Read Me
Project Name: Flight Booking System
Team Member: Xi Tu, Zhenxiang Guan (Both From Section B)

Description: Our project is to make a system which can help people book flight tickets by creating the account and fulfill their requirements. In this system, people can buy all the tickets to the destination within several airlines, thus, they can use the system to compare the tickets. Moreover, when people search for the airline, it will give you all the information about a specific airline . Therefore, by using our system, it can make travel much easier.

Link to the UML class diagram:
https://lucid.app/lucidchart/invitations/accept/inv_a2376778-9a06-478d-b5f2-604a3dd20898?viewport_loc=-328%2C-99%2C2420%2C1616%2C0_0

Passenger

Passengers

- firstName: String
- lastName: String
- username: String
- password: String
- email: String
- dateOfBirth: Date
- phone: int
- passportNumber: String

Description: In the Passenger model, we need the passengers' first name, last name, date of birth, and check the passport number to check the validation of the user, then the system can use the information user provided to help them book the ticket and hotels. By entering the username and password which means they can create an account in our system so that it will be much more convenient for them in later use. Finally, the system will send the confirmation to the user's email or text, therefore, they can easily approach their booking information.

Domain

- Flight

Description: In the Flight domain, it will have the **type** of the plane, and the **capacity** which limits the passengers in one plane. Moreover, the **endurance** determines the distance that one plane can travel

● Tickets

Description: In the Ticket domain, it contains all information that people need to know to get on a plane. It has the **date** when the plane takes off. The detail of the flight also includes on the ticket such as **from**, **destination**, **seat, and class** which are essential for passengers to know before they board.

● Airlines

Description: In the Airline domain, it includes the **name** of the airline, the **phone** which can be used to contact the stuff in one airline to get help. Also, it provides the **country** of the headquarters of airlines.

Portable enumeration

Flight Class is our portable enumeration. This means that within our ticket table the flight class must always be a given value within the flight class table, since there is a constraint. Flight class has a fixed set of flight, and there should always be a flight class associated with a ticket. The values we chose for genre are: ECONOMY, FIRST, BUSINESS

User to domain object relationship

Passenger -> Flight: one to many. Passengers can book different flights based on their needs on the ticket such as the travel routes, time, date, etc.

Domain to domain object relationship

Flight -> Ticket : one to many. There could be lots of tickets for one plane, but the number of the tickets cannot exceed the capacity of the plane.

Airline -> Flight: one to many. One airline can have several planes, but it can be various types of planes, and each plane must belong to one airline.

Description of the user interface requirements

- ***Passenger List*** - displays a list of all passengers, and navigate to flights for that passenger
- ***Passenger Editor*** - displays a particular user for editing or allows creating a new passenger
- ***Flight List*** - displays a list of all flights and navigate to the tickets for that flight
- ***Flight Editor*** - displays a particular flight for editing or allows creating a new flight
- ***Ticket List*** - displays a list of all the tickets
- ***Ticket Editor*** - displays a particular ticket for editing or allows creating a new ticket