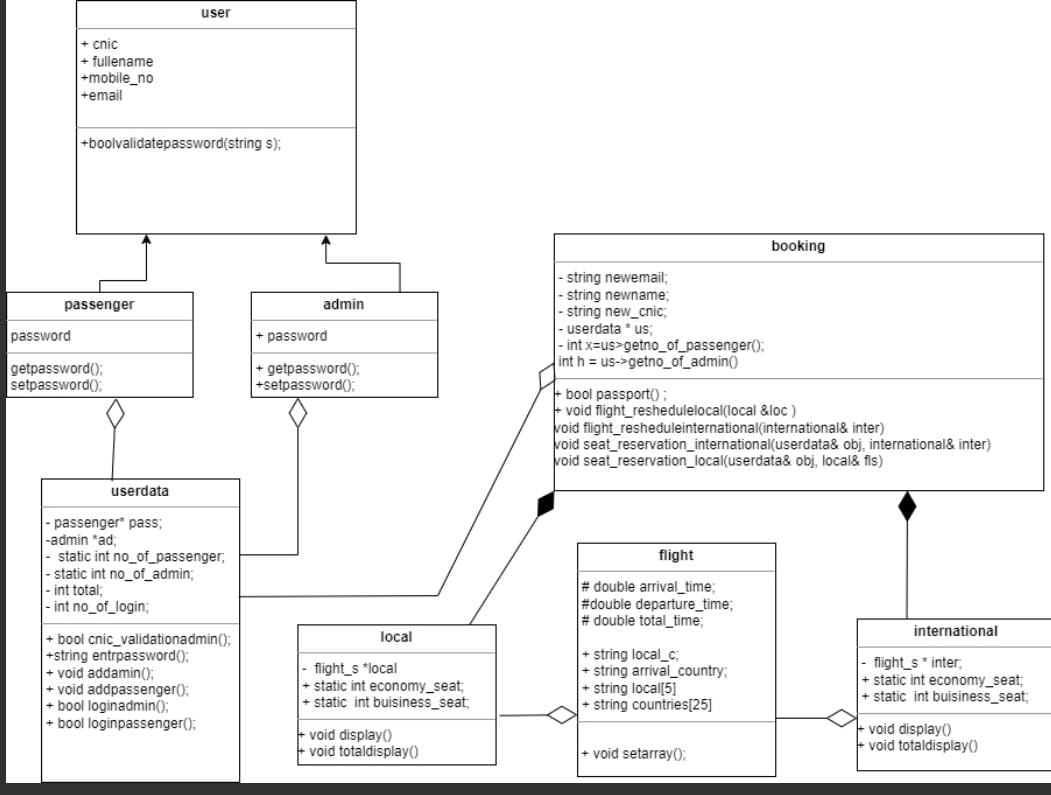
# FLIGTH MANAGEMENT SYSTEM (NAFS)

Name: shoukat khan

Roll no: 22i-8784

Section: SE(E)

# CLASS DIAGRAMME OF SYSTEM



The "Passenger Management System" Explanatory Report Model:

# Details of the login registration model:

To manage and organize passenger and administrative data in a transportation company, the "Passenger Management System" concept was created. The organization can efficiently and swiftly store and access data about passengers and administrative.

Four classes make up the model: "user," "passenger," "admin," and "userdata." The "user" class contains five variables: CNIC, complete name, cellphone number, and email. For these variables, this class also includes getter and setter functions. Both the "admin" and "passenger" classes derive from the "user" class and have a password extra variable. The password variable also has getter and setter functions in these classes. Passenger and administrative data are managed and stored by the "userdata" class.

The full breakdown of each class and its purposes is as follows:

First, there is the "User" class, which includes five private variables: CNIC, full name, mobile number, and email. There is a parameterized constructor and a default constructor for this class. Every private variable in the class has a getter and a setter. Additionally, the class contains a member method named "validatePassword," which accepts a string as input and returns true if the password satisfies the necessary requirements. The password must include a minimum of one digit, one special character, one lowercase letter, and one uppercase letter.

The "Passenger" class, which derives from the "User" class and contains a separate private variable for the password, is the second class. The class has a parameterized constructor as well as a default constructor. The password variable has getter and setter procedures in the class.

