# Airline Booking system

# “Travelista”

| NAME | ID |
|---|---|
| بسام عماد حمدي | 20210222 |
| جميل محمد جميل | 20210252 |
| ايه محمد علي | 20210210 |
| ايمان ايهاب ابراهيم | 20210201 |
| بثينة عصام محمد | 20210218 |
| تريفينا رضا امين | 20210236 |

## About The Project

### Project Overview:

**“Travelista”** is an airline booking system, which is a web application designed to facilitate the booking and management of flights for customers. The system allows users to search for flights based on their preferred criteria, such as destination and departure airport. Once a flight is selected, users can book tickets and see their tickets. The web application tries to give the customer the best experience possible.

### Key Features:

**Flight Search**: Users can search for flights based on criteria such as destination, departure airport

**Booking Management**: Customers can view their bookings and tickets

**Flight Information**: Display detailed information about flights, including departure and arrival times, aircraft model, and flight duration.

**User Authentication**: Secure login and registration functionality for customers.

**Admin Panel**: An admin panel to manage flights, airports, aircraft, and customer bookings.

### Technologies Used:

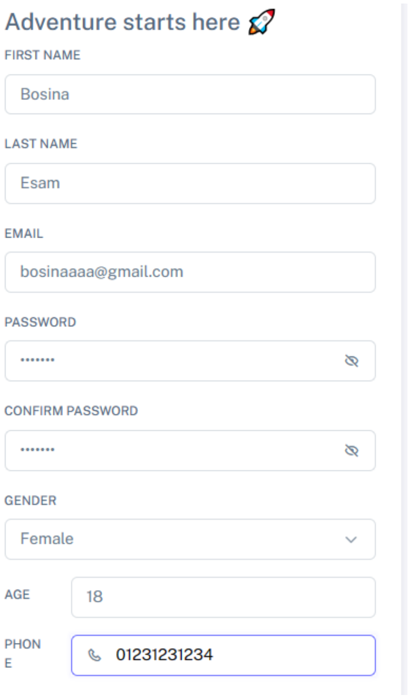
**Frontend**: HTML, CSS, JavaScript, Bootstrap

**Backend**: java spring boot

**Database**: PostgreSQL

**Other**: AJAX, jQuery

# Customer interface



Adventure starts here 🚀

FIRST NAME: Bosina

LAST NAME: Esam

EMAIL: bosinaaaa@gmail.com

PASSWORD: •••••••

CONFIRM PASSWORD: •••••••

GENDER: Female

AGE: 18

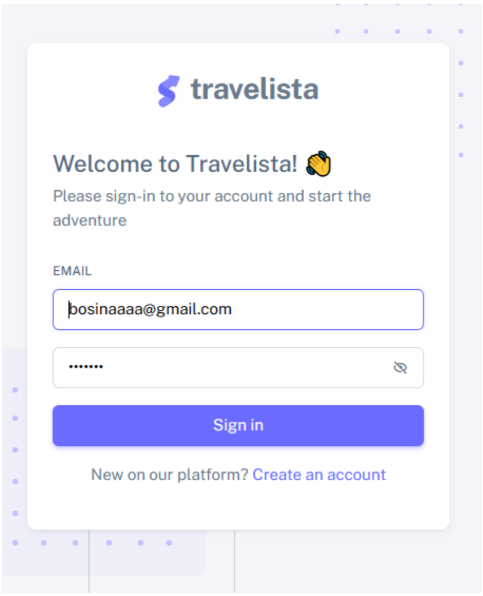
PHONE: ☎ 01231231234

## first: sign up

the user must create an account to access all the functions of the website. To sign up, they need to enter their first and last name, a unique email that is not already in the database, a password, gender, an age between 18 and 102, a phone number, and the role, which will always be set to 'user' because we have added an admin account hard-coded to restrict access to the admin panel.

## Second: log in

After the user signs up, the website will display the login page. The user can then generate a token by entering their email and password. Once the token is generated, it will guide them to the customer interface.



travelista

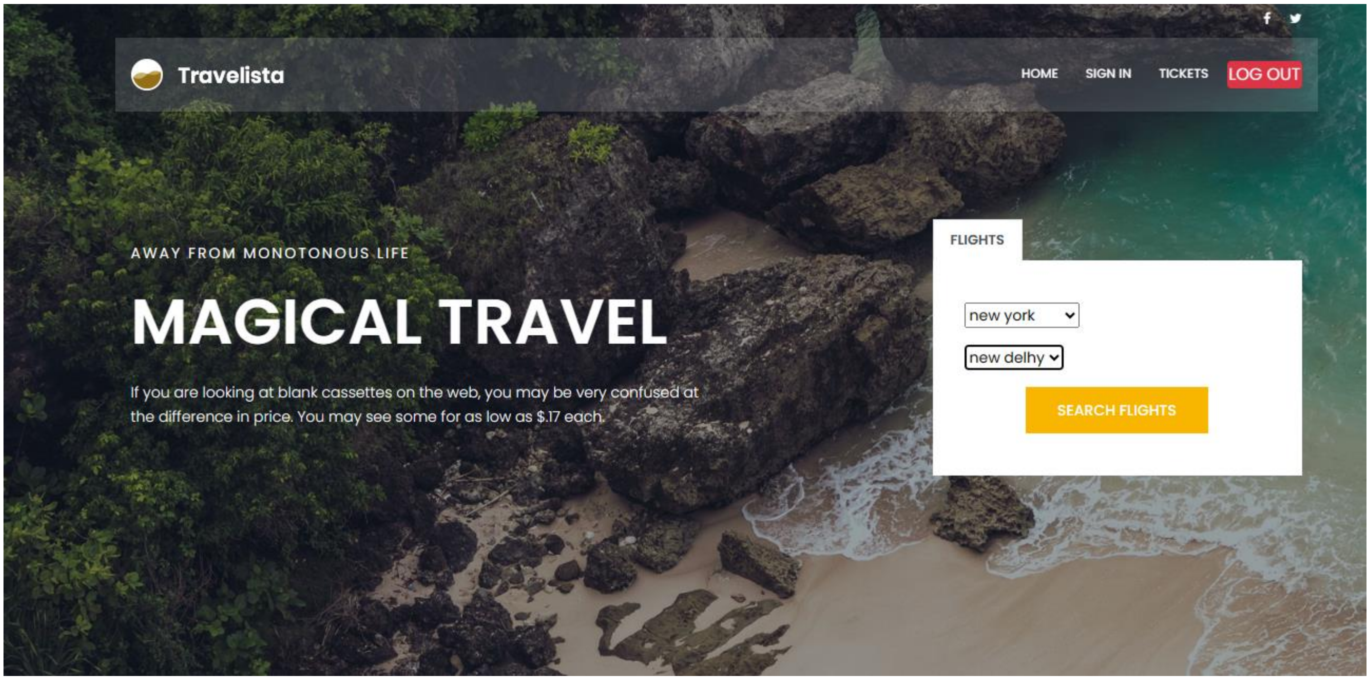
Welcome to Travelista! 👋

Please sign-in to your account and start the adventure

EMAIL: bosinaaaa@gmail.com

•••••••

New on our platform? Create an account



AWAY FROM MONOTONOUS LIFE

MAGICAL TRAVEL

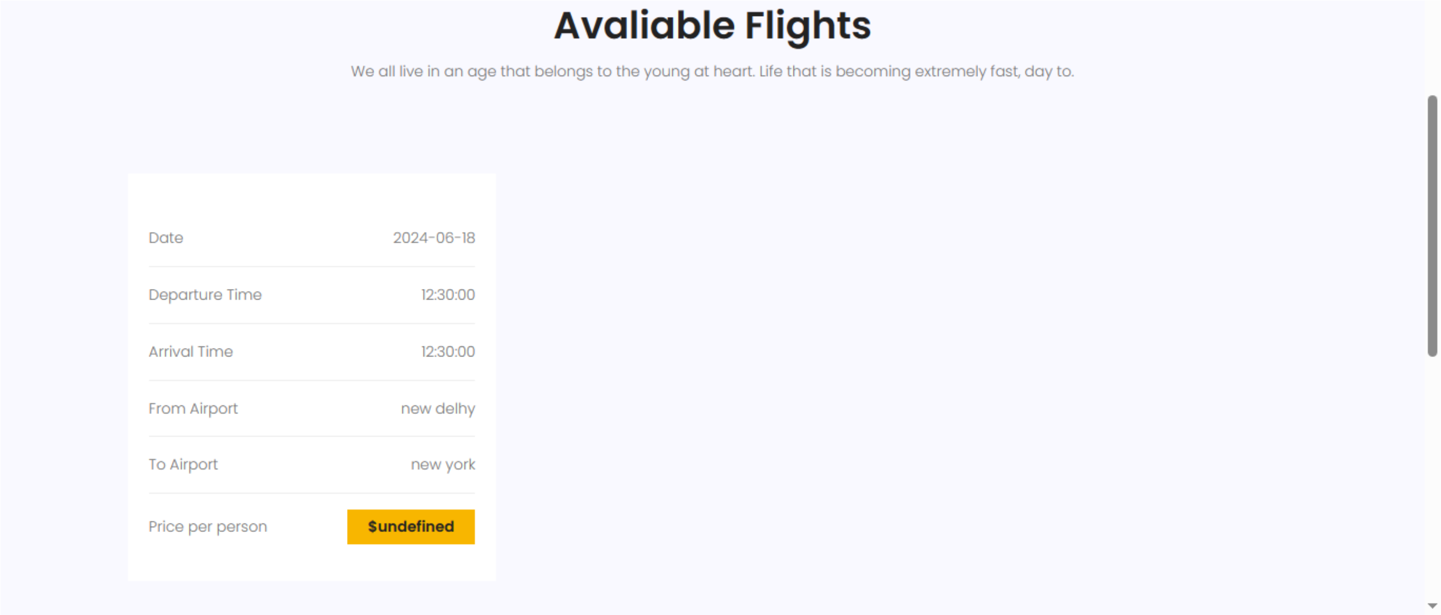
If you are looking at blank cassettes on the web, you may be very confused at the difference in price. You may see some for as low as $.17 each.

new york

new delhy

## third: search for the flight

After the user logs in, they can search for the flight they want by selecting the departure and destination airports. For example, the user can search for a flight from New York to New Delhi.



