;

}

return 0;

}

int displayFlight(string from = "any", string destination = "any", string date = "any", int option = nr\_flights\*2)

{

cout << "\t From To Time Price " << endl;

cout << "\n";

int temp\_nr\_flights = 0;

int j = 0;

for (int i=0; i <= nr\_flights; i++)

{

if (flightFrom[i] == from || from == "any")

{

if (flightFrom[i] == "none")

{

continue;

}

if (flightDestination[i] == destination || destination == "any")

{

if (flightDate[i] == date || date == "any")

{

if (option == nr\_flights\*2)

{

temp\_nr\_flights+=1;

GeneralShowSingleFlight(flightFrom[i], flightDestination[i], flightDate[i], flightPrice[i], temp\_nr\_flights);

}

else

{

j+= 1;

if (option == j)

return i;

}

}

}

}

}

return temp\_nr\_flights;

}

int newPasswordCheck(string password) // this will tell if a passwordsi valid

{

int alphabetCount = 0, numberCount = 0;

for (int i = 0; i< password.length(); i++)

{

for (int j = 0; j < CheckAlphabet.length(); j++)

{

if (password[i] == CheckAlphabet[j])

{

alphabetCount+=1;

}

if (password[i] == CheckNumber[j])

{

numberCount+=1;

}

}

}

// cout << numberCount << " " << alphabetCount;

if (alphabetCount < 8 || numberCount < 4)

{

return 0;

}

return 1 ;

}

int checkIdPassword(string username, string password, string role ="none") // more validations

{

for (int i = 0; i <= nr\_users; i++)

{

if (username == Username[i])

{

if (password == Password[i])

{

if (role == "isadmin")

{

if (Role[i] == "Admin")

return 1;

else

return 0;

}

if (role == "unknown")

{

return 1;

}

}

if (role == "none")

{

return 1; // giving 1 if no present

}

}

}

return 0;

}

int enterIdPassword(string username, string password, string role)

{

if (checkIdPassword(username, password) == 1)

{

// username already exit

return 0;

}

nr\_users += 1;

Username[nr\_users] = username;

Password[nr\_users] = password;

Role[nr\_users] = role;

storeIdPass();

cout << "added successfully";

}

int subShowAllAnnouncement(string method = "show all", int option = -1, string message = "none")

{

int index = 1;

int none = 0;

for(int i = 0; i<= nr\_announcements; i++)

{

string tempAnnouncement = announcmentMessage[i];

if (tempAnnouncement == "none")

{

none+=1;

continue;

}

if (method == "show all")

{

cout << "\t" << index << ". " ;

index+=1;

for (int j = 0; j<=tempAnnouncement.length(); j++)

{

if (j>=30)

{

cout << " ...";

break;

}

cout << tempAnnouncement[j];

}

cout << endl;

}

if (method == "show specific")

{

if (option == index)

{

cout << "\t" << index << ". " ;

for (int j = 0; j<=tempAnnouncement.length(); j++)

{

if (j%60 == 0 && j!=0)

{

cout << endl<< "\t ";

}

cout << tempAnnouncement[j];

}

cout << endl;

}

index+=1;

// cout << endl;

}

if (method == "remove")

{

if (option == index)

{

announcmentMessage[i] = "none";

cout << endl;

}

index+=1;

}

if (method == "modify")

{

if (option == index)

{

announcmentMessage[i] = message;

cout << endl;

}

index+=1;

}

}

}

void exitToUserInterface(string option, int intoption = 1)

{

if (option == "exit" || option == "Exit" || option == "EXIT" || option == "-1")

{

printUserInterface();

}

if (intoption == -1)

{

printUserInterface();

// printAdminInterface();

}

}

void exitToAdminInterface(string option, int intoption = 1)

{

if (option == "exit" || option == "Exit" || option == "EXIT" || option == "-1")

{

printAdminInterface();

}

if (intoption == -1)

{

printAdminInterface();

}

}

// ==================================================================//

// ================ VERSION 2.1 CHANGES ===================//

void GeneralShowSingleFlight(string flight\_from,string flight\_to,string flight\_date,string flight\_price, int index = -1)

{

string temp;

if (index == -1)

{

cout << "\t " << flight\_from;

}

else

{

cout << "\t" << index << " " << flight\_from;

}

temp = flight\_from;

for (int i = 0; i <(19-temp.length()); i++)

{

cout <<" ";

}

cout << flight\_to;

temp = flight\_to;

for (int i = 0; i <(17-temp.length()); i++)

{

cout <<" ";

}

cout <<flight\_date;

for (int i = 0; i <(19-10); i++)

{

cout <<" ";

}

cout << flight\_price;

if (index != -2)

{

cout << endl;

}

}

int subReturnCountAnnouncement()

{

int our\_no = 0;

for(int i = 0; i <= nr\_announcements; i ++)

{

if (announcmentMessage[i] == "none")

{

continue;

}

our\_no += 1;

}

return our\_no ;

}

int subReturnCountComplains()

{

int our\_no = 0;

for(int i = 0; i < nr\_complains; i ++)

{

if (complainMessages[i] == "none")

{

continue;

}

our\_no += 1;

}

return our\_no;

}

bool isLocationAvailable(string location, string fromTo)

{

if (fromTo == "from")

{

for(int i = 0; i <= nr\_flights; i++)

{

if (flightFrom[i] == location)

{

return true;

}

}

}

else if (fromTo == "to")

{

for(int i = 0; i <= nr\_flights; i++)

{

if (flightDestination[i] == location)

{

return true;

}

}

}

return false;

}

void gotostartupscreen(string option)

{

if (option == "exit" || option == "Exit" || option == "EXIT" || option == "-1")

printStartupScreen();

}

string enterDestinationLocation()

{

string to;

while(true)

{

cout << "\n\n";

cout << "\t Enter Your destination(Name, any): ";

cin >> to;

exitToUserInterface(to);

if (to == "any")

{

return to;

}

if (isLocationAvailable(to, "to"))

{

return to;

break;

}

else

{

cout << "\t please enter valid information";

}

}

}

string enterFromLocation()

{

string from;

while(true)

{

cout << "\n\n";

cout << "\t Enter pickup point(Name, any): ";

cin >> from;

exitToUserInterface(from);

if (from == "any")

{

return from;

}

if (isLocationAvailable(from, "from"))

{

return from;

break;

}

else

{

cout << "\t no flight available from this location";

}

}

}

string enterDateCorrectly()

{

string fulldate;

while(true)

{

cout << "\n\n";

cout << "\t Enter Your fullDate(use 01-01-2003, any): ";

cin >> fulldate;

exitToUserInterface(fulldate);

if (fulldate == "any")

