Infosys
Springboard

Documentation Report on

Flight Management System

Submitted By

- Ojas Tyagi
- Srivardhan Patel
- Ruchi Binayake
- Nagur Vali
- Gangothri

Under Guidance of

Mr. Suramya Biswas

INDEX

Flight Management System

Project Overview

Introduction

The Airlines Flight Management System is a web-based application designed to facilitate airline operations management, including user registration, login, and comprehensive flight management. The system leverages Java Spring Boot for backend development and JSP for the front-end, incorporating various modern web technologies such as Bootstrap for styling.

Objective

The primary objective of this project is to provide a robust and secure system for managing airline operations, ensuring a seamless user experience for both customers and administrators.

Features

- **User Registration and Login**
- **User Role Management (Customer/Admin)**
- **Password Validation with Special Character Requirements**
- **Dynamic Error Messaging using Bootstrap Alerts**
- **Flight Management**
  - CRUD operations for flights
  - CRUD operations for airports
  - CRUD operations for routes
  - Booking management
  - Modification options for existing entries
  - Deletion of entries
- **Exception Handling**
  - Meaningful error messages and prevention of duplicate user registrations
  - Handling database exceptions and providing user-friendly feedback

Project Setup

Prerequisites

- **JDK 11 or higher**
- **Maven**
- **IDE (Eclipse, IntelliJ IDEA, etc.)**
- **MySQL Database**

Installation Steps

1. **Clone the Repository**
   - Provide the Git repository link and the command to clone it.

   bash
   Copy code

```
git clone <repository-link>
```

2. **Configure Database**
    - Create a database named flight_management_system.
    - Update the application.properties file with your database credentials.

properties

```
Copy code
spring.datasource.url=jdbc:mysql://localhost:3306/flight_management_system
spring.datasource.username=<your-database-username>
spring.datasource.password=<your-database-password>
spring.jpa.hibernate.ddl-auto=update
```

Technologies Used

1. **Java Spring Boot:**
   Spring Boot is a framework that simplifies the development of Java-based applications by providing a suite of tools and features to streamline setup and development. It offers a variety of pre-configured modules and integrates seamlessly with popular frameworks, making it easier to create standalone, production-grade applications.
2. **JSP (JavaServer Pages):**
   JavaServer Pages (JSP) is a technology that helps developers create dynamically generated web pages based on HTML, XML, or other document types. JSPs are compiled into servlets by the server, allowing for the embedding of Java code within HTML to manage the display of dynamic content.
3. **Bootstrap 5:**
   Bootstrap is a popular front-end framework that provides a collection of CSS and JavaScript tools for creating responsive, mobile-first websites. Bootstrap 5 is the latest version, offering an array of new features, streamlined components, and improved grid systems for building modern web interfaces efficiently.
4. **JavaScript:**
   JavaScript is a versatile scripting language primarily used to enhance the interactivity and functionality of web pages. It enables dynamic content updates, form validation, animations, and various user interface enhancements, making it an essential tool for web development.
5. **CSS (Cascading Style Sheets):**
   CSS is used to style and layout web pages. It controls the presentation of HTML elements, including colors, fonts, spacing, and positioning. CSS helps create visually appealing and consistent designs across different devices and screen sizes.
6. **Maven:**
   Maven is a build automation and dependency management tool primarily used for Java projects. It simplifies the project setup by managing dependencies, compiling code, running tests, and packaging the project into deployable formats, ensuring consistent and reproducible builds.
7. **Tomcat Server:**
   Apache Tomcat is an open-source web server and servlet container used to deploy and run Java web applications. It provides an environment for executing Java servlets and JSPs, serving as a reliable and lightweight server for developing and testing web applications.

- **Build the Project** Explain how to build the project using Maven.
- **Run the Application** Provide the command to run the application using Maven.

Scope:

There are two categories of people who would access the system: customer and administrator. Each of these would have some exclusive privileges.

1. The customer can:

- Create his user account.
- Login into the application.
- Check for available flights.
- Make a booking.
- View the bookings made.
- Cancel a booking.

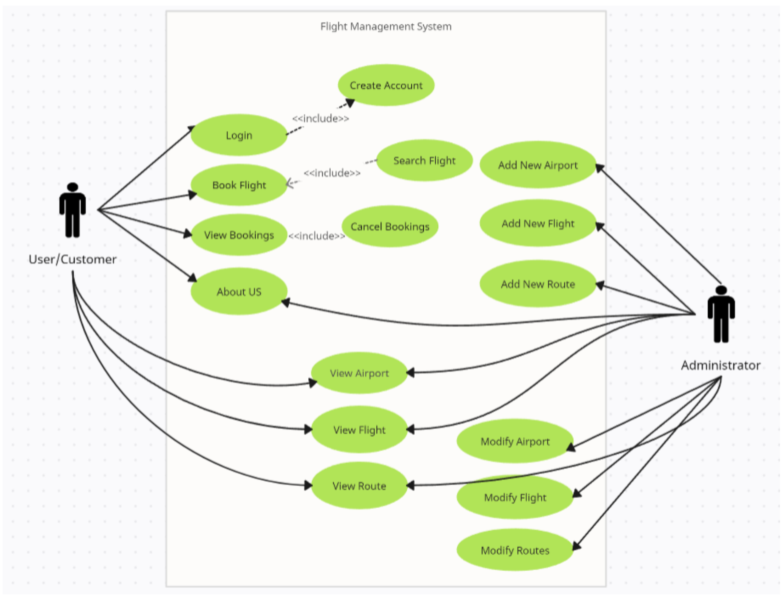
2. The administrator can:

- Login into the application.
- Add flight, airport, and route details.
- View the flight, airport, and route details.
- Delete or modify the flight, airport, and route details.

