FLIGHT BOOKING SYSTEM REPORT

Ziad Essam

Design And Architecture OF The System

1. Introduction

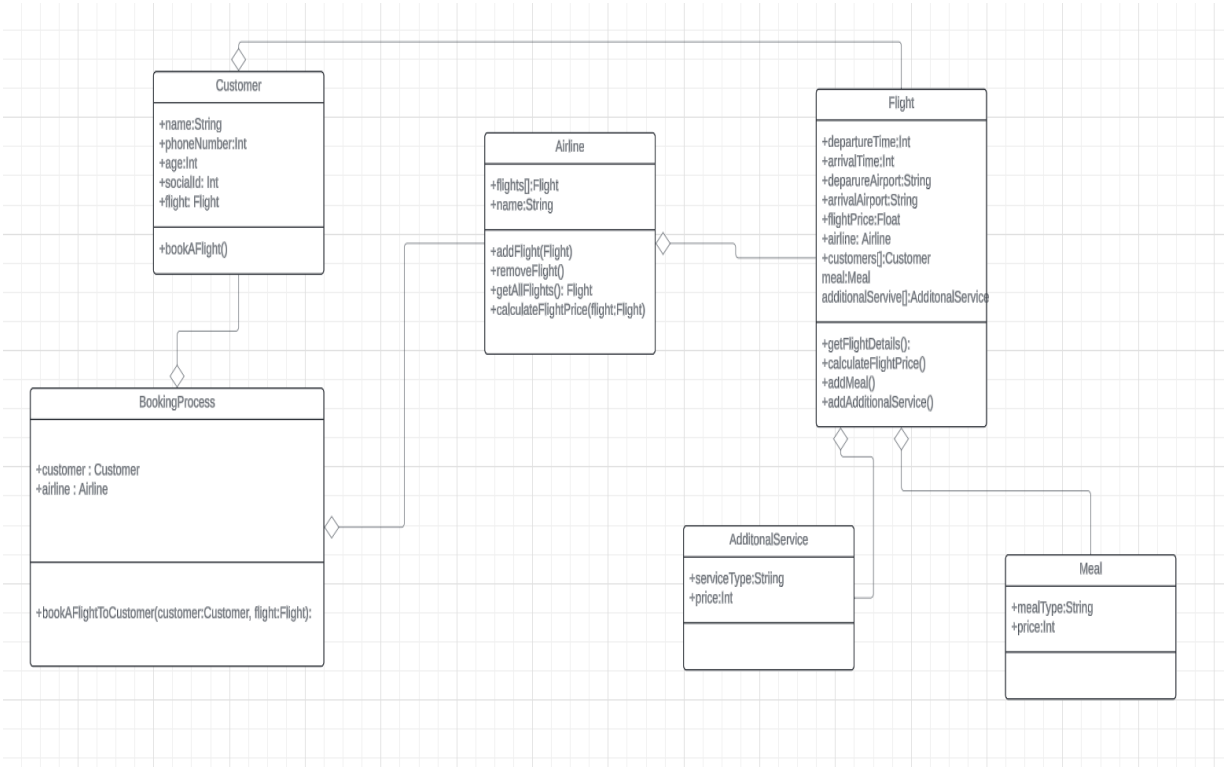
I created a simple flight booking system that allows users to search for flights based on the airports of departure and arrival, select the best flight, enter their credentials, and select other flight options like meals or extra services like Wi-Fi or extra luggage.

The system consists of 6 essential classes, that I tried with every class to apply the 5 solid principles on them, I make sure that every class has one responsibility. The system was designed to be flexible and easy to maintain, the classes interact with each other to achieve the overall functionality, such as searching for flights, booking a flight, and handling user credentials.

The system also allows users to select additional options like meal preferences and extra services. These features make the booking process more personalized and user-friendly. By adhering to the SOLID principles, the system is more modular, which means that changes or extensions can be made to individual parts without affecting the whole system.

2. Design Explanation

UML Diagram



The diagram below represents the overall architecture of the flight booking system. The system is designed following the SOLID principles to ensure that each class has a clear, single responsibility and that the code is modular, maintainable, and scalable.

- **Flight Class**:

  This class represents a flight. It stores details like flight number, departure and arrival times, airports, and price. By keeping flight-related data in one place, the system can easily manage and display flight information, also it has methods like *addMeal*, *addAdditionalService* and *calculateFlightPrice* that calculate the flight price based on its actual price and any add ons.