{

return fulldate;

}

if (fulldate.length() != 10)

{

cout << "\t please enter valid info";

continue;

}

fulldate[2] = '-';

fulldate[5] = '-';

return fulldate;

}

}

string enterfoodCorrectly()

{

// cin.ignore();

string food;

while (true)

{

cout << "\n";

cout << "\t Enter new food type (veg, non-Veg): ";

cin >> food;

exitToUserInterface(food);

if (food == "non-veg" || food == "Non-Veg" || food == "Non-veg" || food == "non veg")

{

food = "Non-Veg";

break;

}

if (food == "veg" || food == "Veg")

{

food = "Veg";

break;

}

cout << "\t please chose from above options, try again \n\n ";

}

return food;

}

void changePrice()

{

int temp\_nr\_flights;

temp\_nr\_flights = displayFlight();

if (temp\_nr\_flights == 0)

{

cout << "\n\tNo flights available, returning...";

getch();

printAdminInterface();

}

int option;

while(true)

{

cout << "\n\tEnter flight you want to change Price: ";

cin >> option;

exitToAdminInterface("any", option);

if (option < 1 || option > temp\_nr\_flights)

{

cout << "invalid option, try again\n";

}

else

break;

}

string newPrice;

cout << "\n\tenter new price: ";

cin >> newPrice;

int j=1;

for(int i = 0; i <= nr\_flights; i++)

{

if (flightFrom[i] == "none")

{

continue;

}

if(j==option)

{

int j = i;

// change flight from database

flightPrice[i] = newPrice;

cout << "\n\tflight changes successfully";

}

j+=1;

}

storeAvailableFlights();

getch();

printAdminInterface();

}

string printTime()

{

time\_t ct = time(0);

string currenttime = ctime(&ct);

return currenttime;

}

// Complain Section Can do a fun here maybe

void seeComplain()

{

int index = 1;

cout << " Your complains are: \n\n";

for(int i = 0; i<=nr\_complains; i++)

{

if(complainId[i] == currentId)

{

cout << "\t "<< index << ". " << complainMessages[i] << "\n\n";

index+=1;

}

}

if (index == 1)

{

cout << "\t no complains available to see";

}

// getch();

printUserInterface();

}

void removeComplain()

{

int nr\_complain = 0;

int index = 1;

cout << " Your all complains are: \n\n"; // showing all complains

for(int i = 0; i<=nr\_complains; i++)

{

if(complainId[i] == currentId)

{

nr\_complain+=1;

cout << "\t"<< index << ". " << complainMessages[i] << "\n\n";

index+=1;

}

}

if (nr\_complain == 0)

{

cout << "\n no complains available.";

getch();

printUserInterface();

}

int option;

while(true)

{

cout << " enter complain you want to remove: "; // selecting complains

cin >> option;

exitToUserInterface("any", option);

if (option < 1 || option > nr\_complain)

{

cout << "\t enter valid nr.\n";

}

else

break;

}

int j=1;

for(int i = 0; i<=nr\_complains; i++)

{

if(complainId[i] == currentId)

{

if(option == j)

{

cout << "\n The complain, \n ----> " << complainMessages[i] << "\n has been deleted." << endl;

complainId[i] = "none";

complainMessages[i] = "none";

}

j+=1;

}

}

storeComplains();

getch();

printUserInterface();

}

void addFileComplain()

{

string userComplain ;

cout << "\n\t";

cout << "Enter complain your : "; // selecting complains

cin.ignore();

getline(cin, userComplain);

exitToUserInterface(userComplain); // if -1 entered, go back

cout << "\n\t";

nr\_complains += 1;

complainId[nr\_complains] = currentId;

complainMessages[nr\_complains] = userComplain;

cout << "\n\n\t Your message has been saved";

storeComplains();

// getch();

printUserInterface();

}

void adminPrintComplain()

{

int tempNrComplains = 0;

cout << "\n These are complains that are issues by user\n\n" << endl;

cout << "\tUser id Message\n"<< endl; // 31

for(int i = 0; i<= nr\_complains; i++)

{

if (complainMessages[i] != "none")

{

tempNrComplains += 1;

cout << "\t" << tempNrComplains << " " << complainId[i] ;

string id\_temp = complainId[i];

for(int i = 0; i < (31-id\_temp.length()); i++ )

{

cout << " ";

}

string tempMessage = complainMessages[i];

for(int j = 0; j<=tempMessage.length();j++)

{

if (j >= 20)

{

cout << "....." ;

break;

}

cout << tempMessage[j];

}

cout << endl;

}

}

if (tempNrComplains == 0)

{

cout << "no complains available";

getch();

printAdminInterface(); // go back to admin interface

}

int option;

while(true)

{

cout << "\n\tenter message you see properly: ";

cin >> option;

exitToAdminInterface("any", option);

if (option <1 || option > tempNrComplains)

{

cout << "\tinvalid complain id, try again; \n";

}

else

break;

}

int j = 1;

for(int i = 0; i<=nr\_complains; i++)

{

if (complainMessages[i] != "none")

{

if(option == j)

{

cout << "\n\t---> " << complainMessages[i]; // showing message

}

j+=1;

}

}

getch();

printAdminInterface();

}

// Announcement Section

void seeAnnouncements()

{

cout << "\nAll aannouncements are: \n\n";

subShowAllAnnouncement("show all");

int option;

while(true)

{

cout << "\nchose announcement you want to see: ";

cin >> option;

exitToUserInterface("any", option);

// cout << "\n" << announcmentMessage[option-1];

if(option < 1 || option > subReturnCountAnnouncement())

{

cout << "please enter valid announcement number\n";

}

else

{

break;

}

}

subShowAllAnnouncement("show specific", option); // show announcemtn by a admin

getch();

printUserInterface();

}

void adminModifyAccnouncement()

{

cout << "chose announcement you want to Modify \n" << endl;

subShowAllAnnouncement("show all");

int option;

while(true)

{

cout << "\nchose announcement you want to modify: ";

cin >> option;

exitToAdminInterface("any", option); ); // if -1 entered, go back

if (option < 1 || option > (subReturnCountAnnouncement()))

{

cout << "invalid option, try again\n\n";

}

else

break;

}

string new\_announcement;

cout << "\nEnter new announcement";

cin.ignore();

getline(cin, new\_announcement);

subShowAllAnnouncement("modify", option, new\_announcement);

cout << "Selected announcement has been modified"; // enter a number for ann

storeAnnouncments();

// getch();

printAdminInterface();

}

void adminRemoveAccnouncement() // remove from database

{

cout << " chose announcement you want to reomve " << endl;

subShowAllAnnouncement("show all");

int option;

while(true)