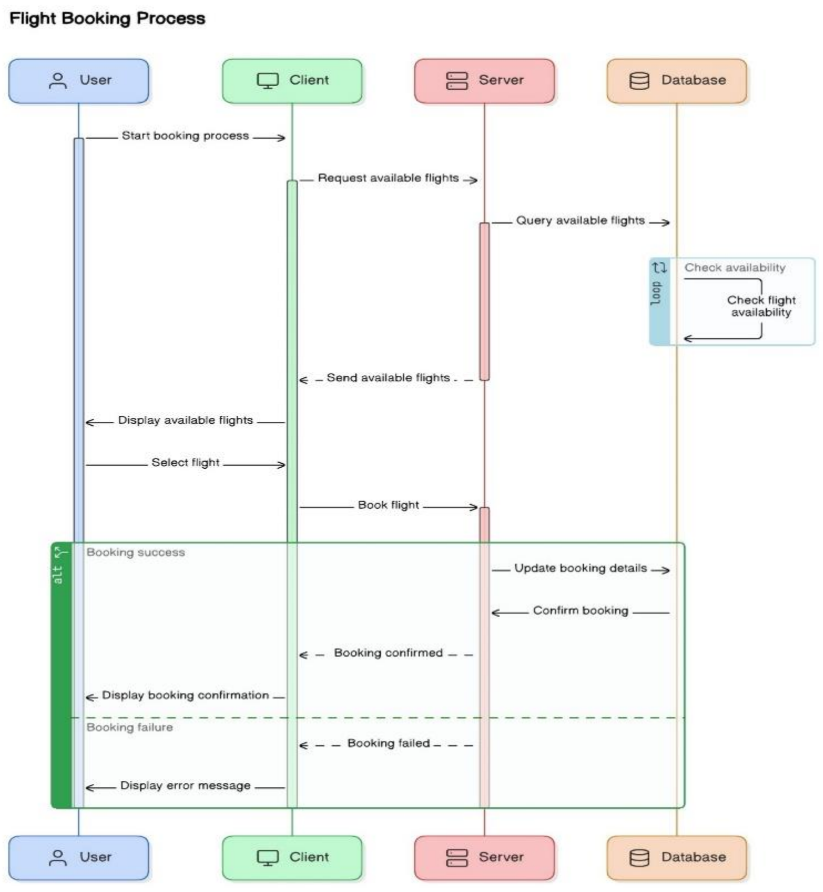
3. Out scope:

- The following functionalities have not been covered under the application:
- The application does not cover boarding pass generation and seating plans.
- Third-party applications like email & SMS integrations.
- Payments are not yet accepted by the application.

Use Case Diagrams:



Sequence Diagram:



Implementation Details

Security Configuration

Explain the purpose of the security configuration:

- **SecurityConfig.java:** Configure Spring Security to handle authentication and authorization.
    - Public access for the registration endpoint.
    - Authentication for other endpoints.
    - Custom login and logout pages.

Password Validation

Explain the password validation logic:

- **newUser.js:** JavaScript file to handle password validation on the client side.
    - Ensures passwords match.
    - Validates passwords against a regex pattern for special characters and length.
    - Displays error messages using Bootstrap alerts.

Java Class

```
@Entity
public class FlightUser extends User {
   @Id
   private String username;
   private String password;
   private String email;
   private String type;
           //getters setters
}
```

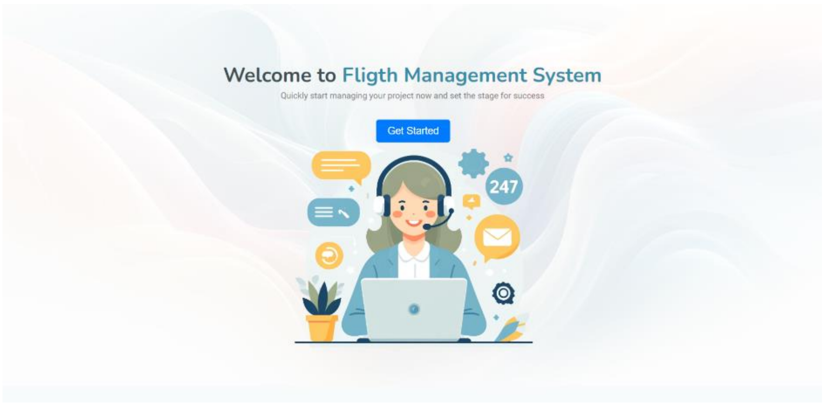

Welcome to Fligth Management System
Quickly start managing your project now and set the stage for success
Get Started

