**Spring Boot Airline Management System Documentation**

**Name: Lande Sudam**

**Roll No: 94**

# 1. Introduction

This documentation provides a detailed guide for creating an Airline Management System web application using Spring Boot. The purpose of this application is to manage airline resources, including flights, passengers, and bookings, with a user-friendly interface and RESTful APIs.

**2. Objectives**

* To create an airline management application using Spring Boot.
* To understand the structure of a Spring Boot project in a real-world scenario.
* To implement CRUD operations for flights and passengers.
* To build and run RESTful web services for booking and cancellation.
* To learn how to configure persistence and test the application.

**3. Technologies Used**

* Spring Boot
* Java 11 or higher
* Maven
* Spring Web
* Spring Data JPA
* H2/MySQL Database
* IDE: IntelliJ IDEA / Eclipse

**4. Project Structure**

css

CopyEdit

AirlineManagementSystem/

├── src/

│ ├── main/

│ │ ├── java/com/example/airline/

│ │ │ ├── AirlineManagementSystemApplication.java

│ │ │ ├── controller/

│ │ │ │ ├── FlightController.java

│ │ │ │ ├── PassengerController.java

│ │ │ │ └── BookingController.java

│ │ │ ├── model/

│ │ │ │ ├── Flight.java

│ │ │ │ ├── Passenger.java

│ │ │ │ └── Booking.java

│ │ │ └── repository/

│ │ │ ├── FlightRepository.java

│ │ │ ├── PassengerRepository.java

│ │ │ └── BookingRepository.java

│ └── resources/

│ └── application.properties

├── pom.xml

**5. Main Application Class**

**AirlineManagementSystemApplication.java**

java

CopyEdit

package com.example.airline;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AirlineManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(AirlineManagementSystemApplication.class, args);

}

}

**6. Controller Classes**

**FlightController.java**

java

CopyEdit

@GetMapping("/flights")

public List<Flight> getAllFlights() { ... }

@PostMapping("/flights")

public Flight addFlight(@RequestBody Flight flight) { ... }

**PassengerController.java**

java

CopyEdit

@GetMapping("/passengers")

public List<Passenger> getAllPassengers() { ... }

@PostMapping("/passengers")

public Passenger addPassenger(@RequestBody Passenger passenger) { ... }

**BookingController.java**

java

CopyEdit

@PostMapping("/book")

public Booking bookFlight(@RequestParam Long flightId, @RequestParam Long passengerId) { ... }

@PostMapping("/cancel")

public Booking cancelBooking(@RequestParam Long bookingId) { ... }

**7. application.properties**

properties

CopyEdit

# Database configuration

spring.datasource.url=jdbc:h2:mem:airlinedb

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

**8. Sample pom.xml**

xml

CopyEdit

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>AirlineManagementSystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**9. How to Run the Application**

1. Open the project in your preferred IDE (IntelliJ IDEA, Eclipse).
2. Build the project using Maven.
3. Run the AirlineManagementSystemApplication.java main class.
4. Open a web browser and go to http://localhost:8080/flights to view all flights.
5. Use API testing tools (e.g., Postman) to test endpoints for adding flights, passengers, and bookings.

**10. Conclusion**

This Airline Management System application demonstrates how to build a fully functional Spring Boot application with RESTful APIs. It covers project setup, data persistence, and core airline operations, making it a useful guide for understanding enterprise Java development.

**11. References**

1. <https://spring.io/guides/gs/spring-boot/>
2. <https://www.baeldung.com/spring-boot-start>
3. <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
4. <https://www.geeksforgeeks.org/spring-boot-restful-web-services/>
5. <https://www.javatpoint.com/spring-boot-crud-rest-api>

.