{

cout << "\n chose announcement you want to remove: ";

cin >> option;

exitToAdminInterface("any", option);

// if (option < 1 || option > nr\_announcements+1)

if (option < 1 || option > (subReturnCountAnnouncement()))

{

cout << " invalid option, try again\n\n";

}

else

break;

}

subShowAllAnnouncement("remove", option);

cout << " Selected announcement has been removed";

storeAnnouncments();

getch();

printAdminInterface();

}

void adminAddAccnouncement()

{

string announcement;

cout << "\t enter new announcement: ";

cin.ignore();

getline(cin, announcement);

exitToAdminInterface(announcement);

nr\_announcements+=1;

announcmentMessage[nr\_announcements] = announcement;

cout << "\n\t Your record has been saved ";

storeAnnouncments();

// getch();

printAdminInterface();

}

// Admin Interface Section

int adminAddFlight()

{

string to, from, date, price;

cout << "\n";

cout << "\tEnter flight Departure spot: ";

cin >> from;

exitToAdminInterface(from);

cout << "\tEnter flight Destination: ";

cin >> to;

exitToAdminInterface(to);

cout << "\tEnter Date of flight: ";

cin >> date;

exitToAdminInterface(date);

cout << "\tEnter Price of flight: ";

cin >> price;

exitToAdminInterface(price);

nr\_flights+=1;

flightFrom[nr\_flights] = from;

flightDestination[nr\_flights] = to;

flightDate[nr\_flights] = date;

flightPrice[nr\_flights] = price;

cout << "\nThe following flight has been added in database\n\n";

cout << "\t From To Time Price " << endl;

int j = nr\_flights;

GeneralShowSingleFlight(flightFrom[j], flightDestination[j], flightDate[j], flightPrice[j]);

// cout << "\t" << flightFrom[nr\_flights] << " " << flightDestination[nr\_flights] << " "<< flightDate[nr\_flights] << " " << flightPrice[nr\_flights] << endl;

storeAvailableFlights();

getch();

printAdminInterface();

}

int adminRemoveFlight() // option validation self

{

int temp\_nr\_flights = 0;

// cout << "\t From To Time Price " << endl;

temp\_nr\_flights = displayFlight();

if (temp\_nr\_flights == 0)

{

cout << "\n\tNo flights available, returning...";

getch();

printAdminInterface();

}

int option;

while(true)

{

cout << "\n\tEnter flight you want to remove: ";

cin >> option;

exitToAdminInterface("any", option);

if (option < 1 || option > temp\_nr\_flights)

{

cout << "invalid option, try again\n";

}

else

break;

}

int j=1;

for(int i = 0; i <= nr\_flights; i++)

{

if (flightFrom[i] == "none")

{

continue;

}

if(j==option)

{

int j = i;

GeneralShowSingleFlight(flightFrom[j], flightDestination[j], flightDate[j], flightPrice[j]);

for(int k = 0; k <= nr\_booking; k++)

{

if (bookedUsersFlight[k] == i)

{

bookedUsersId[k] = "none";

bookedUsersFlight[k] = -1;

bookedUsersFood[k] = "none";

}

}

// removing flight from database

flightFrom[i] = "none";

flightDestination[i] = "none";

flightDate[i] = "none";

flightPrice[i] = "none";

}

j+=1;

}

storeAvailableFlights();

getch();

printAdminInterface();

}

int adminSeeBookedFlight()

{

cout << "\t ID From To Time Price " << endl;

int tempIndex = 1;

for(int i = 0; i <= nr\_booking; i++)

{

int j = bookedUsersFlight[i];

if (bookedUsersId[i] == "none" || flightFrom[j] == "none")

{

continue;

}

string temp;

cout << "\t" << tempIndex << " " << bookedUsersId[i];

tempIndex += 1;

temp = bookedUsersId[i];

for (int i = 0; i <(19-temp.length()); i++)

{

cout <<" ";

}

cout << flightFrom[j];

temp = flightFrom[j];

for (int i = 0; i <(19-temp.length()); i++)

{

cout <<" ";

}

cout << flightDestination[j] ;

temp = flightDestination[j] ;

for (int i = 0; i <(17-temp.length()); i++)

{

cout <<" ";

}

cout <<flightDate[j];

for (int i = 0; i <(19-10); i++)

{

cout <<" ";

}

cout << flightPrice[j] << endl;

// cout << bookedUsersId[i] << " " << bookedUsersFood[i] << " " << flightFrom[j] << " " << flightDestination[j] << " " << flightDate[j] << " " << flightPrice[j];

// cout << endl;

}

cout << "\n\nPress any key to continue ;" <<endl;

// getch();

printAdminInterface();

}

void adminSearchFlight()

{

string to, from, date;

cout << "enter pickup point (name/ any): ";

cin >> from;

exitToAdminInterface(from);

cout << "enter destinaton point (name/ any): ";

cin >> to;

exitToAdminInterface(to);

cout << "enter Date (01-01-2003/ any): ";

cin >> date;

cout << "\n";

exitToAdminInterface(date);

displayFlight(from=from, to, date);

cout << "\nAll results re shown...";

getch();

printAdminInterface();

}

void adminRemoveCustomer()

{

cout << " Chose user you want to remove. He/She must be User \n\n";

cout << "\t UserID Passward Role\n" << endl;

string temp;

int temp\_nr\_customers = 0;

// cout << ;

for(int i = 0; i<=nr\_users; i++)

{

if(Role[i] == "User")

{

temp\_nr\_customers +=1;

temp = Username[i];

cout << "\t" << temp\_nr\_customers << " " << Username[i];

for (int i = 0; i < (16-temp.length()); i++)

{

cout << " ";

}

temp = Password[i];

cout << Password[i];

for (int i = 0; i < (18-temp.length()); i++)

{

cout << " ";

}

cout << Role[i] << endl;

}

}

if (temp\_nr\_customers == 0)

{

cout << "\n No Users to show";

getch();

printAdminInterface();

}

int option;

while(true)

{

cout << "\n enter user you want to remove";

cin >> option;

exitToAdminInterface("any", option);

if (option < 1 || option > temp\_nr\_customers)

{

cout << "\n invalid option, try again\n";

}

else

break;

}

int j = 1;

for(int i = 0; i<=nr\_users; i++)

{

if(Role[i] == "User")

{

if (j==option)

{

// removing lfights if any

for (int k = 0; k<= nr\_flights; k++)

{

if (bookedUsersId[k] == Username[i])

{

bookedUsersId[k] = "none";

bookedUsersFlight[k] = -1;