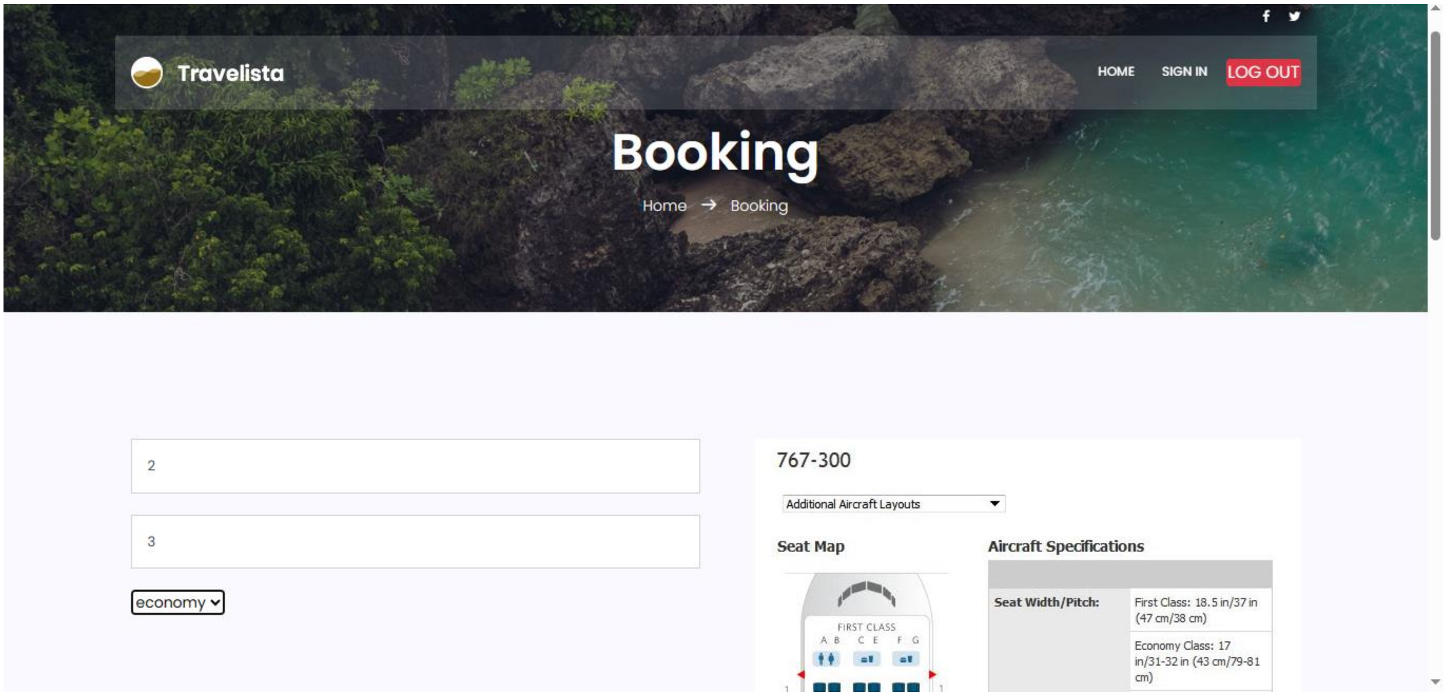
Avaliable Flights

We all live in an age that belongs to the young at heart. Life that is becoming extremely fast, day to.

| | |
|---|---|
| Date | 2024-06-18 |
| Departure Time | 12:30:00 |
| Arrival Time | 12:30:00 |
| From Airport | new delhy |
| To Airport | new york |
| Price per person | $undefined |

## fourth: available flights

On this page, the user sees the available flights based on the departure and destination airports they entered earlier. It displays all the details the user needs to know if this flight is perfect for them or not. Then, the user clicks on the flight they want and goes to the booking page.



2

3

economy

767-300

Additional Aircraft Layouts

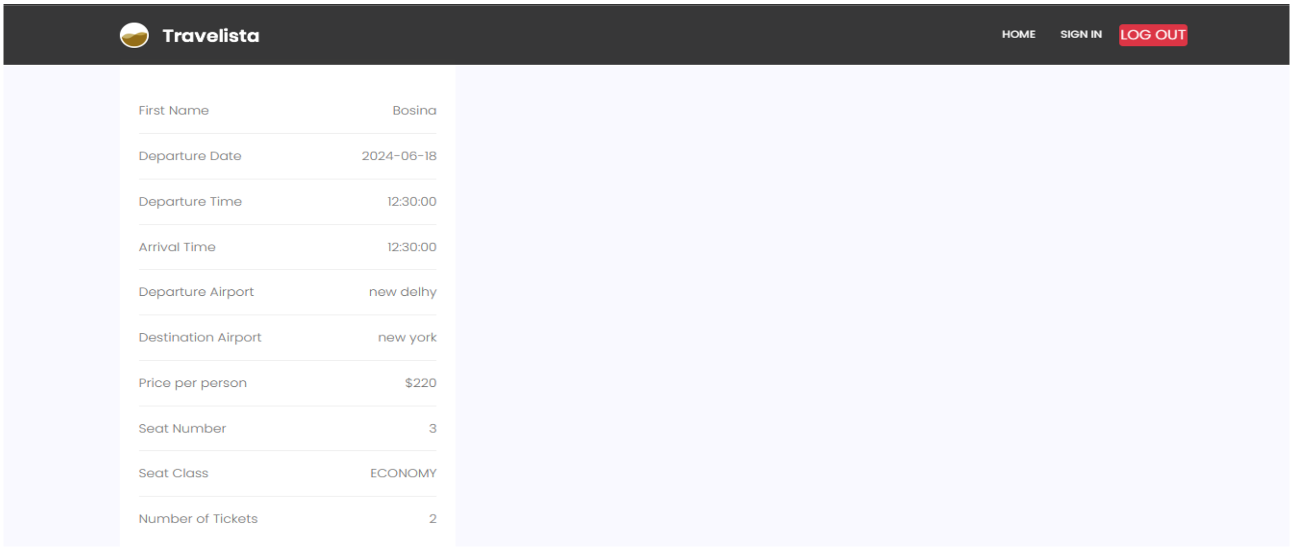
Seat Map

Aircraft Specifications

| | |
|---|---|
| Seat Width/Pitch: | First Class: 18.5 in/37 in (47 cm/38 cm)<br>Economy Class: 17 in/31-32 in (43 cm/79-81 cm) |

## fifth: book the flight

After the user clicks on the flight they want, they choose how many tickets they want for this flight and the seat number for the first ticket. Then, the system will allocate the other seats behind it for the other tickets. The user also selects the class (economy, business, or first class) and then clicks on "Get the tickets".

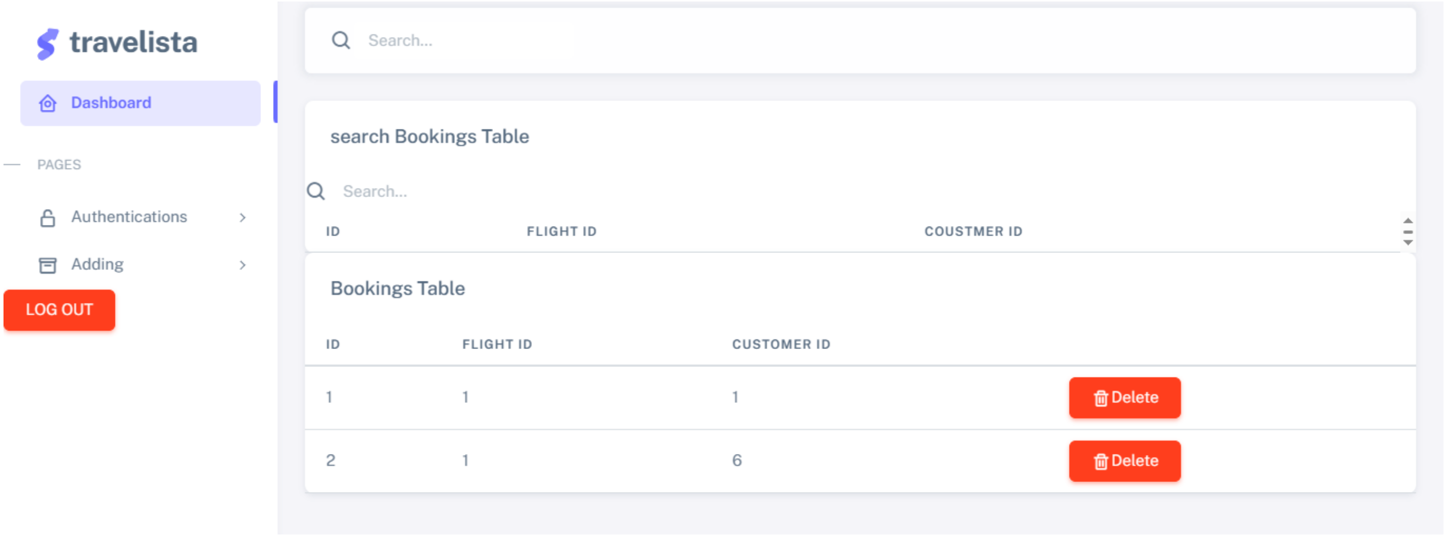


| | |
|---|---|
| First Name | Bosina |
| Departure Date | 2024-06-18 |
| Departure Time | 12:30:00 |
| Arrival Time | 12:30:00 |
| Departure Airport | new delhy |
| Destination Airport | new york |
| Price per person | $220 |
| Seat Number | 3 |
| Seat Class | ECONOMY |
| Number of Tickets | 2 |

## Sixth: get the tickets

This page shows the user all the tickets they have booked for any flight, along with all the details the user need.

