# Spring Boot CRUD Operations : Onilne Complaint Management System

Name: Machave Sahil

Roll.no.: 97

## 1. Introduction

* This documentation provides a comprehensive guide to building an Airline Management System using Spring Boot. The system performs CRUD (Create, Read, Update, Delete) operations on airline resources such as flights and passengers using RESTful APIs and stores data in an H2 or MySQL database. This project showcases the use of Spring Boot, Spring Data JPA, and REST controllers to create a modular and scalable backend service.
* **2. Objectives**
* To understand and implement CRUD operations using Spring Boot.
* To build REST APIs for managing airline entities like flights and passengers.
* To connect with databases using Spring Data JPA.
* To manage dependencies using Maven.
* To implement a layered architecture including Controller, Service, and Repository layers.
* **3. Technologies Used**
* Spring Boot
* Spring Data JPA
* Spring Web
* Maven
* Java 11 or higher
* H2 or MySQL Database
* IDE: IntelliJ IDEA / Eclipse
* **4. Project Structure**
* css
* CopyEdit
* AirlineManagementSystem/
* ├── src/
* │ ├── main/
* │ │ ├── java/com/example/airline/
* │ │ │ ├── controller/
* │ │ │ ├── model/
* │ │ │ ├── repository/
* │ │ │ └── service/
* │ └── resources/
* │ └── application.properties
* ├── pom.xml
* **5. Entity Class (Flight.java)**
* java
* CopyEdit
* package com.example.airline.model;
* import jakarta.persistence.\*;
* @Entity
* public class Flight {
* @Id
* @GeneratedValue(strategy = GenerationType.IDENTITY)
* private Long id;
* private String flightNumber;
* private String destination;
* private String departureTime;
* private int availableSeats;
* // Getters and Setters
* }
* **6. Repository Interface (FlightRepository.java)**
* java
* CopyEdit
* package com.example.airline.repository;
* import com.example.airline.model.Flight;
* import org.springframework.data.jpa.repository.JpaRepository;
* public interface FlightRepository extends JpaRepository<Flight, Long> {
* }
* **7. Service Class (FlightService.java)**
* java
* CopyEdit
* package com.example.airline.service;
* import com.example.airline.model.Flight;
* import com.example.airline.repository.FlightRepository;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Service;
* import java.util.List;
* import java.util.Optional;
* @Service
* public class FlightService {
* @Autowired
* private FlightRepository repository;
* public Flight save(Flight flight) {
* return repository.save(flight);
* }
* public List<Flight> getAll() {
* return repository.findAll();
* }
* public Optional<Flight> getById(Long id) {
* return repository.findById(id);
* }
* public void delete(Long id) {
* repository.deleteById(id);
* }
* }
* **8. Controller Class (FlightController.java)**
* java
* CopyEdit
* package com.example.airline.controller;
* import com.example.airline.model.Flight;
* import com.example.airline.service.FlightService;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.http.ResponseEntity;
* import org.springframework.web.bind.annotation.\*;
* import java.util.List;
* @RestController
* @RequestMapping("/api/flights")
* public class FlightController {
* @Autowired
* private FlightService service;
* @PostMapping
* public Flight create(@RequestBody Flight flight) {
* return service.save(flight);
* }
* @GetMapping
* public List<Flight> getAll() {
* return service.getAll();
* }
* @GetMapping("/{id}")
* public ResponseEntity<Flight> getById(@PathVariable Long id) {
* return service.getById(id)
* .map(ResponseEntity::ok)
* .orElse(ResponseEntity.notFound().build());
* }
* @PutMapping("/{id}")
* public ResponseEntity<Flight> update(@PathVariable Long id, @RequestBody Flight flight) {
* return service.getById(id).map(existing -> {
* existing.setFlightNumber(flight.getFlightNumber());
* existing.setDestination(flight.getDestination());
* existing.setDepartureTime(flight.getDepartureTime());
* existing.setAvailableSeats(flight.getAvailableSeats());
* return ResponseEntity.ok(service.save(existing));
* }).orElse(ResponseEntity.notFound().build());
* }
* @DeleteMapping("/{id}")
* public ResponseEntity<Void> delete(@PathVariable Long id) {
* service.delete(id);
* return ResponseEntity.noContent().build();
* }
* }
* **9. application.properties**
* properties
* CopyEdit
* # For H2 database
* spring.datasource.url=jdbc:h2:mem:airlinedb
* spring.datasource.driverClassName=org.h2.Driver
* spring.datasource.username=sa
* spring.datasource.password=
* spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
* spring.h2.console.enabled=true
* **10. 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-data-jpa</artifactId>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-web</artifactId>
* </dependency>
* <dependency>
* <groupId>com.h2database</groupId>
* <artifactId>h2</artifactId>
* <scope>runtime</scope>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-test</artifactId>
* <scope>test</scope>
* </dependency>
* </dependencies>
* </project>
* **11. Testing the API**
* **POST** /api/flights – Add a new flight
* **GET** /api/flights – Get all flights
* **GET** /api/flights/{id} – Get flight by ID
* **PUT** /api/flights/{id} – Update flight
* **DELETE** /api/flights/{id} – Delete flight
* **12. Conclusion**
* The Airline Management System illustrates how to create a robust backend system using Spring Boot. It follows a layered architecture and clean coding principles, and serves as a practical starting point for real-world airline or transportation service applications.
* **13. References**
* <https://spring.io/projects/spring-boot>
* <https://docs.spring.io/spring-boot/docs/current/reference/html/>
* <https://www.baeldung.com/spring-boot-crud-thymeleaf>
* <https://www.geeksforgeeks.org/spring-boot-crud-operations/>
* <https://www.javatpoint.com/spring-boot-crud-example>