}

}

// removing from database

Username[i] = "none";

Password[i] = "none";

Role[i] = "none";

}

j+=1;

}

}

cout << "\n User removed Successfully and with his bookings";

// storeIdPass(); // atoring all data

// storeBookedFlights(); // stirng all data

getch();

printAdminInterface();

}

// Help Section

void printHelpMenu()

{

cout << "This is help menu \n\n";

cout << "Contact No: "<< contactNo << "\n";

cout << "Email: " << email << endl;

cout << helpMessage;

// getch();

printUserInterface();

}

void adminEditHelpMenu()

{

string option;

cout << "\ndo you want to change contact nr(y/n): "; // asking option

cin >> option;

exitToAdminInterface(option);

if (option == "y")

{

string new\_contact;

cout << "enter new contact nr: ";

cin >> new\_contact;

exitToAdminInterface(new\_contact);

contactNo = new\_contact;

}

cout << "\ndo you want to change Email(y/n): ";

cin >> option;

exitToAdminInterface(option);

if (option == "y")

{

string new\_email;

cout << "enter new email: ";

cin >> new\_email;

exitToAdminInterface(new\_email);

email = new\_email;

}

cout << "\ndo you want to change Help Message(y/n): ";

cin >> option;

exitToAdminInterface(option);

if (option == "y")

{

string new\_message;

cout << "enter new message: ";

cin >> new\_message;

helpMessage = new\_message;

}

cout << "\ndo you want to reset evrything to default: "; // asking if wnt to reset

cin >> option;

exitToAdminInterface(option);

if (option == "y")

{

contactNo = "04665952275";

email = "flightmanagement@gmail.com";

helpMessage = "You can find help here";

}

cout << "\nDone. Press anything to go back to admin interface...";

printAdminInterface();

}

// User Interface Section

void bookFlight()

{

string name, passport, from, destination, fullDate, timings, foodType;

cout << "\n\n";

cout << " Chose Option" << endl;

cout << "\n\n\n";

cout << "\t Enter Your Name: ";

cin >> name;

exitToUserInterface(name);

while (true)

{

cout << "\n\n";

cout << "\t Enter 7 digit passport Number ";

cin >> passport;

exitToUserInterface(passport);

if (passport.length() != 7)

{

cout << "\t please enter valid passport no.";

}

else

{

break;

}

}

from = enterFromLocation();

destination = enterDestinationLocation();

fullDate = enterDateCorrectly();

cout << "\n";

foodType = enterfoodCorrectly();

system("cls");

printHeader();

cout << "\n\t Available Flights are: \n\n";

int temp\_ans = displayFlight(from, destination, fullDate);

if (temp\_ans == 0)

{

int option;

cout << "\t no flights avail" << endl;

cout << "\t Enter 1 to try again, enter 2 to go back: ";

cin >> option;

exitToUserInterface("any", option);

if (option == 1)

{

printHeader();

bookFlight();

}

else

{

printUserInterface();

}

}

cout << "\n\n";

int option;

while(true)

{

cout << "\n\n\t chose option: ";

cin >> option;

cout << "\n";

exitToUserInterface("any", option);

if (option< 1 || option > temp\_ans)

{

cout << "\t please enter valid flight.\n";

}

else

break;

}

int selectedOption = displayFlight(from, destination, fullDate, option);

string temp = flightFrom[selectedOption];

cout << "\t " << flightFrom[selectedOption];

for(int i = 0; i < (19- temp.length()); i++)

{

cout << " ";

}

temp = flightDestination[selectedOption];

cout << flightDestination[selectedOption];

for(int i = 0; i < (17- temp.length()); i++)

{

cout << " ";

}

cout << flightDate[selectedOption];

cout << " ";

cout << flightPrice[selectedOption] << endl;

nr\_booking+=1;

bookedUsersId[nr\_booking] = currentId;

bookedUsersFlight[nr\_booking] = selectedOption;

bookedUsersFood[nr\_booking] = foodType;

cout << "\n\t This flight has been booked.\n\n";

storeBookedFlights();

// cout << bookedUsersId[nr\_booking] << bookedUsersFlight[nr\_booking] << bookedUsersFood[nr\_booking]; // storing in txt

cout << "\t Press anything to go back to user interface";

getch();

printUserInterface();

}

void cancelFlight()

{

cout << "\n";

cout << "\t From To Time Price " << endl;

int temp\_nr\_flights=0;

for (int i = 0; i <= nr\_booking; i++)

{

if (bookedUsersId[i] == currentId)

{

int j = bookedUsersFlight[i];

if (flightFrom[j] == "none" || flightDestination[j] == "none") // if db shows none

{

continue;

}

// printing with proper syntax

temp\_nr\_flights += 1;

GeneralShowSingleFlight(flightFrom[j], flightDestination[j], flightDate[j], flightPrice[j], temp\_nr\_flights);

}

}

if (temp\_nr\_flights == 0)

{

cout << "\n\t no flight available to cancel, going back";

getch();

printUserInterface();

}

int option;

while(true)

{

cout << "\n\t enter flight you want to cancel: "; // asking for input

cin >> option;

cout << "\n";

exitToUserInterface("any", option);

if (option< 1 || option > temp\_nr\_flights)

{

cout << "\t plz enter valid flight nr.\n";

}

else

break;

}

int k=1;

for (int i = 0; i <= nr\_booking; i++)

{

if (bookedUsersId[i] == currentId)

{

if (option == k)

{

int j = bookedUsersFlight[i];

if (flightFrom[j] == "none" || flightDestination[j] == "none") // means if null in db

{

continue;

}

int selectedOption = j;

GeneralShowSingleFlight(flightFrom[selectedOption], flightDestination[selectedOption], flightDate[selectedOption], flightPrice[selectedOption], option);

// string temp = flightFrom[selectedOption];

// cout << "\t " << flightFrom[selectedOption];

// for(int i = 0; i < (19- temp.length()); i++)

// {

// cout << " "; // printing a space

// }

// temp = flightDestination[selectedOption];

// cout << flightDestination[selectedOption];

// for(int i = 0; i < (17- temp.length()); i++)

// {

// cout << " ";

// }

// cout << flightDate[selectedOption];

// cout << " ";

// cout << flightPrice[selectedOption] << endl;

// cout << "\n\t " << flightFrom[j] << " " << flightDestination[j] << " "<< flightDate[j] << " " << flightPrice[j] << endl;

bookedUsersId[i] = "none"; /// storing in db

bookedUsersFlight[i] = -1; // storing in db as a null

bookedUsersFood[i] = "none";

}

k+=1;

}

}

storeBookedFlights();

cout << "\n\t This Flight has been cancelled ";

getch();

printUserInterface();