# Admin interface



search Bookings Table

Search...

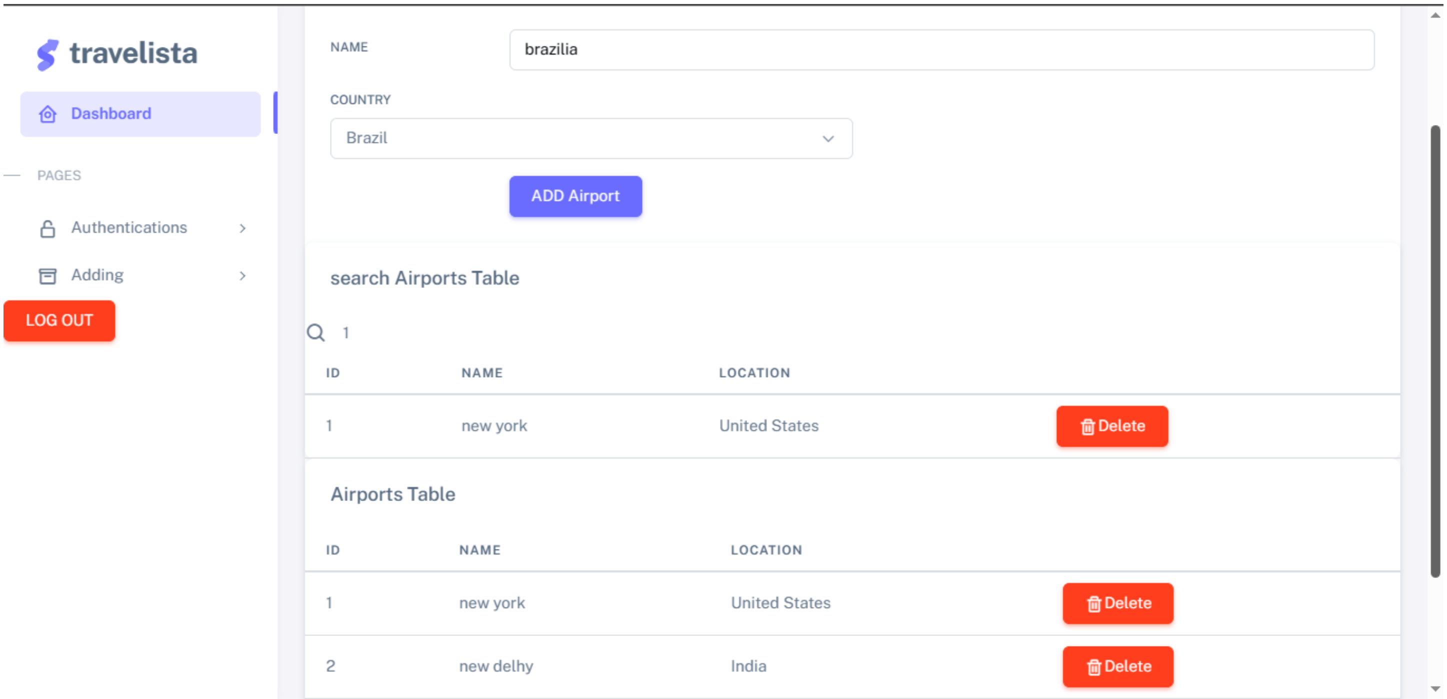
| ID | FLIGHT ID | COUSTMER ID |
|---|---|---|

Bookings Table

| ID | FLIGHT ID | CUSTOMER ID | |
|---|---|---|---|
| 1 | 1 | 1 | Delete |
| 2 | 1 | 6 | Delete |

## First: the booking table

After the admin log in to his account the first thing, he will see the booking table that show all the booking for all the customers and he can search by booking id and delete or cancel any booking



NAME: brazilia

COUNTRY: Brazil

search Airports Table

1

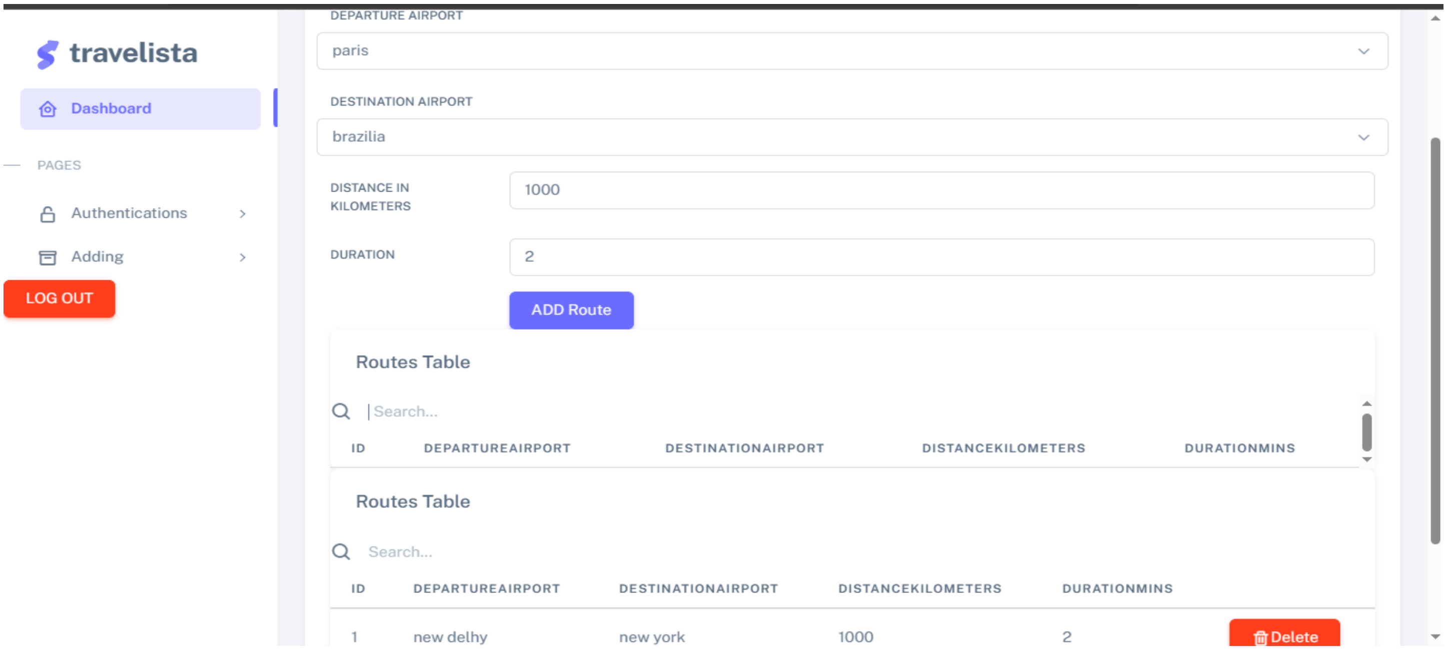
| ID | NAME | LOCATION | |
|---|---|---|---|
| 1 | new york | United States | Delete |

Airports Table

| ID | NAME | LOCATION | |
|---|---|---|---|
| 1 | new york | United States | Delete |
| 2 | new delhy | India | Delete |

## second :add airport

the first thing you do if you have an empty data base you will add airport by entering the name of the airport and the location (country) and the system will generate the id and the admin also can delete and search any airport