The "Admin" class has a second private variable for the password and is an inheritor of the "User" class. The class has a parameterized constructor as well as a default constructor. The password variable has getter and setter procedures in the class. The "restrictPassengers" member function of the "Admin" class limits the number of passengers who are permitted to register.

# 4- The "Userdata" Class:

This class controls and holds admin and traveler data. For the admin information, passenger information, the number of passengers registered, and the number of administrators registered, the class has a private variable. Every private variable in the class has a getter and a setter. The admin data and passenger data are initialized in the class' constructor. The class provides a function to enter a password and member functions to validate the CNIC of the passenger and admin. The class also contains two member functions: "getno\_of\_login," which returns the passenger information for the logged-in user, and "setno\_of\_login," which modifies the number of users who are logged in. The class also has a member function named "restrictPassengers," which limits the number of passengers who are permitted to register.

In summary, the "Passenger Management System" model is intended to handle and arrange administrative and passenger data within a transportation organization. The model was created in C++ using object-oriented programming techniques. The organization can efficiently and rapidly store and access admin and passenger data thanks to the model's four classes, "user," "passenger," "admin," and "userdata."

# Flight schedule module:

A flight's fundamental components, including arrival\_time, departure\_time, total\_time, local\_c=local country, and arrival\_country, are represented by the class flight\_s. Additionally, the class has two arrays called local and nations that each record the names of five local cities and twenty-five different countries.

To manipulate these characteristics, the class defines a number of methods. At a certain index in the local array, the getlocal method delivers the name of the nearby city. The international country's name at a certain index in the nations array is returned by the getinternational method. The setarray method replaces the name of the last city in the array with the name of the local city at a given index. The local\_c attribute's value is set by the setlocal method. The name of the arrival nation is returned by the getarrival method. The local city's name is returned by the getlocalcountry function. The departure\_time attribute's value is set by the setdeparture method. The departure\_time attribute's value is returned by the getdeparture\_time method. The arrival\_time attribute's value is set by the setarrivaltime method. The arrival\_time attribute's value is returned by the getarrivaltime method. The total\_time attribute's value is set by the settotaltime method. The value of the total\_time attribute is returned by the gettotaltime method.

# International flight :

An array of type flight\_s named internation is included in the class international, which is a integration of flight\_s, and is used to store international flights. Two static properties, economy\_seat and business\_seat, which represent the quantity of seats in the two classes, respectively, are also included.

The international class uses a number of techniques. The international array's index-specific flight information is returned by the getflight\_international method. The user is prompted by the display method to choose a departure airport and the flight's direction (north or south). The total\_time attribute is then set based on the chosen country, and a random index is generated for each flight's arrival\_country attribute in the international array. The total\_time is set to 20.0 if the United States is the arriving country, and to 2.5 if China is. The range for other nations is between 15.0 and 24.0.

One of the members of the international class, internation, is a pointer to an array of five flight\_s objects. A default constructor for the international class initializes internation by allocating an array of 5 flight\_s objects.

A menu allowing for the selection of the departure airport and flight direction (north or south) is displayed using the display() function of the international class. The function sets the local\_c variable of each flight\_s object in the international array to the name of the relevant city depending on the chosen departure airport. The h variable of the international class stores the flight direction (north or south).

After then, each flight\_s object in the international array has its arrival country determined by a random index between 0 and 24. The related flight\_s object's arrival\_country variable contains the arrival country information.

The total\_time variable of each flight's object is assigned to a particular value depending on the chosen arrival nation. The time is recorded as a double and expressed in hours.

Overall, the creation of an array of 5 flight\_s objects with randomly chosen arrival nations and flight times depending on the chosen departure airport and flight direction is the responsibility of the international class. The user can choose these options using a straightforward interface provided by the display() method. The data members and member functions required to store and manage flight information are provided by the flight\_s class.

# class locally:

A pointer to an array of flight\_s objects is contained in the private member fls of the class local integration is used, which is defined. Using the new operator, the constructor allocates space for 10 flight\_s structures.

Additionally, the economy\_seat and business\_seat static integer member variables have been defined. These variables can be used without first constructing a local class object.

The user is presented with a menu to choose a departure airport and a direction of travel (north or south) using the member method display() of the class local. After using srand to seed the random number generator, the function iterates through the flight\_s array, setting several properties for each flight, including the city of departure, the city of arrival, and the total duration of the flight.