}

void displaySpecificUserFlight()

{

int temp\_nr\_flights = 0;

cout << "\t From To Time Price " << endl;

cout << "\n";

for (int i = 0; i <= nr\_booking; i++)

{

if (bookedUsersId[i] == currentId)

{

int j = bookedUsersFlight[i];

if (flightFrom[j] == "none" || flightDestination[j] == "none")

{

continue;

}

// proper index

temp\_nr\_flights+=1;

GeneralShowSingleFlight(flightFrom[j], flightDestination[j], flightDate[j], flightPrice[j], temp\_nr\_flights);

// old no indendation

// cout << "\t\t" << flightFrom[j] << " " << flightDestination[j] << " "<< flightDate[j] << " " << flightPrice[j] << endl;

}

}

if (temp\_nr\_flights == 0)

{

cout << "\n\tNo flights are boooked as of now.";

}

// getch();

printUserInterface();

}

void changeFood()

{

int temp\_nr\_flights = 0;

cout << "Please chose flight you want to change food type \n";

cout << "\t From To Time Food Type " << endl;

cout << "\n";

for (int i = 0; i <= nr\_booking; i++)

{

if (bookedUsersId[i] == currentId && flightFrom[bookedUsersFlight[i]] != "none")

{

int j = bookedUsersFlight[i];

// proper indentation

temp\_nr\_flights+=1;

string temp;

cout << "\t" << temp\_nr\_flights << " "<< flightFrom[j];

temp = flightFrom[j];

for (int i = 0; i <(19-temp.length()); i++)

{

cout <<" ";

}

cout << flightDestination[j];

temp = flightDestination[j];

for (int i = 0; i <(17-temp.length()); i++)

{

cout <<" ";

}

cout <<flightDate[j];

for (int i = 0; i <(19-10); i++)

{

cout <<" ";

}

cout << bookedUsersFood[i] << endl;

}

}

if (temp\_nr\_flights == 0)

{

cout << "\nNo flight has booked by user. Returning...";

getch();

printUserInterface();

}

int option;

while(true)

{

cout << "\n\t Enter flight you want to change food: ";

cin >> option;

exitToUserInterface("any", option);

if (option < 1 || option > temp\_nr\_flights)

{

cout << "please enter right option\n";

}

else

break;

}

string new\_food;

new\_food = enterfoodCorrectly();

int k = 1;

for (int i = 0; i <= nr\_booking; i++)

{

if (bookedUsersId[i] == currentId && flightFrom[bookedUsersFlight[i]] != "none")

{

if (k == option)

{

bookedUsersFood[i] = new\_food;

cout << "\n\t changed successsfully";

}

k+=1;

}

}

storeBookedFlights();

getch();

printUserInterface();

}

int changeId()

{

string oldPassword, newId;

cout << " You must first validate before changing password" << endl;

cout << " enter old password: ";

cin >> oldPassword;

exitToUserInterface(oldPassword);

int tempUserIndex = -1;

for (int i = 0; i <=nr\_users; i++)

{

if (currentId == Username[i])

{

if (oldPassword == Password[i])

{

cout << "\n Validation Successful";

tempUserIndex = i;

}

else

{

cout << "\n Validation failed, ";

getch();

printUserInterface();

}

}

}

if (tempUserIndex == -1)

{

cout << "\n Validation failed, ";

getch();

printUserInterface();

}

cout << " enter new username: ";

cin >> newId;

exitToUserInterface(newId);

Username[tempUserIndex] = newId;

cout << " Id changed";

storeIdPass();

getch();

printUserInterface();

}

int changePassword()

{

string oldPassword, newPassword;

cout << " You must first validate before changing password" << endl;

cout << " enter old password: ";

cin >> oldPassword;

exitToUserInterface(oldPassword);

int tempUserIndex;

for (int i = 0; i <=nr\_users; i++)

{

if (currentId == Username[i])

{

if (oldPassword == Password[i])

{

cout << "\n Validation Successful";

tempUserIndex = i;

}

else

{

cout << "\n Validation failed, ";

getch();

printUserInterface();

}

}

}

cout << "\n enter new password: ";

cin >> newPassword;

exitToUserInterface(newPassword);

if (newPasswordCheck(newPassword) == 0)

{

cout << " invalid, follow insturctions, returning";

getch();

printHeader();

changePassword();

}

Password[tempUserIndex] = newPassword;

cout << " Password changed";

storeIdPass();

getch();

printUserInterface();

}

// Main Interfaces

void printAdminInterface()

{

cin.ignore();

printHeader();

cout << "\n\n";

cout << " Chose Option" << endl;

cout << "\n\n";

cout << "\t 1. Add Flight 7. Search Flights " ;

cout << "\n";

cout << "\t 2. Remove Flight 8. Change Price";

cout << "\n";

cout << "\t 3. Check Booked Flights 9. Add announcement";

cout << "\n";

cout << "\t 4. See Complain Box 10. Remove announcement ";

cout << "\n";

cout << "\t 5. See How To Guide 11. Modify announcement ";

cout << "\n";

cout << "\t 6. Change Contact info 12. Remove customer ";

cout << "\n";

cout << "\t 13. Exit ";

cout << "\n\n";

string option;

// option = " ";

printEnterOption();

// cin.ignore();

getline(cin, option);

if ( !SeachWord("see", option) && !SeachWord("remove", option) && (option == "1" || ( ( SeachWord("enter", option) || SeachWord("add", option) || SeachWord("new", option)) && (SeachWord("flight", option) || SeachWord("flights", option)) )))

{

// Book Flight

printHeader();

adminAddFlight();

}

else if (option == "2" || ( ( SeachWord("remove", option) || SeachWord("cancel", option) || SeachWord("delete", option)) && (SeachWord("flight", option) || SeachWord("flights", option) )) )

{

// Remove Flight

printHeader();

adminRemoveFlight();

}

else if (option == "3" || ( ( SeachWord("see", option) || SeachWord("show", option) || SeachWord("check", option) || SeachWord("booked", option)) && (SeachWord("flight", option) || SeachWord("flights", option)) ))

{

// Book Flight

// getch();

printHeader();

adminSeeBookedFlight();

}

else if (option == "4" || (( SeachWord("see", option) || SeachWord("show", option) || SeachWord("box", option)) && (SeachWord("complain", option) || SeachWord("complains", option) )) )

{

// Add Complain

printHeader();

adminPrintComplain();

}

else if (option == "5" || SeachWord("guide", option))

{

// See how to guide

// getch();

// printHeader();

animStartupOneTimeAnim();

printAdminInterface();