DEPARTURE AIRPORT: paris

DESTINATION AIRPORT: brazilia

DISTANCE IN KILOMETERS: 1000

DURATION: 2

Routes Table

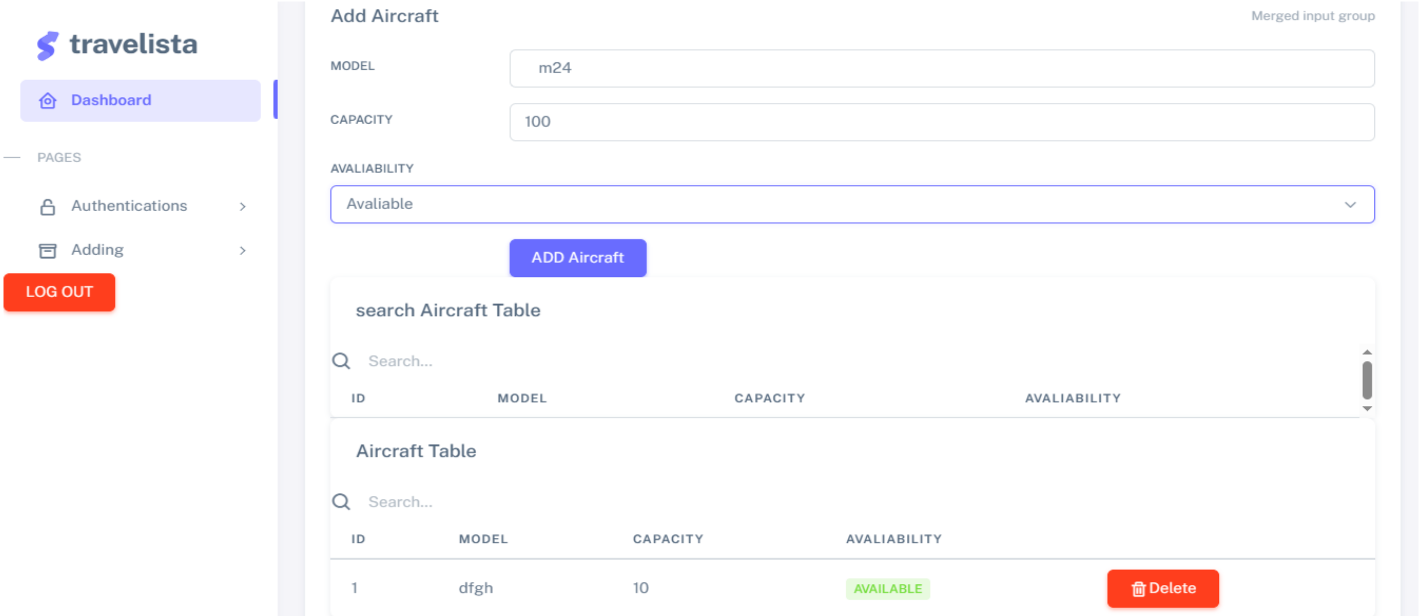
| ID | DEPARTUREAIRPORT | DESTINATIONAIRPORT | DISTANCEKILOMETERS | DURATIONMINS |
|---|---|---|---|---|

Routes Table

| ID | DEPARTUREAIRPORT | DESTINATIONAIRPORT | DISTANCEKILOMETERS | DURATIONMINS | |
|---|---|---|---|---|---|
| 1 | new delhy | new york | 1000 | 2 | Delete |

## Third: add route

After we add the airport, we go to add route where the admin can view all the routs and delete any route and search for the route by the id and definitely add a new route by selecting the departure and destination airport and the duration and the distance and that's it the route added



Add Aircraft

MODEL: m24

CAPACITY: 100

AVALIABILITY: Avaliable

search Aircraft Table

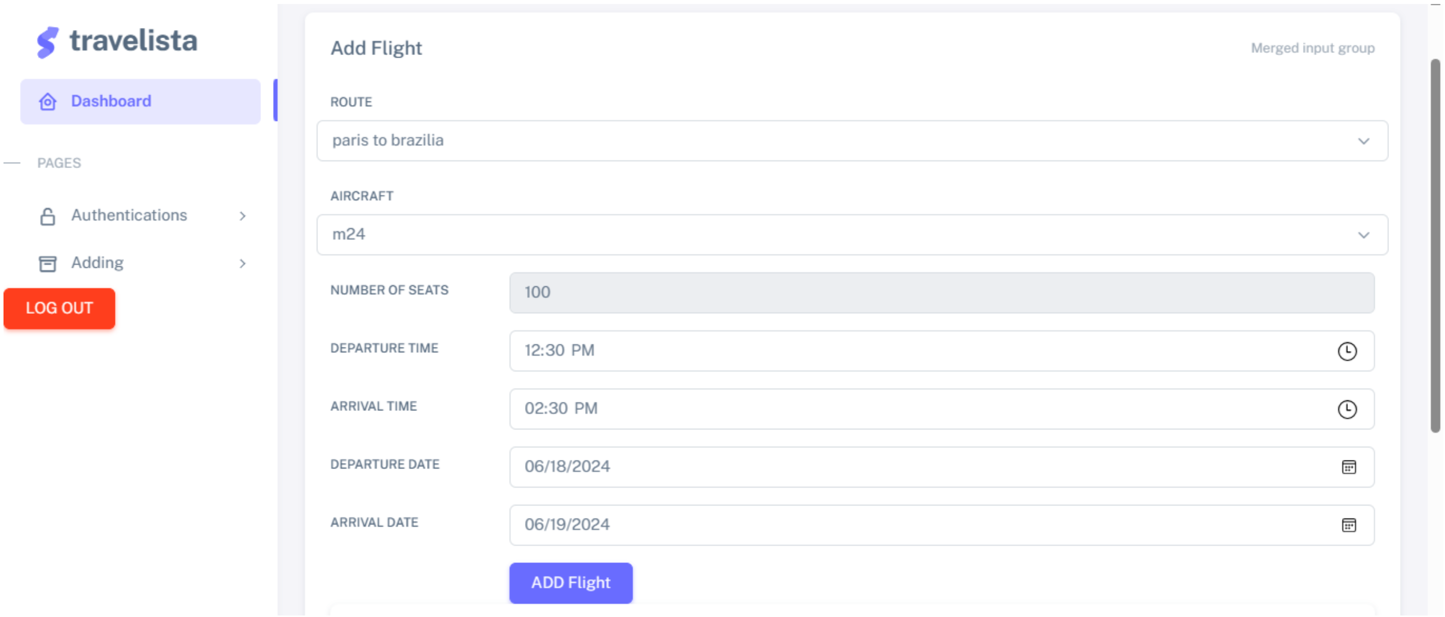
| ID | MODEL | CAPACITY | AVALIABILITY |
|---|---|---|---|

Aircraft Table

| ID | MODEL | CAPACITY | AVALIABILITY | |
|---|---|---|---|---|
| 1 | dfgh | 10 | AVAILABLE | Delete |

## fourth: add Aircraft

now in the aircraft page we can view all the aircrafts and if it's available or not and search by the id of the aircraft and delete any aircraft the admin wants and adding a new aircraft by entering the model's name and the availability and the capacity  and immediately the system will generate seats for this aircraft according to the capacity and the percent is 60% economy and 20% business and 20% first-class



Add Flight

ROUTE: paris to brazilia

AIRCRAFT: m24

NUMBER OF SEATS: 100

DEPARTURE TIME: 12:30 PM

ARRIVAL TIME: 02:30 PM

DEPARTURE DATE: 06/18/2024

ARRIVAL DATE: 06/19/2024

## Fifth: add Flight

And last but not least the flights the admin can view all the flights with all the details and also search by flight id and delete any flight and add flight by selecting the route from the one we added earlier and select the aircraft we added earlier and departure and arrival date and time and the price and that’s how to make a flight from the beginning

## Functional Requirements

1. The application should allow the user to sign up and enter his personal data
2. The application should allow the user to log in to his account
3. The application should allow the user to log out
4. The application should not allow the customer to open the admin Pannel
5. The application should allow the customer to search the flights by departure and destination airport
6. The application should allow the customer to book a ticket for the flight he wants
7. The application should allow the customer to choose the seat number and class he wants
8. The application should allow the customer choose any number of tickets
9. The application should allow the customer to view all his booking
10. The application should allow the admin to view all the booking and search and delete
11. The application should allow the admin to view and delete and search and add airports
12. The application should allow the admin to view and delete and search and add routes
13. The application should allow the admin to view, delete, search, add aircrafts
14. The application should allow the admin to view, delete, search, add flights

## Nonfunctional Requirements

### Look and feed

1. The application shall use calm colors
2. The application shall user some smooth animation

### Usability and humanity

3. The application shall be easy for all ages 16+
4. The application shall be easy and user friendly for most of people without training

### Performance

5. The application shall be responsive and load quickly
6. The application shall be reliable and available all the time
7. The application shall appear the available flight after searching in 5 second
8. The application shall save any data the customer or the admin added in 5 seconds
9. The application shall handle up to 5 users simultaneously

### Maintainability and support

10. The application shall be easy to maintain
11. The application shall be working with all the operating systems and browsers
12. The application shall be compatible with all devices like tablets and smartphones and desktop computers