  The *Flight* class is associated with the *BookingProcess* class, which uses flight information to complete the booking. Also the class has an aggregation relationship with two classes *Meal* and *AdditionalService*

- **Airline Class:**

  This class manages the airline's data, including the flights it offers and any related taxes. It handles the addition of flights and ensures that all airline operations are organized. It has methods like addFlight and *calculateFlightPrice* that takes the flight that its price should be calculated

  The *Airline* class has an aggregation realationship with the Flight class, as it manages a collection of flights,

- **Customer Class:**

  The *Customer* class represents the users of the system. It stores and manages customer details, such as name, phone number, age, and social ID.

  The *Customer* class has an aggregation relationship with the *Flight* class, as every customer has a flight.

- **BookingProcess Class:**

  The *BookingProcess* class manages the entire booking process, coordinating between *Customer*, *Flight*, and *Airline* to complete a booking. It ensures that all steps necessary for booking a flight are carried out in sequence, from verifying flight availability to updating the flight and customer records**.**

  The *BookingProcess* class has an aggregation relationship with *Customer* class and *Airline* class.
  It is dependent on the Flight class to verify availability and the Customer class to register the booking.

- **Meal Class:**

  The *Meal* class represents different meal options that a user can select during the flight booking process. It encapsulates the details of each meal option, such as size (small, medium, large) and the corresponding price.

  The *meal* class is a part of the *flight* class, it can be added to the flight instance.