The flight's departure city is set using the setarray() function of the flight\_s class, which is invoked based on the flight's index in the fls array. To set the flight direction (north or south) for each flight, the setlocal() function of the flight\_s class is used. Based on the direction of travel and a random index, the flight\_s class's getlocal() function is used to retrieve the name of the flight's departure or arrival city.

The arrival city of the flight determines the value of the totaltime property of each flight\_s object. The duration of the flight is determined by the totaltime attribute. Depending on the flight's arrival city, different values are set for the totaltime attribute. The flight\_s class's settotaltime() function is used to set the totaltime attribute.

The program seems to simulate a system for scheduling flights that creates ten trips with randomly chosen cities for departure and arrival and a random direction of travel. The user can choose the departure city and travel direction from a menu that is presented to them by the display() function. The flights with the necessary properties are then added to the fls array of flight\_s objects.

The static integer variables economy\_seat and business\_seat are both present in this class. Since they are declared static, these variables are considered to belong to the class itself and not to any individual class instance. This implies that a single copy of these variables is shared by all objects belonging to this class.

A user interface for choosing the departure airport and the direction of travel (north or south) is provided by the display function in this class. Then, using getlocal, setlocal, setarray, and other member functions of flight\_s, it sets the arrival airport, the trip time, and other details for the flight\_s objects.

It is important to note that the fls pointer is initialized with an array of 10 flight\_s objects by this class' constructor. The lack of a destructor for this class, however, can result in memory leaks.

In general, this class offers a user interface for a neighborhood airline to generate and manage flight schedules. The basic data structure to represent distinct flights is provided by the flight\_s class.

# Booking class:

A flight reservation is represented by a class booking. Its constructor sets the number of travelers and administrators in the US pointer as the beginning value for a dynamic array of userdata objects. The class's several member functions include:

Gets the passenger's email address using the string getnewemail().

string getnewname(): Gives the passenger's name.

getnew\_cnic() returns the passenger's CNIC (Computerized National Identity Card) number.

bool passport(): Inquires about the traveler's foreign passport and any visa stamps. It returns true if the response is "y" or "Y," and false otherwise.

Allows a passenger to change the time of a local flight. void flight\_reshedulelocal(local &loc). It invites the user to choose a flight ID to postpone after displaying the schedule for nearby flights. The user is then prompted to provide a new departure time for the chosen flight, and the flight schedule is updated as a result.

A passenger may reschedule an international flight using the void function flight\_resheduleinternational(international& inter). It invites the user to choose a flight ID to reschedule after displaying the international flight schedule. The user is then prompted to provide a new departure time for the chosen flight, and the flight schedule is updated as a result.

Allows a passenger to reserve a seat on an international flight with the help of the void function seat\_reservation\_international(userdata& obj, international& inter). It invites the user to choose a flight ID while displaying the international flight schedule. The user is next prompted to choose a seat number and a seat category (economy or business). If the desired seat is available, it is reserved for the traveler, and the flight schedule is updated to reflect the new seat status. It prompts the user to choose an other seat if the initially chosen one is not available.

# Main :

This C++ application implements a reservation system for airlines. It specifies a primary function that presents a menu of choices and carries out different operations in accordance with the user's selection. The application opens with a welcome message before presenting a menu of choices.

The main function specifies a number of variables, including objects for storing flight data and an object for storing booking information, as well as a boolean variable to monitor whether a person is registered in as a passenger or an admin.

The application manages user data, flight data, and booking data using a number of user-defined classes. The userdata class, for instance, includes operations for adding a new admin or passenger user and for logging in as either sort of user. Information about available flights, including their timetables, routes, and availability, may be found in the domestic and international flight classes. The booking class oversees the booking process, which includes arranging flights and reserving tickets.

Depending on the user's selection, the main function employs a switch statement to carry out various operations. For instance, if the user chooses option 1, the application asks them whether they want to register as a new passenger or an administrator before calling the relevant function from the userdata class. If the user chooses option 2, the software asks whether they want to log in as a passenger or an administrator before calling the relevant function from the userdata class to complete the login procedure.