}

else if (option == "6" || ( SeachWord("help", option) || (SeachWord("contact", option)) ) )

{

// Change help menu

printHeader();

adminEditHelpMenu();

}

else if (option == "7" || ( SeachWord("search", option) || (SeachWord("find", option) )) && ((SeachWord("flight", option) || (SeachWord("flights", option)) ) ) )

{

// Search Flight

printHeader();

adminSearchFlight();

}

else if (option == "12" || (( SeachWord("remove", option) || SeachWord("delete", option) ) && ( SeachWord("customer", option) || SeachWord("customers", option) ) ) )

{

// Remove Customer

printHeader();

adminRemoveCustomer();

}

else if ( !SeachWord("see", option) && !SeachWord("remove", option) && (option == "9" || (( SeachWord("add", option) || SeachWord("new", option) ) && ( SeachWord("announcement", option) || SeachWord("announcements", option) ) ) ))

{

// Add Announcement

printHeader();

adminAddAccnouncement();

}

else if (option == "10"|| (( SeachWord("remove", option) || SeachWord("cancel", option) || SeachWord("delete", option)) && ( SeachWord("announcement", option) || SeachWord("announcements", option) ) ))

{

// Remove Announcement

printHeader();

adminRemoveAccnouncement();

}

else if (option == "11" || (( SeachWord("modify", option) || SeachWord("edit", option)) && ( SeachWord("announcement", option) || SeachWord("announcements", option) ) ) )

{

// modify aannouncement

printHeader();

adminModifyAccnouncement();

}

else if (option == "8" || SeachWord("price", option) || SeachWord("Price", option) )

{

// change price

printHeader();

changePrice();

}

else if (option == "13" || SeachWord("Exit", option) || SeachWord("exit", option) )

{

exit();

}

else

{

cout << "no valid option selected";

getch();

printAdminInterface();

}

}

void printUserInterface()

{

cin.ignore();

printHeader();

cout << "\n\n";

cout << " Chose Option" << endl;

cout << "\n\n";

cout << "\t 1. Book Flight 7. Change Food";

cout << "\n";

cout << "\t 2. Cancel Flight 8. See Announcements";

cout << "\n";

cout << "\t 3. See My Flights 9. Change Username";

cout << "\n";

cout << "\t 4. File Complain 10. Change Password ";

cout << "\n";

cout << "\t 5. See Complain 11. Help ";

cout << "\n";

cout << "\t 6. Remove Complain 12. Exit ";

cout << "\n\n";

string option;

printEnterOption();

// cin.ignore();

getline(cin, option);

if ( !SeachWord("see", option) && !SeachWord("cancel", option) && (option == "1" || ( (SeachWord("book", option)||SeachWord("add", option)||SeachWord("new", option)) && (SeachWord("flight", option)||SeachWord("flights", option)) )))

{

// Book Flight

printHeader();

bookFlight();

}

else if (option == "2" || ( (SeachWord("cancel", option)||SeachWord("remove", option)||SeachWord("delete", option)) && (SeachWord("flight", option)||SeachWord("flights", option)) ))

{

// Book Flight

printHeader();

cancelFlight();

}

else if (option == "3" || ( (SeachWord("see", option)||SeachWord("show", option)||SeachWord("boooked", option)) && (SeachWord("flight", option)||SeachWord("flights", option)) ))

{

// Book Flight

printHeader();

displaySpecificUserFlight();

}

else if ( !SeachWord("see", option) && !SeachWord("remove", option) && (option == "4" || ( (SeachWord("add", option)||SeachWord("new", option)||SeachWord("file", option)) && (SeachWord("complain", option)||SeachWord("complains", option)) )))

{

// Add Complain

printHeader();

addFileComplain();

}

else if (option == "5" || ( (SeachWord("see", option)||SeachWord("show", option)) && (SeachWord("complain", option)||SeachWord("complains", option)) ))

{

// See Complain

printHeader();

seeComplain();

}

else if (option == "6" || ( (SeachWord("remove", option)||SeachWord("delete", option)) && (SeachWord("complain", option)||SeachWord("complains", option)) ))

{

// Remove Complain

printHeader();

removeComplain();

}

else if (option == "7" || SeachWord("food", option) || SeachWord("Food", option))

{

// Change Food

printHeader();

changeFood();

}

else if (option == "8" || ( (SeachWord("see", option)||SeachWord("show", option)) && (SeachWord("announcements", option)||SeachWord("announcement", option)) ))

{

// See announcements

printHeader();

seeAnnouncements();

}

else if (option == "9" || ( (SeachWord("change", option)||SeachWord("edit", option)) && (SeachWord("id", option)||SeachWord("username", option)||SeachWord("Username", option) ) ) )

{

// Change Id

printHeader();

changeId();

}

else if (option == "10" || ( (SeachWord("change", option)||SeachWord("edit", option)) && (SeachWord("password", option)||SeachWord("passward", option)||SeachWord("Password", option) ) ))

{

// Change Password

printHeader();

changePassword();

}

else if (option == "11" || SeachWord("help", option) || SeachWord("contact", option) || SeachWord("Help", option) )

{

printHeader();

printHelpMenu();

}

else if (option == "12" || SeachWord("exit", option) || SeachWord("Exit", option) )

{

exit();

}

else

{

cout << "no valid option selected";

getch();

printUserInterface();

}

}

// Sub Interfaces

void printStartupScreen()

{

system("cls");

system("Color 03");

printHeader();

cout << "\n\n";

cout << " Chose Option" << endl;

cout << "\n\n\n";

cout << "\t 1.Login";

cout << "\n\n";

cout << "\t 2.Signup";

cout << "\n\n\n";

string option;

// bool allow = 1;

while(true)

{

cout << "\n\n Enter option: ";

cin >> option;

if (option == "1")

{

// login Screen

// printHeader();

printLoginScreen();

}

else if (option == "2")

{

// login Screen

// printHeader();

printSignupScreen();

}

else

{

cout << "please select valid option";

}

}

}

void printSignupScreen()

{

printHeader(); // print on top on screen after cls

string username, password, role;

cout << "\n\n";

cout << " Chose Option" << endl;

cout << "\n\n\n";

while(true)

{

cout << "\n";

cout << "\t Enter Username: ";

cin >> username;

gotostartupscreen(username);

if (checkIdPassword(username, "none") == 1) // 1 means id already in db

{

cout << "\t already exist, plz try again\n";

}

else{

break;

}

}

cout << "\n\n";

while(true)

{

cout << "\n\t Enter a 8 digit password (must have atleast 7 characters and 3 numeric digits): \n\t ";

cin >> password;

gotostartupscreen(password);

if (newPasswordCheck(password) == 0)