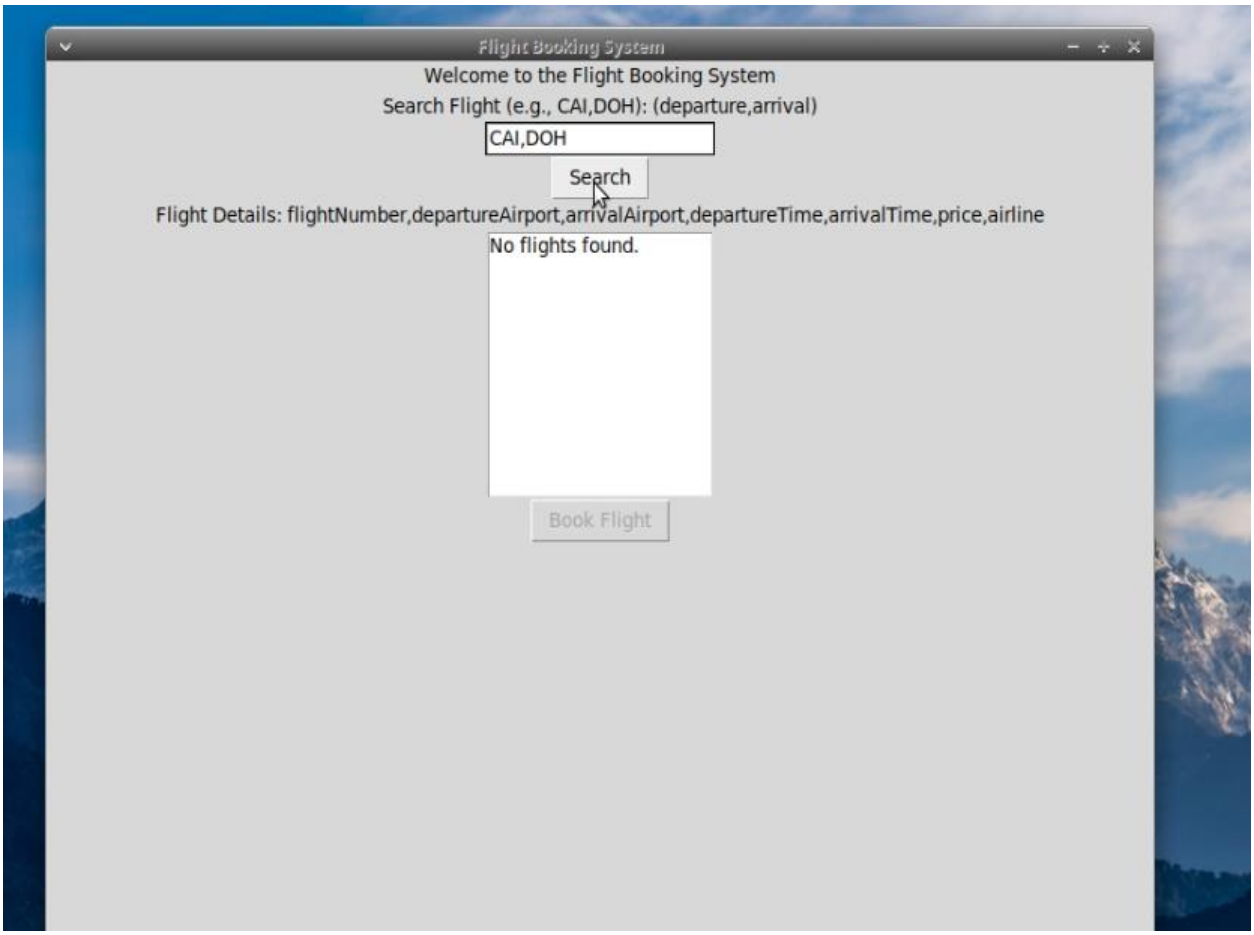
- **AdditionalService Class:**

  The AdditionalService class represents extra services that users can choose, such as Wi-Fi or extra luggage. It holds information about each service, including its availability and cost.

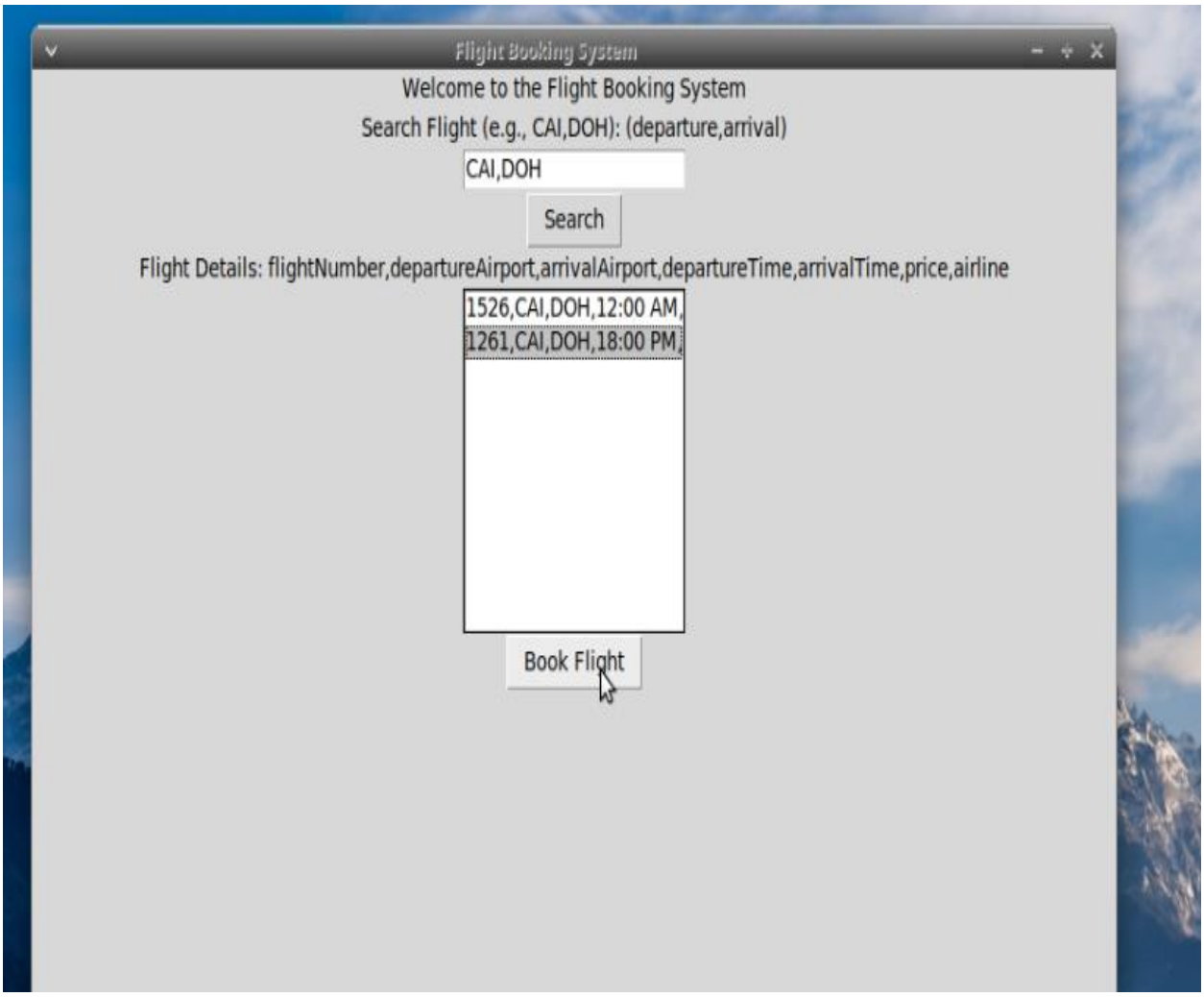
  Like the *Meal* class, *AdditionalService* class is used by the Flight class, so there is an aggregation relationship between them.