### Culture

13. The language used shall be polite, simple, and formal

14. The application shall not display any racist symbols or words

### Legal

15. The application shall respect every country policy in traveling

### Security

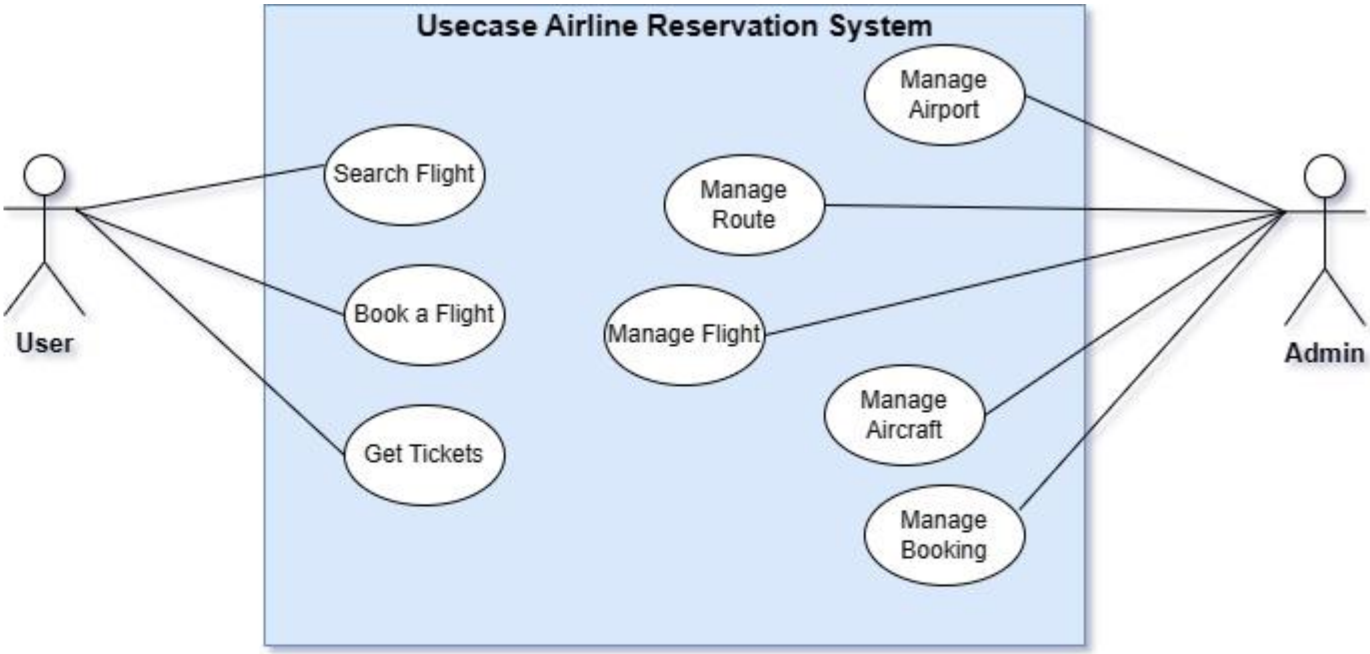
16. The application shall not allow any one to access or edit the data but the user himself and the admin

### Portability

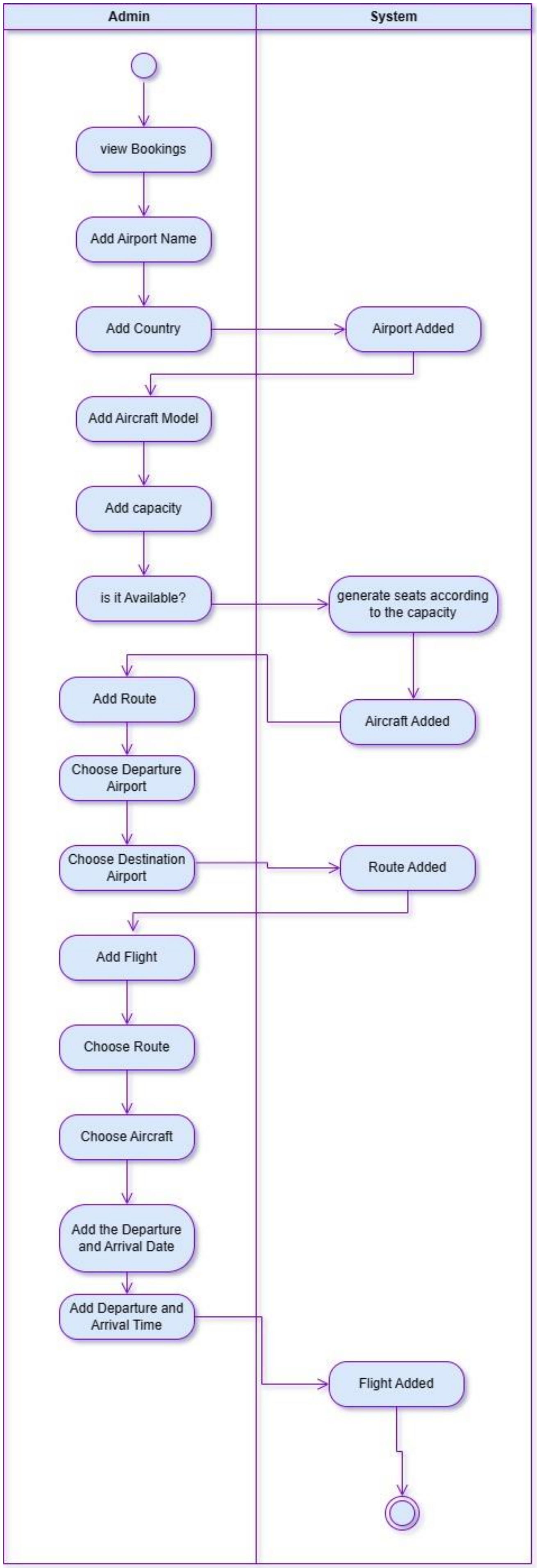
17. The application shall restart after falling in any operation in 1 minute