If the user chooses option 3, the software asks whether they want to book a seat on a domestic or international aircraft. The program calls the seat\_reservation\_local or seat\_reservation\_international function from the booking class to handle the booking procedure if the user is logged in as a passenger. A notification asking the user to log in is displayed by the application if the user is not currently registered in as a passenger.

If the user chooses option 4, the software asks if they want to reschedule a domestic or international flight before calling the relevant function from the booking class to conduct the rescheduling procedure. The application will display a notice asking the user to log in if they are not currently logged in as an administrator.

If the user chooses option 5, the application asks if they want to see the schedule for domestic or international flights and then calls the relevant function from the domestic or international flight class to display the schedule.

Currently, option 6 is user flight cancellation.

Option 7 provides details on the airline and its offerings.

Option 8 and Option 9 are not currently in use.

When option 10 is chosen by the user, the software ends.

# Bonus task:

Dynamic arrays are used

Parameterized constructors are used

Programe does not stop execution

File handling for input data in file

Click here : [Class diagramme link:](https://app.diagrams.net/#G1V1S_DgTRHkFdiFDJ2j4mXgGGmh_dBi_v)

https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=\_blank&layers=1&nav=1&title=Untitled%20Diagram.drawio#R7VxZk9o4EP41VM2mipRv4HGYI9nd2VQ2mc3xRAksjBLZIrYYIL9%2BJUu%2BhAHDGBsyeZlBstRqtbpbn9Rtd8wbf%2FUmBPPZP8SFuGNo7qpj3nYMQ7c0jf3jNWtR09P6osILkSsbZRUf0U8oK2U%2Fb4FcGBUaUkIwRfNi5YQEAZzQQh0IQ7IsNpsSXBx1Djy4UfFxAvBm7Wfk0pmo7Ru9rP4tRN4sGVl3BuKJD5LGcibRDLhkmasy7zrmTUgIFb%2F81Q3EXHiJXD7%2Fuf6MH747b%2F76N%2FoB%2Fhv%2B%2FfjuU1cQuz%2BkSzqFEAb0aNJve4%2BPQdejy1nw5W4a9N9pFumagvQTwAspr47hYDbIcM5nTNdSjM6PBZ%2Fm0Aehh4KOea3NV2mxS8mcVVlxFYUr2gUYebzVhDEMQ1abUWC%2FPPk%2FHmecVNxgEEVJLZvGWG3J6uZq3SzkbCZKl4yh54e7V%2FscMjMMpzSdWtkkhlwnEeRKec1%2BPq7n8Oz59SGdEffqj30sG4WBjScYUsQM61os7m286kO51Ldi5CFhraY4NpMpYvpnDqckoNIt6IYs3wMfYe5Q3kL8BDlVLhvqY96I%2FVzOEIUf52DCey2ZY0r52WsE0lg4s3CVcwHSKN5AwqYfrlkT%2BXQgzVv6N%2BbgRHmZeQvdkXWznKdI%2BgHpoLyUcmaE7Ie0w7hIxt%2B4izM