Fig 0

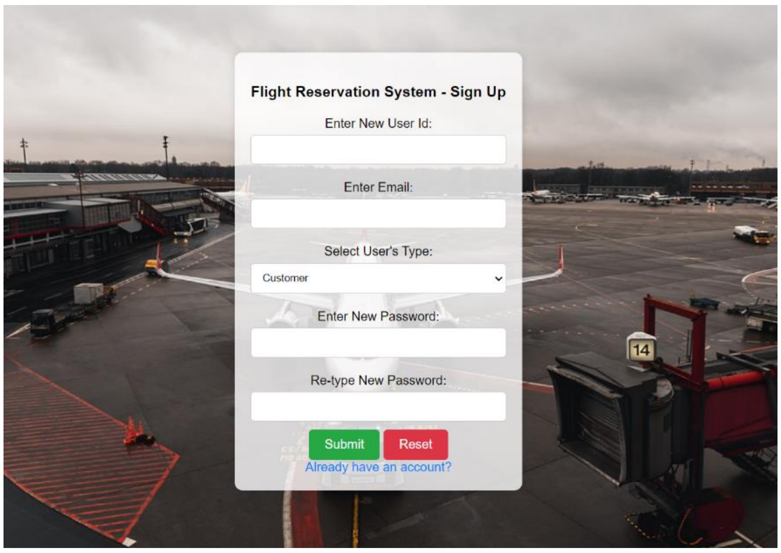


Fig 1.1

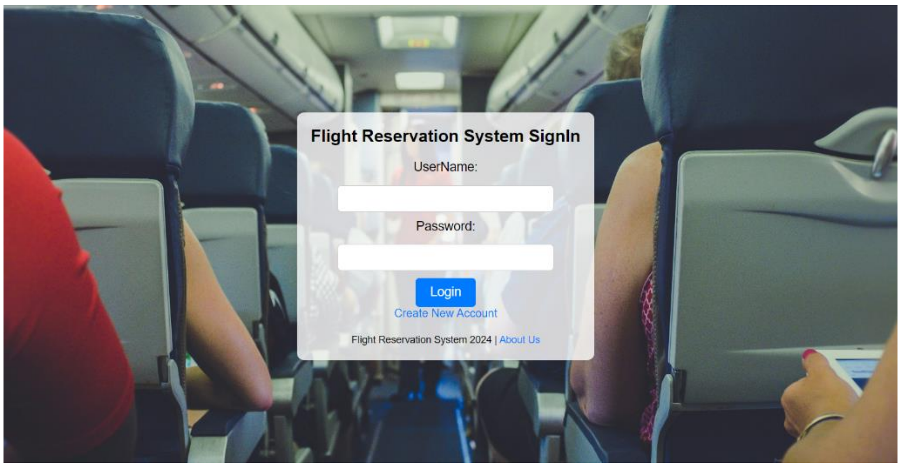

Flight Reservation System SignIn
UserName:
Password:
Login
Create New Account
Flight Reservation System 2024 | About Us

Fig 1.2

Dashboard

For easy navigation, this flight management system considers a proper dashboard

Flight Management

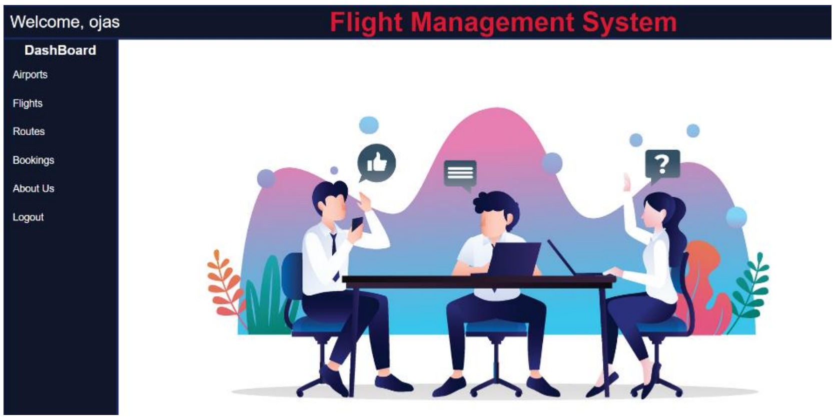


Fig 1

1. **Add New Flights:**

- Admin users can add new flights to the system by providing details such as flight number, departure and arrival airports, departure and arrival times, and the aircraft used.
- Ensures proper validation of input data to avoid errors.

2. **View Flight Details:**

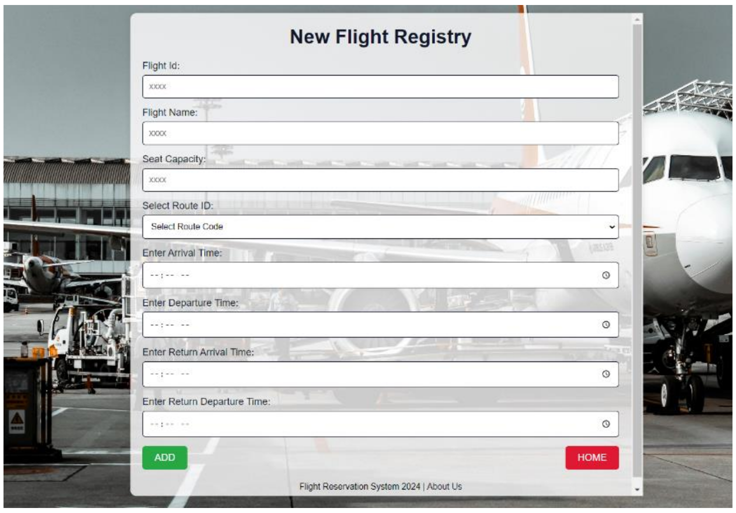
- Both admin and customer users can view flight details.
- Displays comprehensive information about each flight, such as the status, name, and route.

3. **Modify Flight:**

- Similar to the airport modification feature, we have added the ability to update flight details.
- Users can update flight information such as flight number, name, capacity, etc.