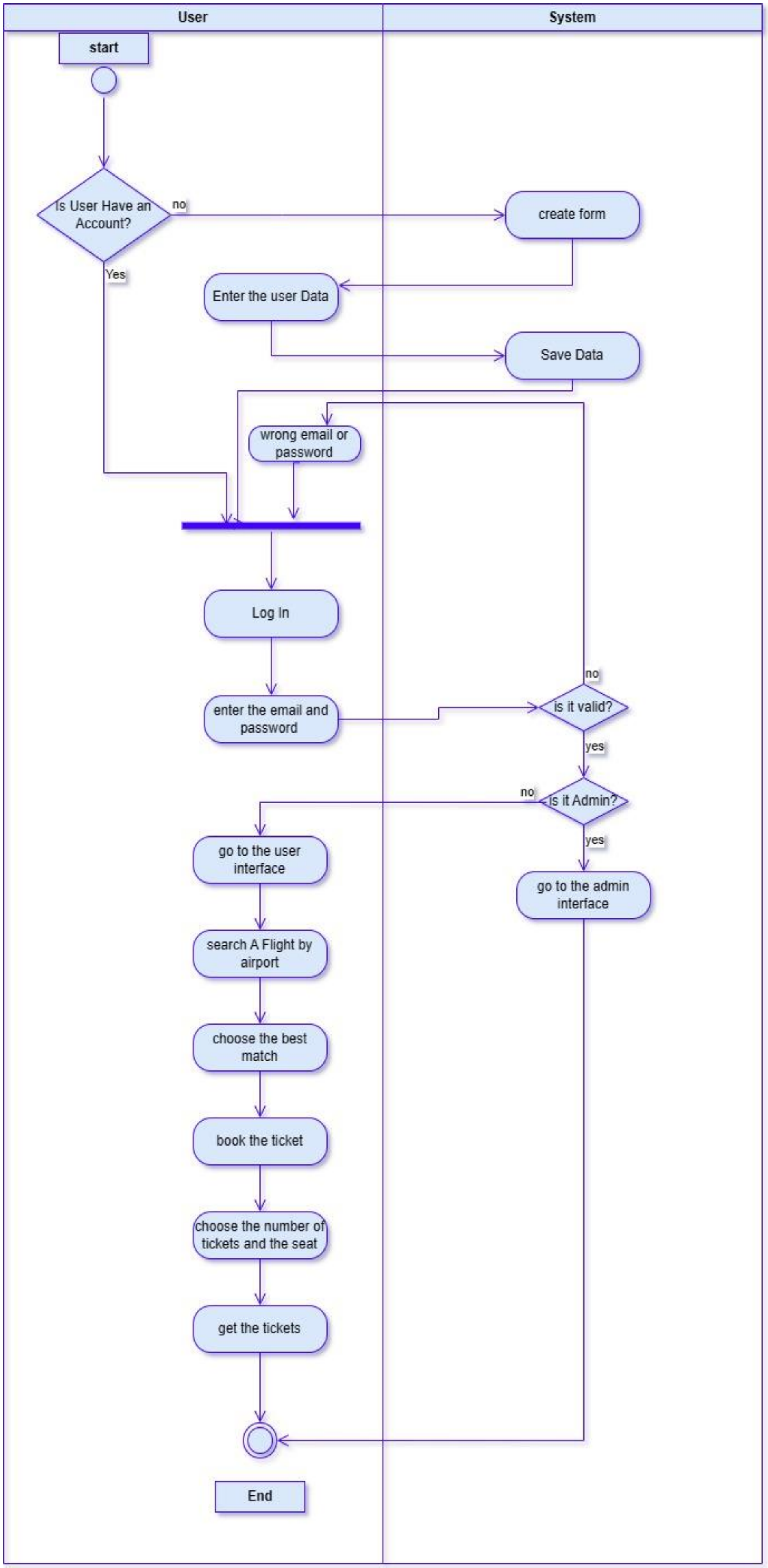
# SRS

## Use case

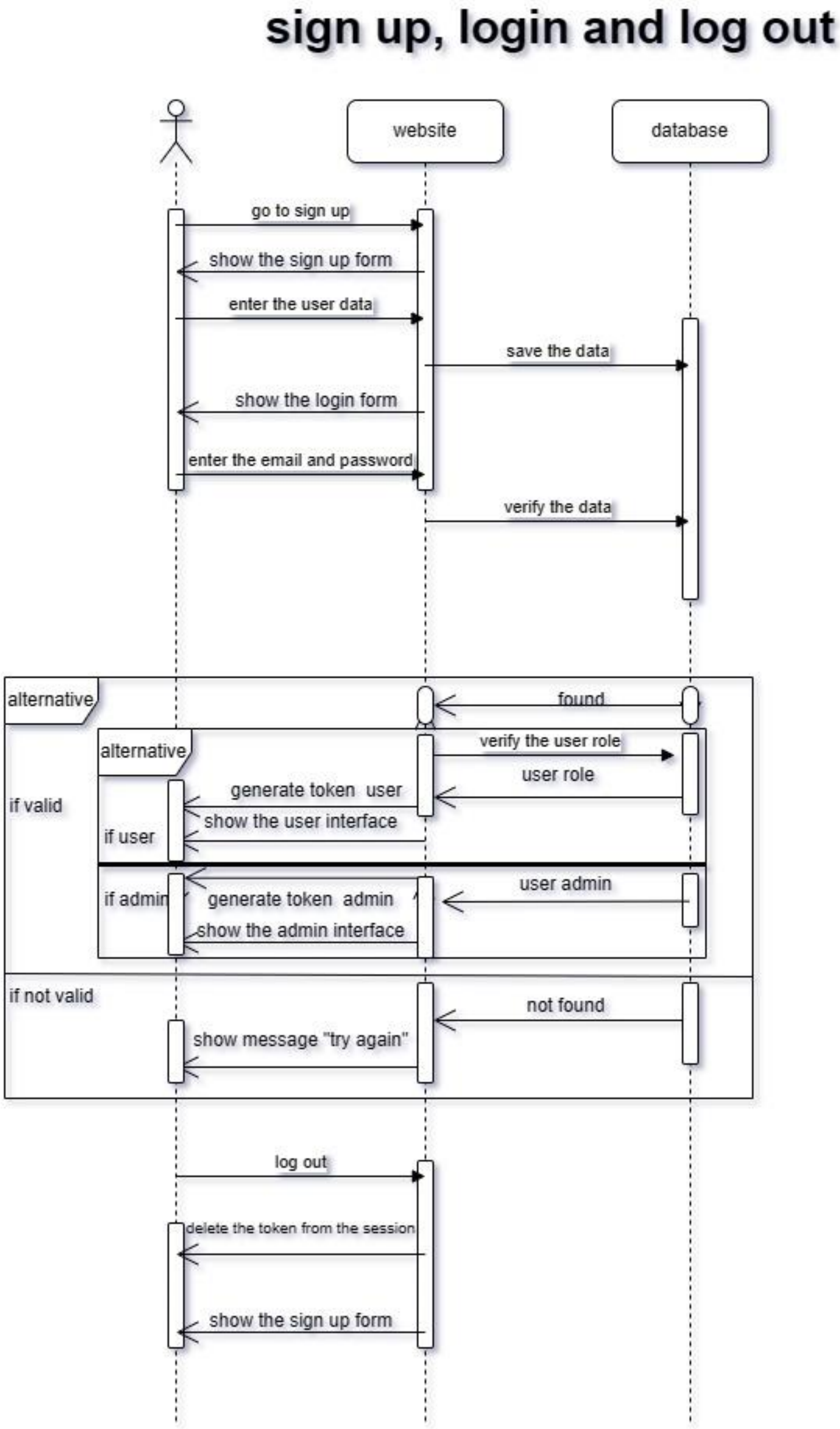


## Activity Diagram

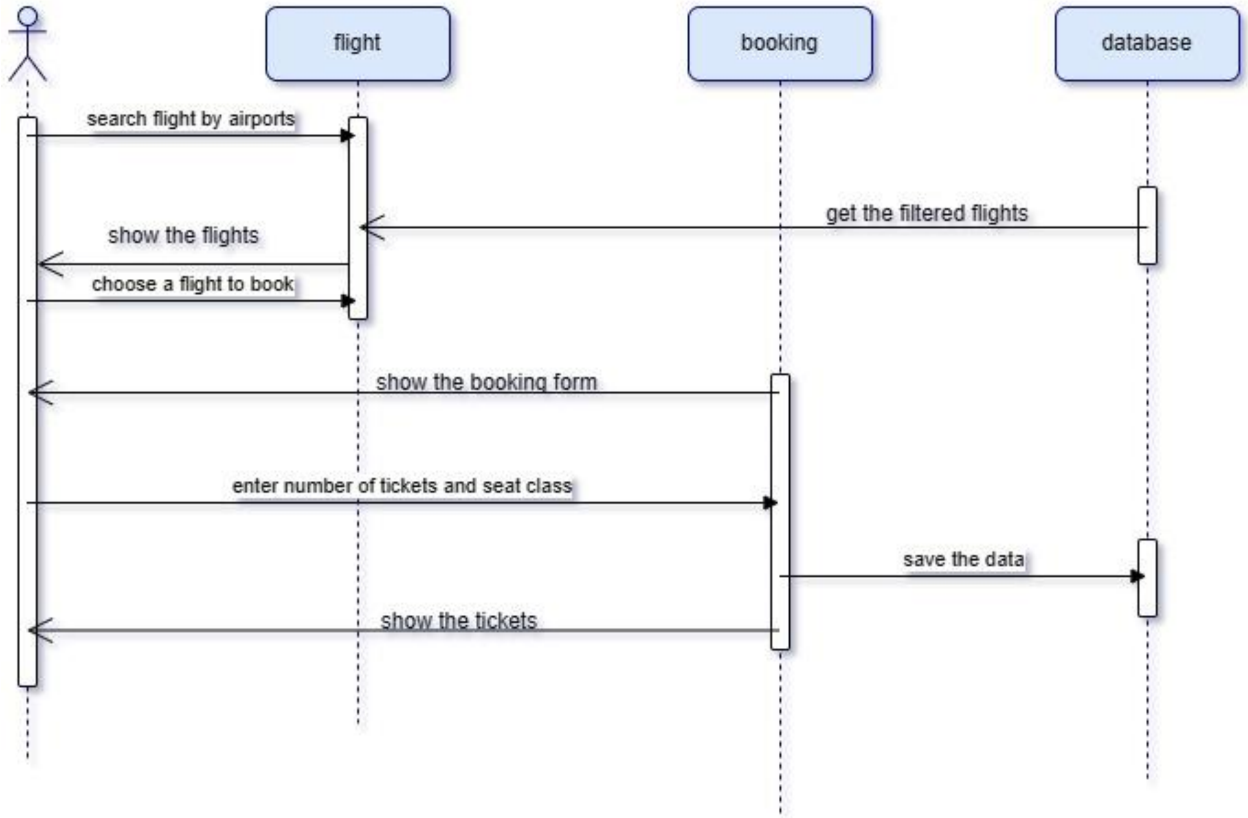

Admin
System
view Bookings
Add Airport Name
Add Country
Airport Added
Add Aircraft Model
Add capacity
is it Available?
generate seats according to the capacity
Aircraft Added
Add Route
Choose Departure Airport
Choose Destination Airport
Route Added
Add Flight
Choose Route
Choose Aircraft
Add the Departure and Arrival Date
Add Departure and Arrival Time
Flight Added


User
System
start
Is User Have an Account?
no
Yes
create form
Enter the user Data
Save Data
wrong email or password
Log In
enter the email and password
no
is it valid?
yes
no
is it Admin?
yes
go to the user interface
go to the admin interface
search A Flight by airport
choose the best match
book the ticket
choose the number of tickets and the seat
get the tickets
End