{

cout << "\t invalid pass plz enter full\n";

}

else{

break;

}

}

cout << "\n";

while(true)

{

cout << "\n\t Chose Admin or User: ";

cin >> role;

gotostartupscreen(role);

if (role == "user" || role == "User")

{

role = "User";

break;

}

else if (role == "admin" || role == "Admin) // helps in user input

{

role = "Admin";

break;

}

cout << "\t please chose from admin or user\n";

}

if (role == "Admin")

{

system("cls");

string adminpass, adminid;

cout << "\n\n\n";

cout << "\t to register as admin, you must first ask an admin to enter his credentials\n";

cout << "\n\n\t Enter Username of a admin: ";

cin >> adminid;

cout << "\n\n";

cout << "\t Enter password for admin : ";

cin >> adminpass;

if (checkIdPassword(adminid, adminpass, "isadmin") == 1)

{

cout << "\n\t authentecation succesful, Admin ";

enterIdPassword(username, password, role);

}

else{

cout << "\n\t Admin Cannot be verified, please try again from start";

getch();

printHeader();

printSignupScreen();

}

cout << "\n\n\n";

}

else if (role == "User") // if matchrd only then

{

cout << "\n\t ";

enterIdPassword(username, password, role); // checking id pass, if matches

}

// cout << "\n\n\n";

// cout << "\t Enter Username of a admin: ";

// cout << "\n\n\n";

// cout << "\t Enter password for admin : ";

// cout << "\n\n\n";

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

everyEnd();

// Startup Screen

printStartupScreen();

}

void printLoginScreen()

{

printHeader();

string username, password;

cout << "\n\n";

cout << " Chose Option" << endl;

bool correctinfo = false;

cout << "\n\n\n";

while(correctinfo == false) // only when not already in database

{

cout << "\t Enter Username: ";

cin >> username;

gotostartupscreen(username);

cout << "\n\n";

cout << "\t Enter Password: ";

cin >> password;

gotostartupscreen(password);

if (checkIdPassword(username, password, "unknown") == 1)

{

cout << "\n\t Authentication successful";

correctinfo = true;

for (int i = 0; i <= nr\_users; i++)

{

if (Username[i] == username)

{

currentId = Username[i];

currentrole = Role[i];

}

}

}

else{

cout << "\n\t Authentication unsuccessful, plz try again";

Sleep(1500);

printLoginScreen();

cout << "\n\n";

}

}

cout << "\n\n\n";

everyEnd();

if (isAnimShown == false) // thid is so only first time anim is shown

{

animStartupOneTimeAnim();

isAnimShown = true;

}

if (currentrole == "Admin")

{

// User Admin Screen

printHeader();

printAdminInterface();

printEnterOption();

}

else if (currentrole == "User")

{

// User interface Screen

// printHeader();

printUserInterface();

printEnterOption();

}

}

void printHeader() // printing header

{

system("cls");

cout << "\n\n";

cout << "\n";

// cout << "\t\t\t\tflight management system " ;

cout << "\t\t\t\tFLIGHT MANAGEMENT SYSTEM " ;

cout << "\n" ;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " " << printTime();

}

// ============================================================= //

void printEnterOption()

{

cout << "\n\n";

cout << " Enter Your Option...";

}

void everyEnd()

{

cout << "\n\n enter any key to continue";

getch();

}

void exit()

{

currentrole = "none";

currentId = "none";

// startup Screen

// system("cls");

// printHeader();

printStartupScreen();

printEnterOption();

}

// ================================== //

// anim functions //

void animhowToUseFull()

{

system("cls"); // clearing screen

// getch();

cout << "\n\n";

animprintHowToLogo();

cout << "\n\n";

animprintRobo();

getch();

}

void animprintAdminInterface()

{

// printHeader();

cout << "\n\n";

cout << " Chose Option" << endl;

cout << "\n\n";

cout << "\t 1. Add Flight 7. Search Flights";

cout << "\n";

cout << "\t 2. Remove Flight 8. Remove customer";

cout << "\n";

cout << "\t 3. Check Booked Flights 9. Add announcement";

cout << "\n";

cout << "\t 4. See Complain Box 10. Remove announcement ";

cout << "\n";

cout << "\t 5. Change Appearence 11. Modify announcement ";

cout << "\n";

cout << "\t 6. Change Contact info 12. Exit ";

cout << "\n\n";

cout << "\n\n\tEnter your choice: ";

}

void animStartupOneTimeAnim() // showing animation

{

animhowToUseFull(); // showing robo guide

system("cls");

printHeader();

animprintAdminInterface();

cout << "\n\n";

animprintRoboInterface();

getch();

}

// VERSION 2.0 CHANGES //

void storeIdPass()

{

fstream file1;

file1.open("textFiles/id\_Pass.txt", ios::out);

for( int i= 0 ; i <= nr\_users ; i++)

{

file1 << Username[i] << "," << Password[i] << "," << Role[i] << endl; // storing data

}

file1.close();

}

void loadIdPass()

{

nr\_users = 0;

string record = "";

fstream file1;

file1.open("textFiles/id\_Pass.txt", ios::in); // opening file in input mode

while(getline(file1, record))

{

Username[nr\_users] = getFieldFromRecord(record, 0);

Password[nr\_users] = getFieldFromRecord(record, 1);

Role[nr\_users] = getFieldFromRecord(record, 2);

nr\_users += 1;

}

nr\_users -=1;

file1.close();

}

void storeAnnouncments()

{

fstream file1;

file1.open("textFiles/announcements.txt", ios::out);

for( int i= 0 ; i <= nr\_announcements ; i++)

{

file1 << announcmentMessage[i] << endl;

}

file1.close();

}

void loadAnnouncments()

{

nr\_announcements = 0;

string record = "";

fstream file1;

file1.open("textFiles/announcements.txt", ios::in);

while(getline(file1, record))

{

announcmentMessage[nr\_announcements] = record;

nr\_announcements += 1;

}

nr\_announcements -=1;

file1.close();

}

void storeAvailableFlights()

{

fstream file1;

file1.open("textFiles/availableFlights.txt", ios::out); // opening file in output mode

for( int i= 0 ; i <= nr\_flights ; i++)

{

file1 << flightFrom[i] << "," << flightDestination[i] << "," << flightDate[i] << "," << flightPrice[i] << endl;

}

file1.close();

}

void loadAvailableFlights()

{

nr\_flights = 0;

string record = "";

fstream file1;

file1.open("textFiles/availableFlights.txt", ios::in); // opening file in input mode

while(getline(file1, record))