Java Class

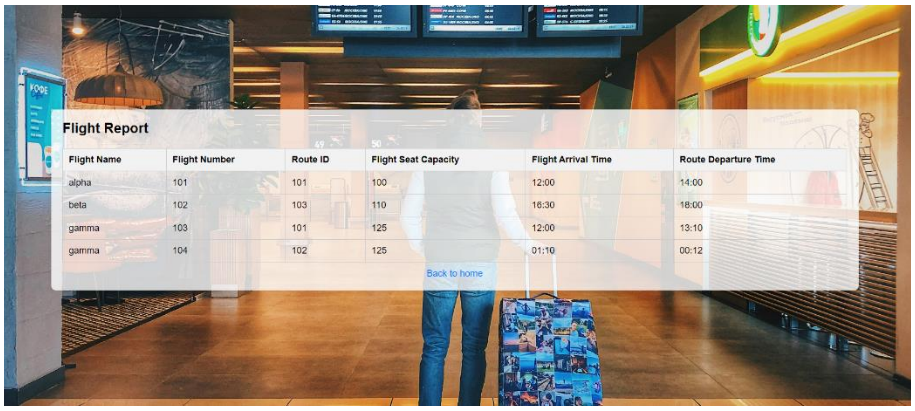
```
@Entity
public class Flight {
   @Id
   private Long flightNumber;
   private String flightName;
   private Long routeId;
   private Integer seatCapacity;
   private String departure;
   private String arrival;
   private Integer  seatBooked;
           //getters setters
}
```



New Flight Registry

Flight Id:	xxxx
Flight Name:	xxxx
Seat Capacity:	xxxx
Select Route ID:	Select Route Code
Enter Arrival Time:	--:-- --
Enter Departure Time:	--:-- --
Enter Return Arrival Time:	--:-- --
Enter Return Departure Time:	--:-- --

Fig 2.1



Flight Report

Flight Name	Flight Number	Route ID	Flight Seat Capacity	Flight Arrival Time	Route Departure Time
alpha	101	101	100	12:00	14:00
beta	102	103	110	16:30	18:00
gamma	103	101	125	12:00	13:10
gamma	104	102	125	01:10	00:12

Fig 2.2

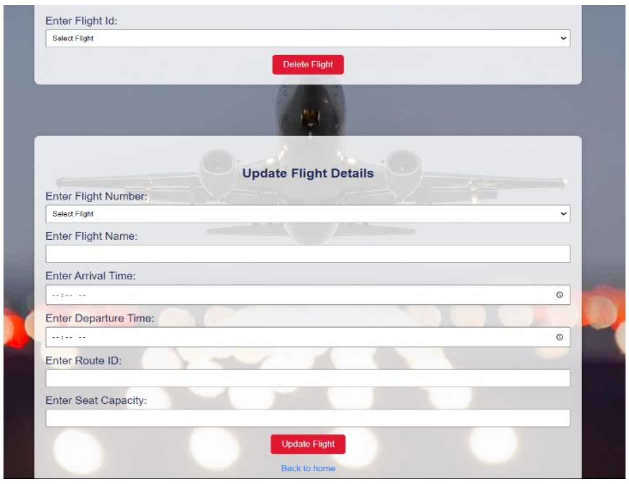


Fig 2.3

Airport Management

1. **Add New Airports:**

- Admin users can add new airports to the system by providing airport code, name, location (city and country), and other relevant details.
- Ensures that all necessary data is captured for each airport.

2. **View Airport Details:**

- Users can view detailed information about airports, including the list of flights departing from or arriving at the airport.
- Provides a comprehensive overview of airport operations.

3. **Modify Airport:**

- We have implemented a feature that allows the modification of existing airport details. Admin can update the information on the airport record entirely.
- The airport modification page includes a form where Admin can update the airport name, location, and other relevant details.

Java Class

```
@Entity
public class Airport {
  @Id
  @Column(nullable = false)
  private String airport code;
  private String airportLocation;
  private String details;
//getters setters
}
```

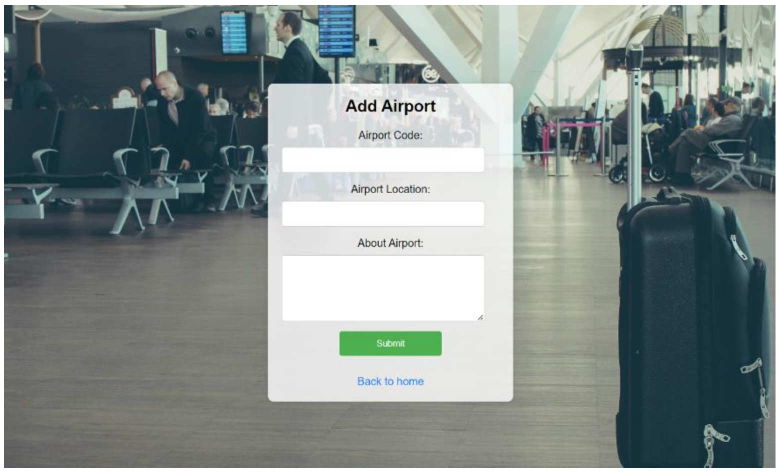
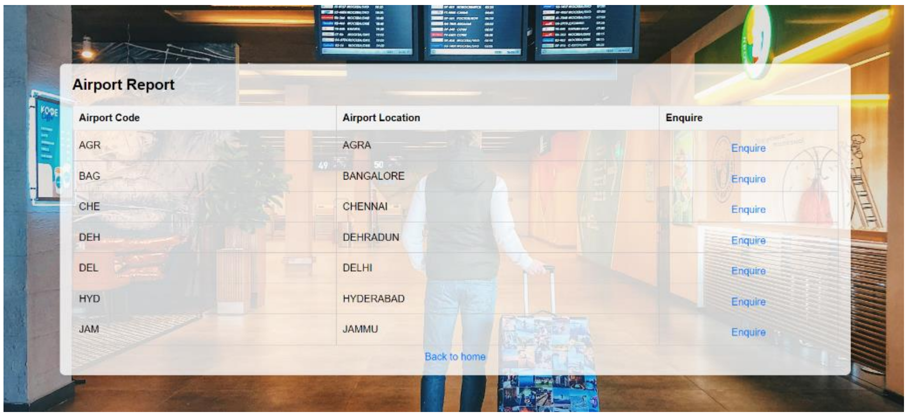