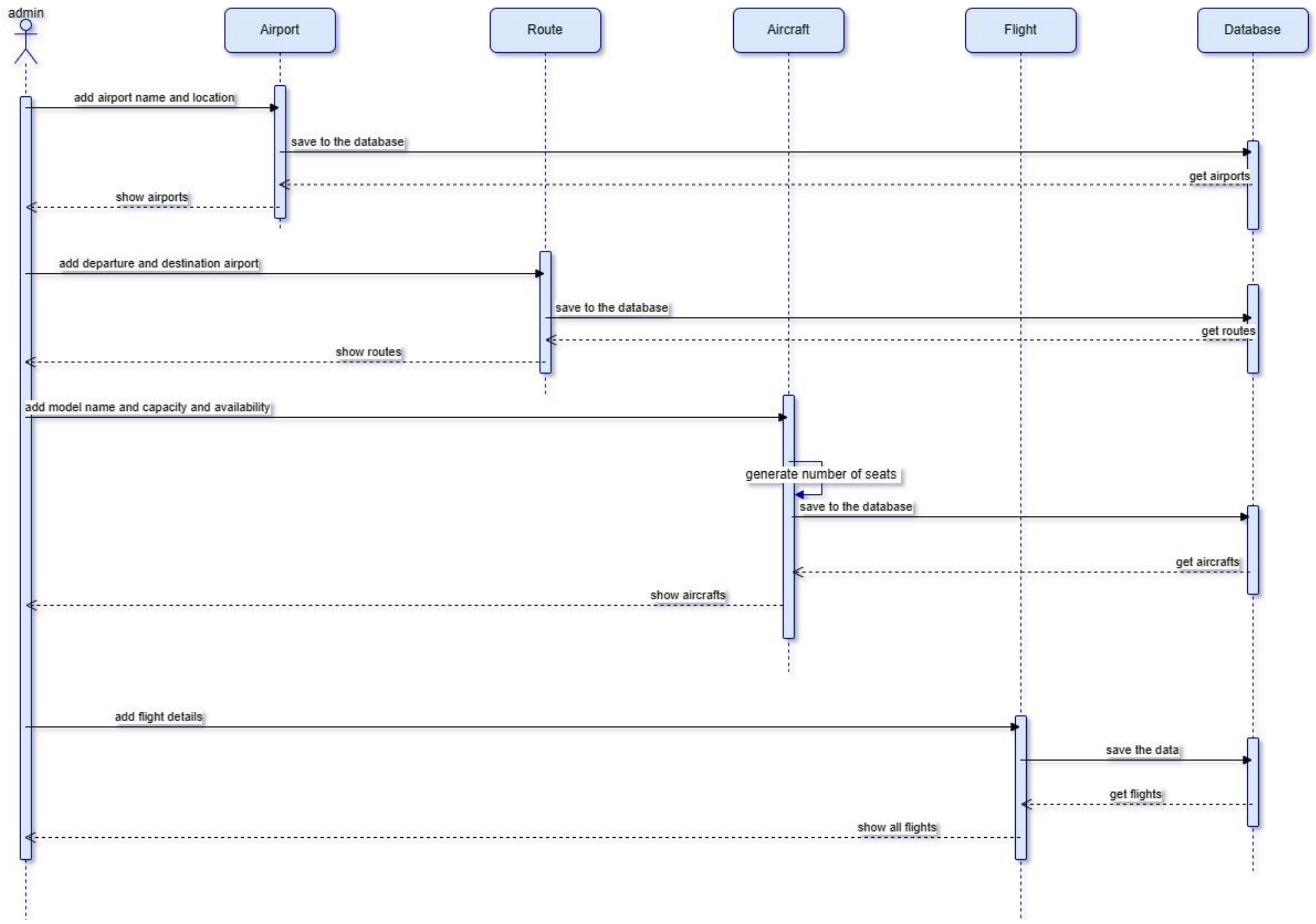
## Sequence Diagrams



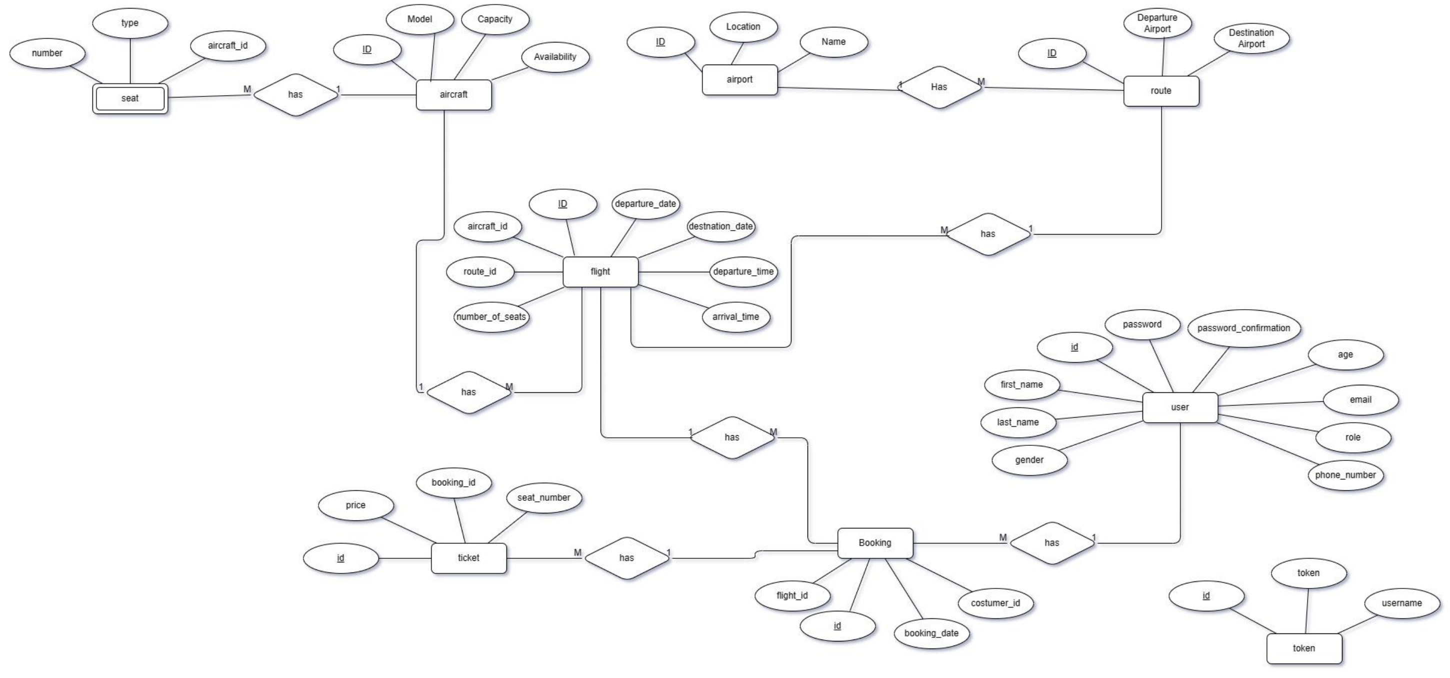
# Booking a Flight

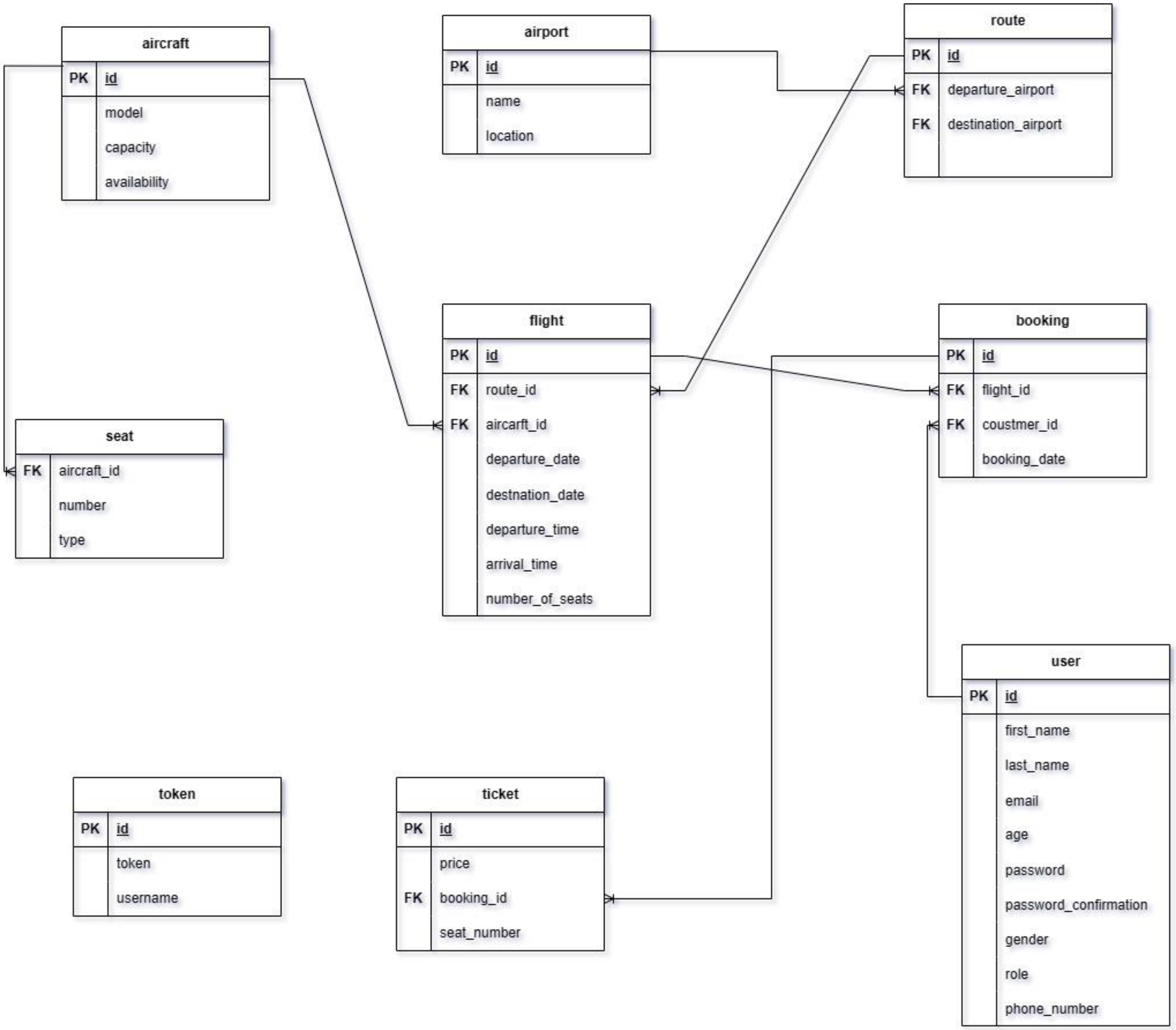


# add flight by admin



# ERD




aircraft
PK id
model
capacity
availability
airport
PK id
name
location
route
PK id
FK departure_airport
FK destination_airport
seat
FK aircraft_id
number
type
flight
PK id
FK route_id
FK aircarft_id
departure_date
destnation_date
departure_time
arrival_time
number_of_seats
booking
PK id
FK flight_id
FK coustmer_id
booking_date
user
PK id
first_name
last_name
email
age
password
password_confirmation
gender
role
phone_number
token
PK id
token
username
ticket
PK id
price
FK booking_id
seat_number