{

flightFrom[nr\_flights] = getFieldFromRecord(record, 0);

flightDestination[nr\_flights] = getFieldFromRecord(record, 1);

flightDate[nr\_flights] = getFieldFromRecord(record, 2);

flightPrice[nr\_flights] = getFieldFromRecord(record, 3);

nr\_flights += 1;

}

nr\_flights -=1;

file1.close();

}

void storeBookedFlights()

{

fstream file1;

file1.open("textFiles/BookedFlights.txt", ios::out);

for( int i= 0 ; i <= nr\_booking ; i++)

{

file1 << bookedUsersId[i] << "," << bookedUsersFlight[i] << "," << bookedUsersFood[i] << endl;

}

file1.close();

}

void loadBookedFlights()

{

nr\_booking = 0;

string record = "";

fstream file1;

file1.open("textFiles/BookedFlights.txt", ios::in);

while(getline(file1, record))

{

bookedUsersId[nr\_booking] = getFieldFromRecord(record, 0); // storing data

bookedUsersFlight[nr\_booking] = stoi(getFieldFromRecord(record, 1));

bookedUsersFood[nr\_booking] = getFieldFromRecord(record, 2);

nr\_booking += 1;

}

nr\_booking -=1;

file1.close();

}

void storeComplains()

{

fstream file1;

file1.open("textFiles/Complains.txt", ios::out); // opening text file

for( int i= 0 ; i <= nr\_complains ; i++)

{

file1 << complainId[i] << endl << complainMessages[i] << endl;

}

file1.close();

}

void loadComplains()

{

nr\_complains = 0;

string record = "";

fstream file1;

file1.open("textFiles/Complains.txt", ios::in);

while(getline(file1, record))

{

complainId[nr\_complains] = record;

getline(file1, record);

complainMessages[nr\_complains] = record; // storing data

nr\_complains += 1;

}

nr\_complains -=1;

file1.close();

}

void loadAllFiles()

{

loadIdPass();

loadAnnouncments();

loadAvailableFlights();

loadBookedFlights();

loadComplains();

}

string getFieldFromRecord(string record, int option) // getting a word from a line

{

int count = 0;

string field = "";

for (int i = 0; i < record.length(); i++)

{

if (record[i] == ',')

{

count ++;

}

else if (count == option) // if option matches with requirement

{

field = field + record[i];

}

}

return field;

}

* **Data Structures** 
  + string Username[20] ;
  + string Password[20] ;
  + string Role[20] ;
  + int nr\_users ;
  + string flightFrom[20] ;
  + string flightDestination[20];
  + string flightDate[20] ;
  + string flightPrice[20] ;
  + int nr\_flights;
  + string bookedUsersId[20] ;
  + int bookedUsersFlight[20] ;
  + string bookedUsersFood[20] ;
  + int nr\_booking ;
  + string complainId[20];
  + string complainMessages[20];
  + int nr\_complains ;
  + string announcmentMessage[20] ;
  + int nr\_announcements ;
* **Function Prototypes**
  + You have to write all your function prototypes here. This will show either your functions are fulfilling the single responsibility principle or not.
  + void printHeader();
  + void printStartupScreen();
  + void printEnterOption();
  + void printLoginScreen();
  + void printSignupScreen();
  + void printUserInterface();
  + void printAdminInterface();
  + void addFileComplain();
  + int enterIdPassword(string, string, string);
  + int checkIdPassword(string, string, string);
  + int newPasswordCheck(string);
  + void bookFlight();
  + int displayFlight(string, string, string, int);
  + void modifyFlight();      // not used anywhere as of now
  + int changePassword();
  + int changeId();
  + void exit();
  + void cancelFlight();
  + void displaySpecificUserFlight();
  + int adminAddFlight();
  + int adminSeeBookedFlight();
  + int adminRemoveFlight();
  + void adminRemoveCustomer();
  + void adminAddAccnouncement();
  + void adminRemoveAccnouncement();
  + void adminModifyAccnouncement();
  + void adminSearchFlight();
  + int subShowAllAnnouncement(string, int, string);
  + void seeAnnouncements();
  + void removeComplain();
  + void adminPrintComplain();
  + void printHelpMenu();
  + void adminEditHelpMenu();
  + void clearWithoutHeader();
  + void exitToUserInterface(string, int);
  + void exitToAdminInterface(string, int);
  + void adminchangeAppearance();
  + void printSky();
  + void printPlane(int);
  + void clearSpace();
  + void oneTimeStartup();
  + int SeachWord(string, string);
  + void changeFood();
  + void everyEnd();

//        ==================================  //

//              animation prototypes               //

* + void animprintRobo();
  + void animprintAdminInterface();
  + void animprintHowToLogo();
  + void animhowToUseFull();
  + void animprintRoboInterface();
  + void animStartupOneTimeAnim();
  + bool isAnimShown = false;

//        ==================================  //

//     VERSION 2.0 CHANGES      //

* + void storeIdPass();
  + void loadIdPass();
  + void storeAnnouncments();
  + void loadAnnouncments();
  + void storeAvailableFlights();
  + void loadAvailableFlights();
  + void storeBookedFlights();
  + void loadBookedFlights();
  + void storeComplains();
  + void loadComplains();
  + void loadAllFiles();
  + string getFieldFromRecord(string record, int option);

//     VERSION 2.1 CHANGES      //

* + void GeneralShowSingleFlight(string, string, string, string, int);
  + int subReturnCountAnnouncement();
  + int subReturnCountComplains();
  + bool isLocationAvailable(string, string);
  + string enterToLocation();
  + string enterFromLocation();
  + string enterDateCorrectly();
  + string enterfoodCorrectly();
  + void gotostartupscreen(string);
  + void changePrice();
  + string printTime();
  + void printUniqueLines();
* **Functions Working Flow**
  + Here you have to draw a diagram that will show how you are calling your functions. This will show how you have designed the flow of your code.
* **Weakness in the Business Application**
  + I think the UI can be a bit better, more representable and beautiful.
  + Can Have more graphical elements
* **Future Directions**
  + Make the application more beautiful and easy to use.
  + Using full sentence function features can be optimized a bit more.

**Formatting Instructions**

1. Heading Size is 16
2. Sub heading size is 14
3. Further heading size is 13
4. Make your heading font bold
5. Text Font size in the paragraph is 12
6. Use Times New Roman Font Style
7. Text paragraphs should be justified. (Justify is feature of MS World)
8. Your code line spacing should be 1. Also remove the spacing before and after the paragraph in your code to make it compact in the word file.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

**Student Reg. No. :**   **Student Name.**

|  |  |
| --- | --- |
| **Checked by:** |  |
| **Comments:** |  |