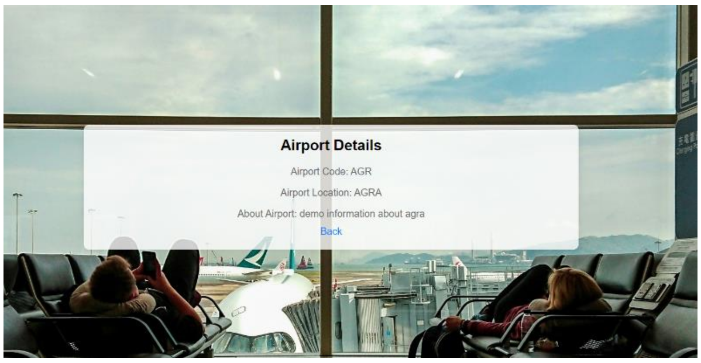
Fig 3.1



Airport Report

Airport Code	Airport Location	Enquire
AGR	AGRA	Enquire
BAG	BANGALORE	Enquire
CHE	CHENNAI	Enquire
DEH	DEHRADUN	Enquire
DEL	DELHI	Enquire
HYD	HYDERABAD	Enquire
JAM	JAMMU	Enquire

Fig 3.2



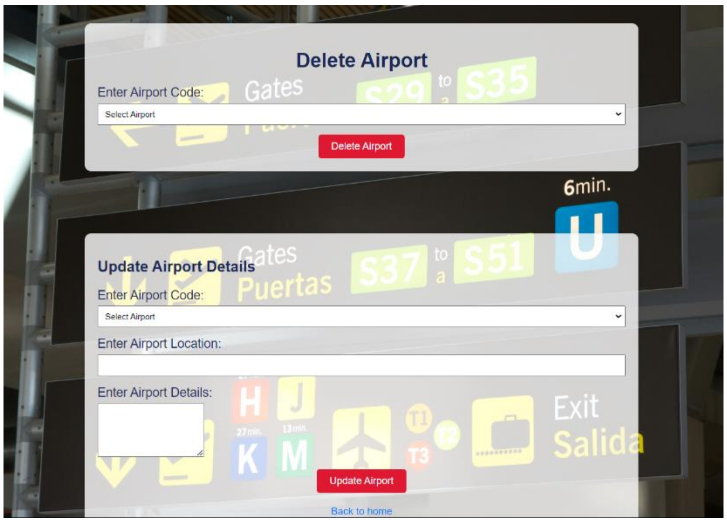
Airport Details

Airport Code: AGR

Airport Location: AGRA

About Airport: demo information about agra

Fig 3.3


Delete Airport

Enter Airport Code:

Select Airport

Delete Airport

Update Airport Details

Enter Airport Code:

Select Airport

Enter Airport Location:

Enter Airport Details:

Update Airport

Back to home

Fig 3.4

Route Management

1. **Add New Routes:**

- Admin users can define new flight routes by specifying the departure and arrival airports.
- Includes route estimated flight time, and any layovers.

2. **View Route Details:**

- Users can view detailed information about flight routes, including the list of flights operating on each route.
- Provides insight into flight connectivity and operational patterns.

3. **Modify Route:**

- We have also implemented the functionality to modify routes within the system.
- Modification Page
- Admin can update route details such as start and end locations.

Java Class

```
@Entity
public class Route {
   @Id
   private Long routeId;
   private String sourceAirportCode;
   private String destinationAirportCode;
   private Double price;
          //getters setters
}
```

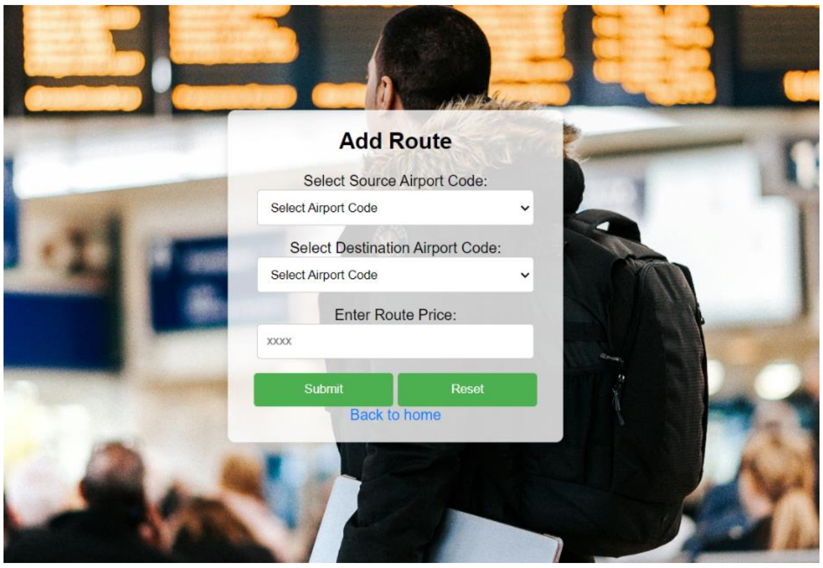
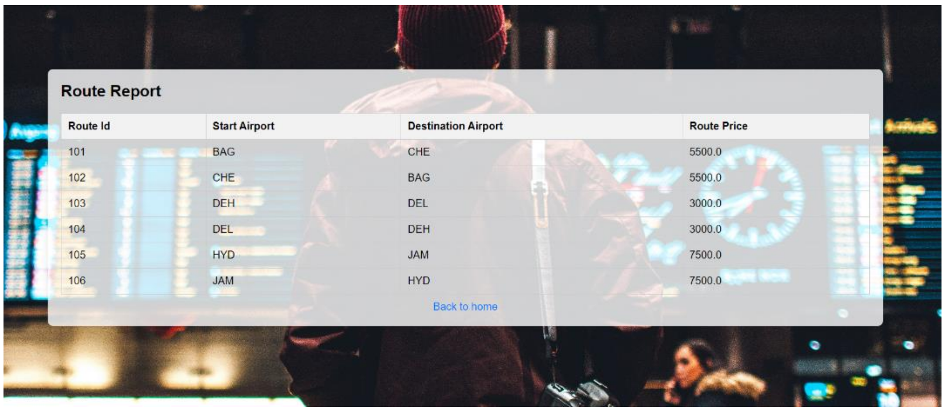