## OCL

- context User

**inv**:

-- First name must not be null or empty

FirstNameRequired: self.firstName <> null and not self.firstName.isEmpty()

-- Last name must not be null or empty

LastNameRequired: self.lastName <> null and not self.lastName.isEmpty()

-- Email must match the valid format

ValidEmail: self.email.matches('[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}')

-- Password must be at least 6 characters long

ValidPassword: self.password.size() >= 6

-- Password confirmation must match password

PasswordConfirmationMatches: self.password = self.passwordConfirmation

-- Age must be between 0 and 110

ValidAge: self.age >= 0 and self.age <= 110

-- Phone number must be at least 6 characters long

ValidPhoneNumber: self.phoneNumber.size() >= 6

- context Airport

**inv**:

-- Airport Id must not be blank

IdNotBlank: not self.Id.oclIsUndefined() and self.Id <> ''

-- Airport name must not be blank

NameNotBlank: not self.name.oclIsUndefined() and self.name <> ''

-- Airport name must be unique among all instances

NameUnique: Airport.allInstances()->select(a | a <> self)->forAll(a | a.name <> self.name)

**pre**:

-- The user must be an admin to manage Airports

User.currentUser.role = Role.ADMIN

- context Route

**inv**:

-- Route must have a departure airport

DepartureAirportRequired: self.departureAirport <> null

-- Route must have a destination airport

DestinationAirportRequired: self.destinationAirport <> null

-- Route's departure and destination airports must be different

DifferentAirports: self.departureAirport <> self.destinationAirport

**pre**:

-- The user must be an admin to manage routes

User.currentUser.role = Role.ADMIN

- context Flight

**inv**:

-- Flight must have a route

RouteNotNull: not self.route.oclIsUndefined() and self.route <> null

-- Flight must have an aircraft

AircraftNotNull: not self.aircraft.oclIsUndefined() and self.aircraft <> null

**pre**:

-- The user must be an admin to manage flights

User.currentUser.role = Role.ADMIN

- Context Aircraft

**pre**:

-- The user must be an admin to manage aircrafts

User.currentUser.role = Role.ADMIN

**inv**:

-- The total number of seats should be equal to the capacity

let totalSeats: Integer = self.numberOfSeats->size() in

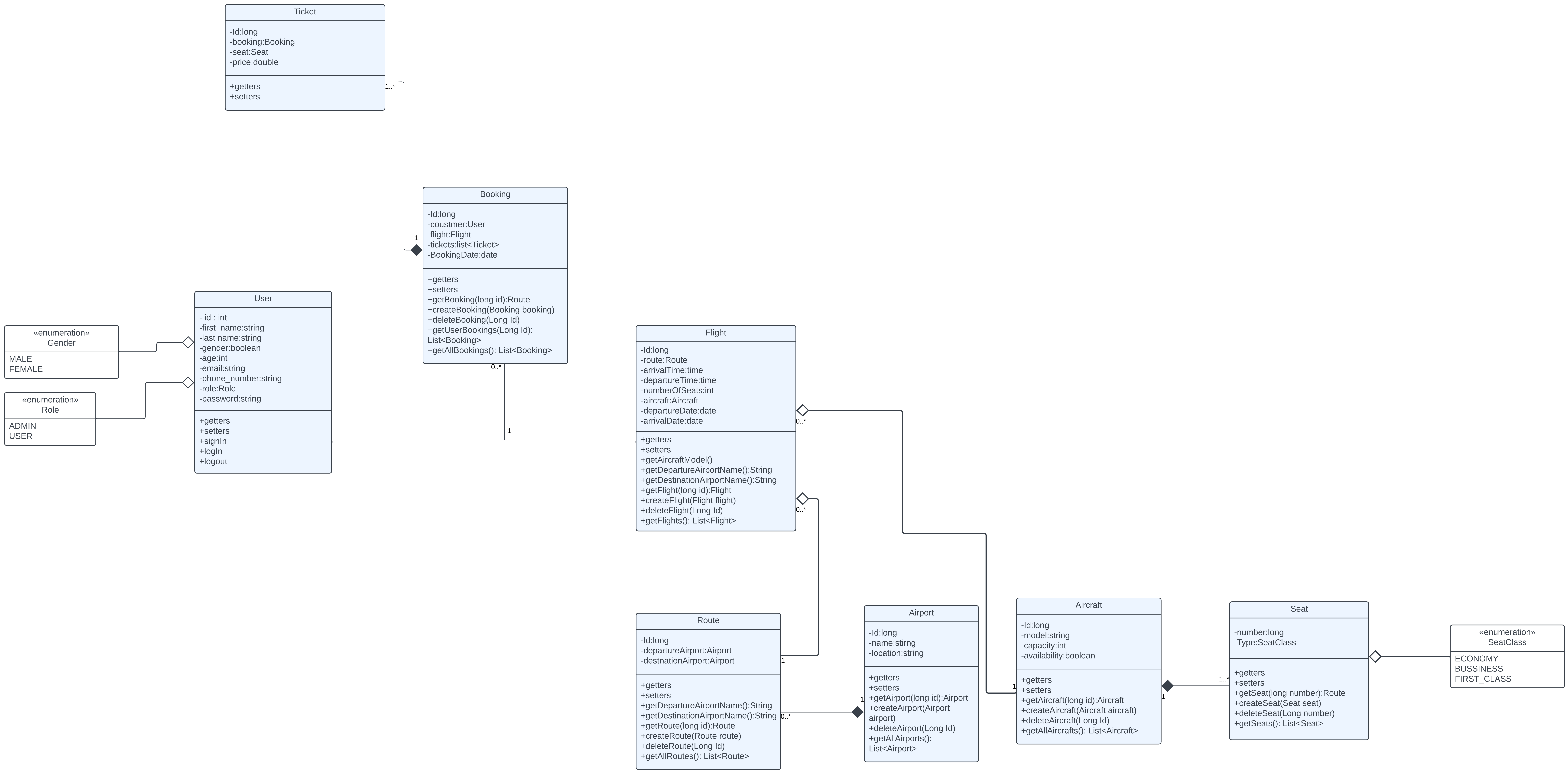
```
   totalSeats = self.capacity
   -- The number of economy seats should be 60% of the total capacity
   and self.numberOfSeats->select(seat | seat.type = SeatType.ECONOMY)->size() = self.capacity * 0.6
   -- The number of business class seats should be 20% of the total capacity
   and self.numberOfSeats->select(seat | seat.type = SeatType.BUSINESS)->size() = self.capacity * 0.2
   -- The number of first class seats should be 20% of the total capacity
   and self.numberOfSeats->select(seat | seat.type = SeatType.FIRST_CLASS)->size() = self.capacity * 0.2
```

- **context Booking**

**inv**:

```
   -- Booking must have a customer
   self.customer <> null


   -- Booking must have a flight
   and self.flight <> null
   -- Booking must have at least one ticket
   and self.tickets->notEmpty()
   -- All tickets in the booking must have the booking reference set to this booking
   and self.tickets->forAll(ticket | ticket.booking = self)
   -- Booking date must be in the past
   and self.bookingDate < LocalDateTime::now()
```


Ticket
-Id:long
-booking:Booking
-seat:Seat
-price:double
+getters
+setters
1..*
1
Booking
-Id:long
-coustmer:User
-flight:Flight
-tickets:list<Ticket>
-BookingDate:date
+getters
+setters
+getBooking(long id):Route
+createBooking(Booking booking)
+deleteBooking(Long Id)
+getUserBookings(Long Id): List<Booking>
+getAllBookings(): List<Booking>
0..*
1
User
- id : int
-first_name:string
-last name:string
-gender:boolean
-age:int
-email:string
-phone_number:string
-role:Role
-password:string
+getters
+setters
+signIn
+logIn
+logout
«enumeration»
Gender
MALE
FEMALE
«enumeration»
Role
ADMIN
USER
Flight
-Id:long
-route:Route
-arrivalTime:time
-departureTime:time
-numberOfSeats:int
-aircraft:Aircraft
-departureDate:date
-arrivalDate:date
+getters
+setters
+getAircraftModel()
+getDepartureAirportName():String
+getDestinationAirportName():String
+getFlight(long id):Flight
+createFlight(Flight flight)
+deleteFlight(Long Id)
+getFlights(): List<Flight>
0..*
0..*
Route
-Id:long
-departureAirport:Airport
-destnationAirport:Airport
+getters
+setters
+getDepartureAirportName():String
+getDestinationAirportName():String
+getRoute(long id):Route
+createRoute(Route route)
+deleteRoute(Long Id)
+getAllRoutes(): List<Route>
1
0..*
1
Airport
-Id:long
-name:stirng
-location:string
+getters
+setters
+getAirport(long id):Airport
+createAirport(Airport airport)
+deleteAirport(Long Id)
+getAllAirports(): List<Airport>
1
Aircraft
-Id:long
-model:string
-capacity:int
-availability:boolean
+getters
+setters
+getAircraft(long id):Aircraft
+createAircraft(Aircraft aircraft)
+deleteAircraft(Long Id)
+getAllAircrafts(): List<Aircraft>
1
1..*
Seat
-number:long
-Type:SeatClass
+getters
+setters
+getSeat(long number):Route
+createSeat(Seat seat)
+deleteSeat(Long number)
+getSeats(): List<Seat>
«enumeration»
SeatClass
ECONOMY
BUSSINESS
FIRST_CLASS