Screenshots of using the application:

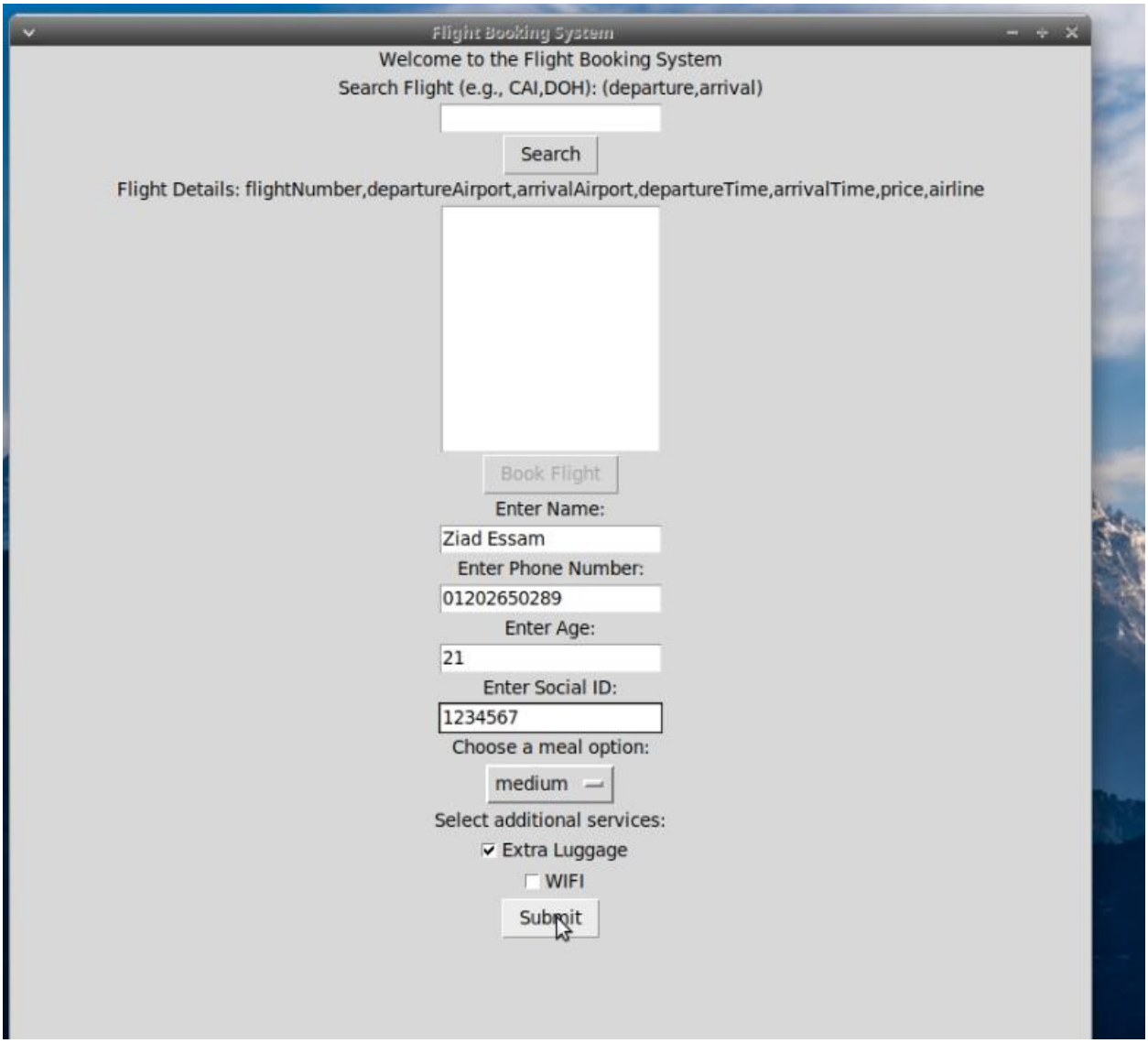
1. The user enter the departure location and the arrival location in the label in this format ‘deaparture,arrival’ , and click search



2. Then the available flights appear. The user should pick the flight that suits him the most and then click book the flight.



3. Then the user proceeds by filling the required details, also there is option it the user want any meal or additional service. Then the user should click submit to get the ticket.



4. After that the user completed the booking of the flight and his ticket appear.