Fig 4.1



Route Report

Route Id	Start Airport	Destination Airport	Route Price
101	BAG	CHE	5500.0
102	CHE	BAG	5500.0
103	DEH	DEL	3000.0
104	DEL	DEH	3000.0
105	HYD	JAM	7500.0
106	JAM	HYD	7500.0

Back to home

Fig 4.2

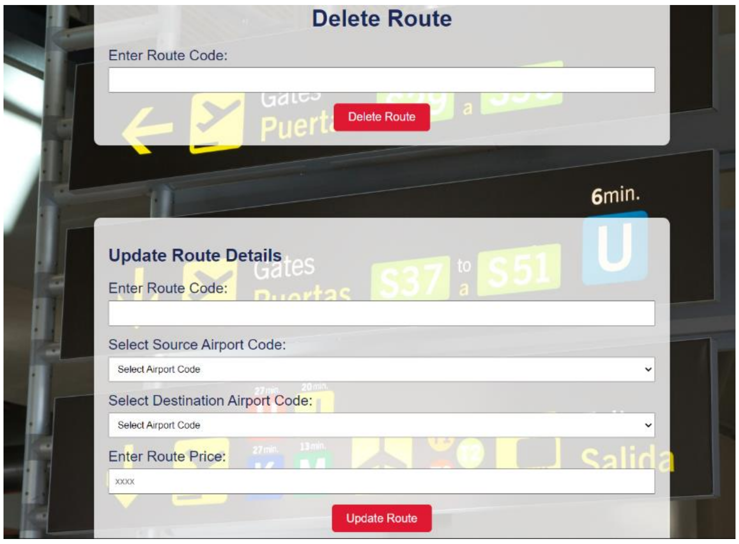


Fig 4.3

Booking Management

1. **Do Bookings:**

- Booking Creation: Users can create new bookings by providing details such as the flight number, passenger details, and seat preferences.
- Passenger Information: The system captures essential passenger information, including name, and date of birth.
- generating booking confirmations (ticket).

2. **View Booking Details:**

- Booking Summary: Users can view a summary of their booking, including flight details, and passenger information.
- Booking History: Provides users with a comprehensive history of all their bookings.

3. **Modify Booking:**

- Cancellation: Users have the option to cancel their bookings. The system processes cancellations and issues refunds according to the airline's cancellation policy.

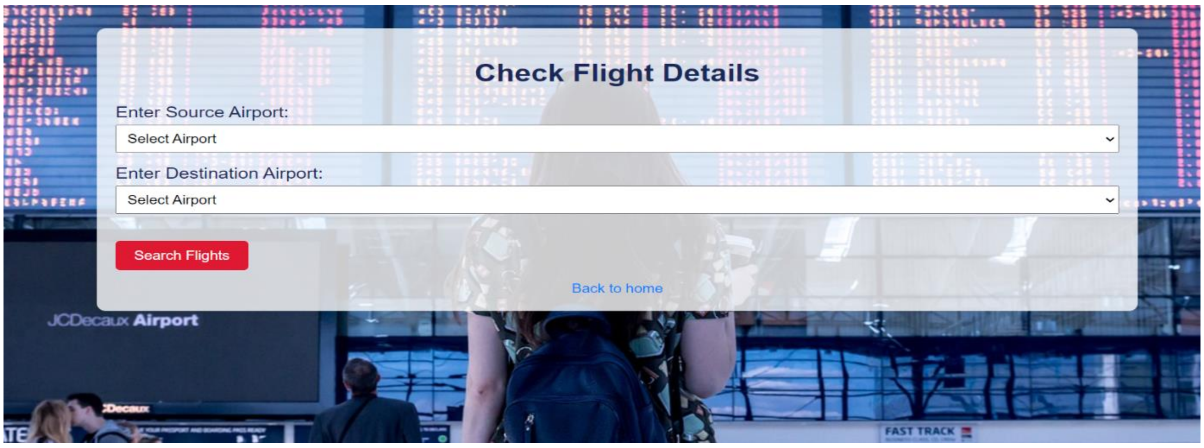
Java classes

```
public class Ticket {
   @Id
   private Long ticketNumber;
   private Long routeId;
   private Long flightNumber;
   private String flightName;
   private Double totalAmount;
         //getters setters
}
```

```
public class Passenger{
   @EmbeddedId
   private TicketPassengerEmbed embeddedId;
   private String ppassengerName;
   private String passengerDob;
   private Double price;
         //getters setters
}
```

```
@Embeddable
public class TicketPassengerEmbed  implements Serializable {
   @NotNull
   private Long ticketNumber;
   @NotNull
   private Long serialNumber;
```

```
        //getters setters
}
```