0DMYQt2xvc2ZuMPBY8%2Fpt7mQKzJlektA9V%2F48SBMWuX3tlGONw0aVhjX27QZWzlDk7vHb%2Fmu0f1aULuCALVrvbXhi6DKII4skZM7cIwHAd1ntMCSLwIWcrMYXJW3zQPiixTL%2BBildy4UBC0qKK8CEHK6%2F8P6vDTspf80%2FvF1J6qK0TkorREU%2FW5a%2B5p5knXgh6XPY0kZkEU7gPjVmMJOZDqQ72tmiHZfmTkUJIQYUPRUBZdmyx12vwxCscw3mBAU0ylF%2Bzysy%2FdN7igI6Co5T2qfIurw9%2ByE42DKaU947ZVdIV%2FZSNDmVSZlyV9Tmfsva3B8YBXV%2BrWnmPpXmpfcwRGzyfHdV9Lw36OU1PSG4Vdt5QaVWswXoTqsmsKF1pupkVa0TjJ5M6%2BzzOeYsostCXOLIMAnQpCEsI89UC4xhAHx%2BNAE%2B3%2F%2BDcTRvDlExLnwyRhiOAtLckNAHCDc2nNTM8AJUcEwIZtaLXEBhinQjGqLA4%2BPsxry%2FPobVbcW%2FbmJYwyrBsIZ2OIit5nF75%2BNxpzie8EX5XJPfuJLFmFGLr0ERE%2BSIMsjQpAcUJKM5CEqplvKdMu0yOw3pIoQlbAuS%2B9eg9dnkV4ESWroGLU6mmutuW4rDuKV01Zgw7zuanI0Ij59JYpUTdjiJ0WnrM9oUdMce2h37th0OhGAQjBgXxj422nW3Fc1JAaJ8tk%2BEbYOME8hOMOzUXbh9O0i3tqjOy8Avdr%2BIX6y%2BvXkJ12sUwOj6%2BSAYfmZkyBdcFobZYwRjMPnuxTc73QnBJBTRKBQgigAuJdmNdU9GLV7Jwhn43fonClwfcWKvgHv%2B8ytdKNVXsn6A8jsETVwMBWREpqMsBnWZsyyflVi9C52RmEgMdi97CmItMPGOXYv2b934rYe8fBvJ2w9Egli9mowuMmZSUMfvwtuIcMYCKToVCb2A64LmBbKdmdSntcyRVJ7YANpQmV38tCSjajD%2FZUBu3VAgt1kS9y6F3Grspj7I7ZwP5JYI7JLw9lAi4nPOjxFctpQlw%2Fexyokyv74HMLWiBzhl5ktFBzAocQBqtkDgXvO0YFZyEfBJ4D7OYoTHHtzHgpcx%2ByQX2bCKIi4mE2RhfNN2OsUwfhLWrx7Gz%2BUZ2P1%2BR8kzcDpH5BnUnRmQ5GDvywywKmYGlCnLM7MFBk5BL82KKSqbmTKavZvQlqyD7Rk8declJPrdpMI%2FN7FLL1iJYe80kra0N4ESbaivqSCro%2FXX0oqGYBn2M%2FV3C8OWAgUHtqLsuYyy2lT%2FjF49ELGSCwJ7ZRdsIsw9ivi1oTqhJqI%2BhYswOCEB8dejCIJ9cjwZKxsSEqyNFyhCAYwilbk6wjXnqS7lfHGRyGsLF0VzHEevSs7I2%2B464jvCfM%2FSPK3N0%2FbvsFhlhK44ZtuuGBZLD%2Fe1Q3TDOtBtj9lRC4b5yMcTCK%2B63Xy9OIlpqW6z31yVDU1L%2F8YNCg6eX4we4OMRbxqIC9Qjff0z5paxV4MPOFKkmiJSIeRjthfhS9WoTUXPfnLuT7MhNcz2oZvX7y3n5WwLfUPdFipe3ermybaFM8r3HBPyPQ5WXRSez%2FKmArgUeeCN4WZ1%2BDjvvilg3HZSX11zVIU4it%2BZePEiyXKXYtywuICEnVOLJLt8WUTpHhY%2F9SBVEmPUKMUvLC9tyzSFuGa8ASfBVKi7RWpJmPukp3POqYxl81Wak5DyIbX8VlQBykg8HcJoBt0FhvGVzZW8uNHSTuIvq2aVJVBpC7HCYedKPfrkKSdQfjttDjQ5ZRg%2BxSRGReK53ESFLhl%2F6xg3yQD1jS8kVWHc5BZMeT7FUSnobAgDvwyMaqvvGZe8kmQZJRjVON3VRb%2F5YEsaXRwMCq%2FDvzZFePDI6GInF7eRL9rviNy0FYRJDgWtBGGUN%2BLMnqJWVYMwfeV1e1NXCNUUhLEUg7HlWe20QZh6Au76ISZRqsTVvyrRdwY5Q%2BpqSVy9zdftk1jWWVuErXxnIn3n4lCL6Jl7CLUeVk8YbFKtUwU1WvvuidFizoaTfBYw%2BUygPTjS3zpmgZBltahdEx3q3Q%2F%2BNYNUH34an96t7h8euvUo1zEwoph60Ws09aJFz6VuwY6agFZZt5SLU8d8bkJF3bpV9tJZY%2FlAmq53CsDS2bst7%2Foyjs4I9gsa61iHg94TbdWt7MAb6nekHquvSzqm0Zges2L2yVPRPPtwrHn3Pw%3D%3D