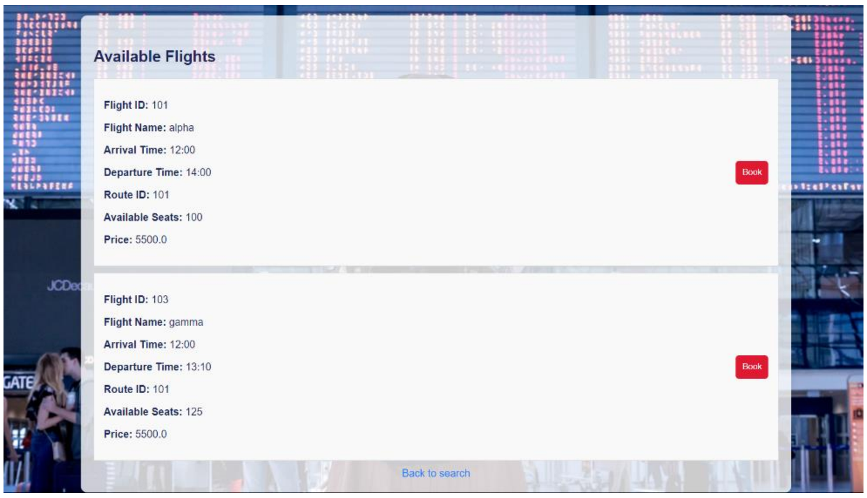


Check Flight Details

Enter Source Airport: Select Airport

Enter Destination Airport: Select Airport

Fig 5.1

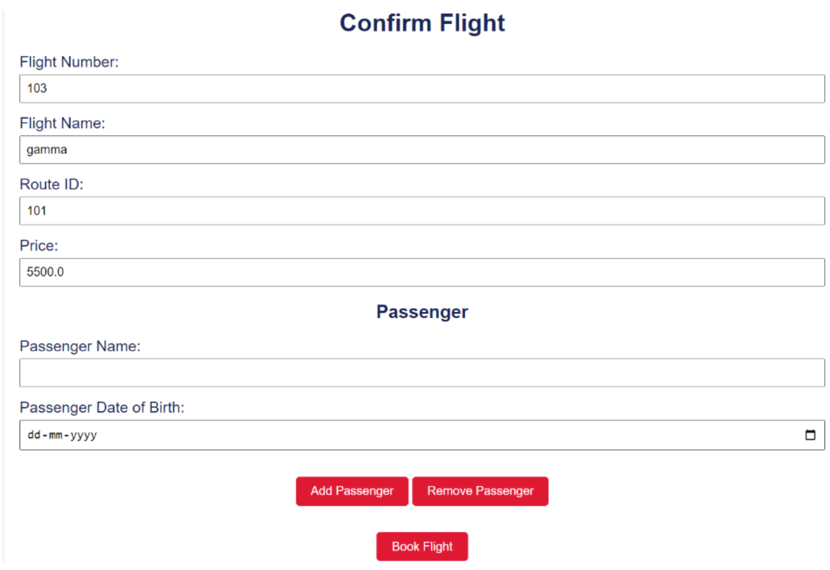


Available Flights

Flight ID: 101
Flight Name: alpha
Arrival Time: 12:00
Departure Time: 14:00
Route ID: 101
Available Seats: 100
Price: 5500.0

Flight ID: 103
Flight Name: gamma
Arrival Time: 12:00
Departure Time: 13:10
Route ID: 101
Available Seats: 125
Price: 5500.0

Fig 5.2



Confirm Flight

Flight Number: 103

Flight Name: gamma

Route ID: 101

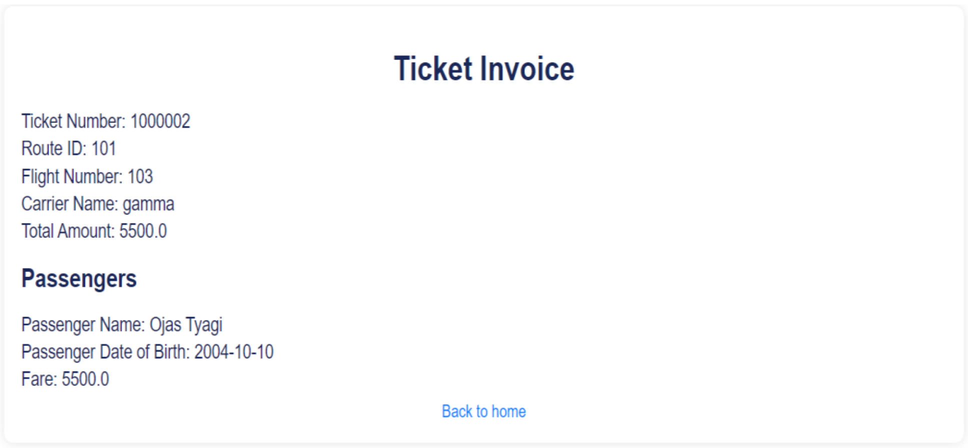
Price: 5500.0

Passenger

Passenger Name:

Passenger Date of Birth: dd-mm-yyyy

Fig 5.3



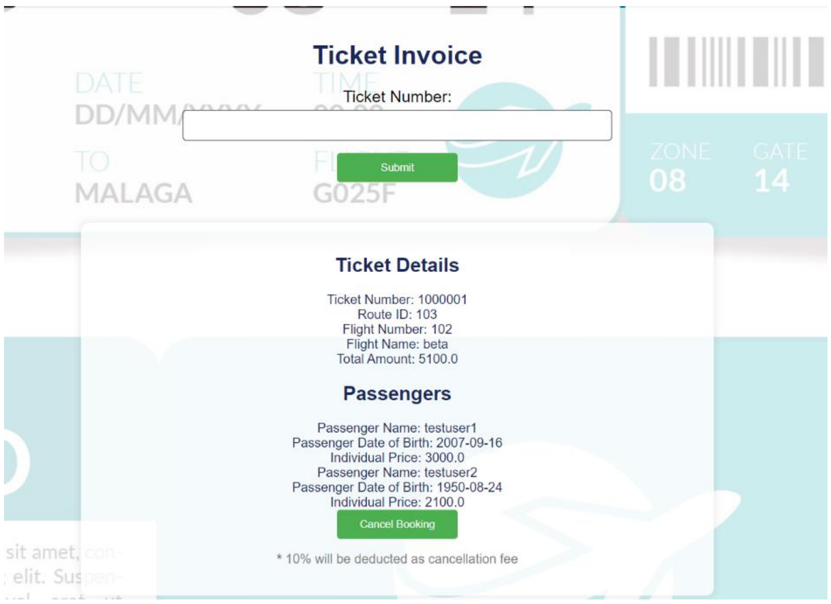
Ticket Invoice

Ticket Number: 1000002
Route ID: 101
Flight Number: 103
Carrier Name: gamma
Total Amount: 5500.0

Passengers

Passenger Name: Ojas Tyagi
Passenger Date of Birth: 2004-10-10
Fare: 5500.0

Fig 5.4



Ticket Invoice

Ticket Number:

Ticket Details

Ticket Number: 1000001
Route ID: 103
Flight Number: 102
Flight Name: beta
Total Amount: 5100.0

Passengers

Passenger Name: testuser1
Passenger Date of Birth: 2007-09-16
Individual Price: 3000.0
Passenger Name: testuser2
Passenger Date of Birth: 1950-08-24
Individual Price: 2100.0

* 10% will be deducted as cancellation fee

Fig 5.5

4. Exception Handling

Exception handling is a critical aspect of any robust application. It ensures that the system can gracefully handle unexpected errors, maintain consistent behavior, and provide meaningful feedback to users. In our Flight Management System, we have implemented comprehensive exception handling to address various types of errors that may occur during runtime.

4.1. General Approach

Our general approach to exception handling involves the use of both custom exceptions and global exception handlers. This strategy allows us to catch specific types of exceptions, provide detailed error messages, and direct users to appropriate error pages or responses.

4.2. Custom Exceptions

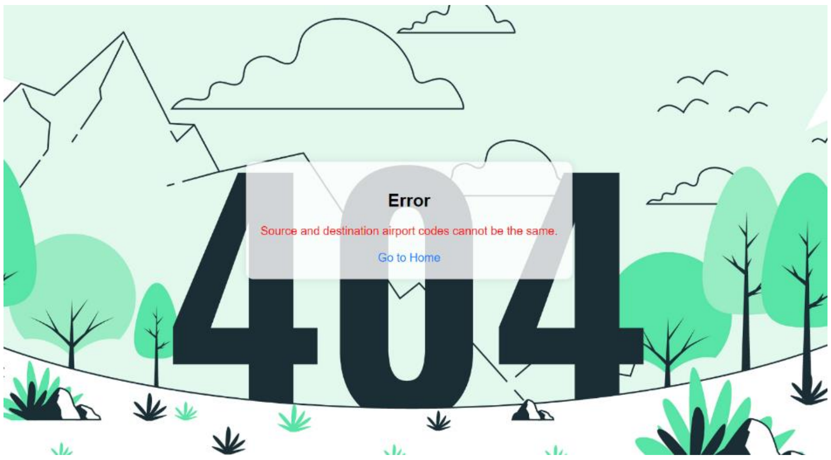
We have defined several custom exception classes to handle specific error scenarios. These custom exceptions extend the RuntimeException class and allow us to encapsulate error details that are relevant to our application's context. Examples of custom exceptions include DatabaseException, UserNotFoundException, and InvalidInputException, etc.

4.3. Global Exception Handling

To manage exceptions across the entire application, we have implemented a global exception handler using the @ControllerAdvice annotation. This global handler intercepts exceptions thrown by any controller and directs them to a centralized error-handling mechanism. The global exception handler provides a consistent way to log errors, display error messages, and redirect users to appropriate error pages.

4.4. User-Friendly Error Messages

A key aspect of our exception-handling strategy is providing user-friendly error messages. Instead of displaying technical error details, which can be confusing and unhelpful, we show clear and concise messages that guide users on how to resolve the issue. For example, instead of showing a stack trace, we might display a message like "The username you entered is already taken. Please choose a different username.



Source and destination airport codes cannot be the same.

Fig 6.1

Conclusion/Summary

In conclusion, the flight management system developed using Spring Boot and HTML & CSS provides a scalable, efficient, and secure solution for modern airline operations. It integrates key functionalities such as flight scheduling, booking management, and real-time updates, while ensuring a user-friendly interface and robust backend support.

References

1. https://www.freeprojectz.com/dfd/flight-management-system-dataflow-diagram
2. https://www.javatpoint.com/er-diagram-for-the-airline-reservation-system
3. https://www.researchgate.net/publication/224087512_Flight_management_system_prediction_and_execution_of_idle-thrust_descents
4. https://youtu.be/FcC8zhtOaSg