Flight Management System - Project Documentation

Overview

The **Flight System** is a RESTful API built with **Spring Boot** and **MySQL** that allows users to manage cities, airports, aircraft, and passengers. The system provides CRUD operations, relationship management, and integrates with a client application via HTTP requests.

Project Structure

```
FlightSystem/
│── SpringBootRestAPI/      # Backend Server - REST API
│── SpringHttpClient/       # Client Application - Calls API
│── .github/workflows/      # CI/CD GitHub Actions
│── pom.xml                 # Parent Maven Configuration
```

Backend (SpringBootRestAPI)

```
SpringBootRestAPI/
│── src/
│   ├── main/
│   │   ├── java/com/keyin/
│   │   │   ├── controller/  # REST API Controllers
│   │   │   ├── domain/      # Entity Models
│   │   │   ├── repository/  # Spring Data JPA Repositories
│   │   │   ├── service/     # Business Logic Layer
│   │   ├── resources/
│   │   │   ├── application.properties  # Database Configurations
│   │   │   ├── schema.sql  # Optional - Database Initialization
│   ├── test/java/com/keyin/  # Unit & Integration Tests
│── pom.xml                 # Maven Dependencies
```

Client (SpringHttpClient)

```
SpringHttpClient/
│── src/
│   ├── main/
│   │   ├── java/com/keyin/client/
│   │   │   ├── models/    # Client Data Models
│   │   │   ├── services/  # API Call Services
│   │   │   ├── RestClient.java  # HTTP Client
│   ├── resources/
│   ├── test/java/com/keyin/client/  # Unit Tests
│── pom.xml                 # Maven Dependencies
```

Technology Stack

- **Backend:** Java, Spring Boot, Spring Data JPA, MySQL
- **Client:** Java (REST API Consumer using RestTemplate)
- **Database:** MySQL
- **Testing:** JUnit, Mockito, Postman
- **CI/CD:** GitHub Actions
- **Build Tool:** Maven

Trunk-Based Development Workflow (PR Process)

1. **Initialize Git**: git init
2. **Push to GitHub**:
   ```
   git remote add origin
   https://github.com/your-username/FlightManagementSystem.git
   git push -u origin main
   ```

3. **Create a Feature Branch**:
   ```
   git checkout -b feature-add-airport-endpoints
   ```

4. **Make changes and commit**:
   ```
   git add .
   git commit -m "Added Airport CRUD API"
   ```

5. **Push Feature Branch**:
   ```
   git push -u origin feature-add-airport-endpoints
   ```
6. **Create a Pull Request (PR)** in GitHub → Merge after review.

Database Schema (SQL)

```
CREATE TABLE cities (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE,
    country VARCHAR(255) NOT NULL
);

CREATE TABLE airports (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    code VARCHAR(10) NOT NULL UNIQUE,
    city_id BIGINT,
    FOREIGN KEY (city_id) REFERENCES cities(id) ON DELETE CASCADE
);

CREATE TABLE passengers (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) NOT NULL UNIQUE,
    city_id BIGINT,
    FOREIGN KEY (city_id) REFERENCES cities(id) ON DELETE CASCADE
);

CREATE TABLE aircraft (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    type VARCHAR(255) NOT NULL,
    airline_name VARCHAR(255) NOT NULL,
    number_of_passengers INT NOT NULL
);
```

REST API Endpoints

Cities

Method	Endpoint	Description
GET	/api/cities	Get all cities
POST	/api/cities	Add a new city
GET	/api/cities/{id}	Get city by ID
PUT	/api/cities/{id}	Update a city
DELETE	/api/cities/{id}	Delete a city

Airports

Method	Endpoint	Description
GET	/api/airports	Get all airports
POST	/api/airports	Add an airport
GET	/api/airports/{id}	Get airport by ID
PUT	/api/airports/{id}	Update an airport
DELETE	/api/airports/{id}	Delete an airport

Running the Project

1Setup MySQL Database

```
mysql -u root -p
CREATE DATABASE flight_db;
```

2Run Backend API

```
cd SpringBootRestAPI
mvn spring-boot:run
```

3Run Client Application

```
cd SpringHttpClient
mvn exec:java -Dexec.mainClass="com.keyin.client.RestClient"
```

Testing with Postman

- **Use Postman to send POST requests** to create cities, airports, passengers, and aircraft.
- **Verify with GET requests** to check stored data.

Sample POST Request (Create City)

```
{
   "name": "New York",
   "country": "USA"
}
```

Sample GET Response (Retrieve Cities)

```
[
   { "id": 1, "name": "New York", "country": "USA" },
   { "id": 2, "name": "Los Angeles", "country": "USA" }
]
```

Conclusion

Complete REST API with Spring Boot & MySQL
CI/CD setup with GitHub Actions
Trunk-Based Development Workflow with PRs
Tested with Postman and JUnit

Project is ready for development and deployment!

Flight_management_db:

Cities ↔ **Airports** (One-to-Many)
Cities ↔ **Passengers** (One-to-Many)
Passengers ↔ **Aircraft** (Many-to-Many)
Aircraft ↔ **Airports** (Many-to-Many)

The schema shows all relationships structure in a relational database.

Cities Table

```
CREATE TABLE cities (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE,
    country VARCHAR(255) NOT NULL
);
```

```
{
  "name": "New York",
  "country": "USA"
}

{
  "name": "Los Angeles",
  "country": "USA"
}

{
  "name": "Toronto",
  "country": "Canada"
}
```

Airports Table

```
CREATE TABLE airports (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    code VARCHAR(10) NOT NULL UNIQUE,
    city_id BIGINT,
    FOREIGN KEY (city_id) REFERENCES cities(id) ON DELETE CASCADE
);
```

```
{
  "name": "JFK International",
  "code": "JFK",
  "cityId": 1
}
```

```
{
  "name": "LAX Airport",
  "code": "LAX",
  "cityId": 2
}

{
  "name": "Toronto Pearson",
  "code": "YYZ",
  "cityId": 3
}
```

Passengers Table

```
CREATE TABLE passengers (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) NOT NULL UNIQUE,
    city_id BIGINT,
    FOREIGN KEY (city_id) REFERENCES cities(id) ON DELETE CASCADE
);
```

```
{
  "firstName": "John",
  "lastName": "Doe",
  "phoneNumber": "123-456-7890",
  "cityId": 1
}

{
  "firstName": "Jane",
  "lastName": "Smith",
  "phoneNumber": "987-654-3210",
  "cityId": 2
}
```

Aircraft Table

```
CREATE TABLE aircraft (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    type VARCHAR(255) NOT NULL,
    airline_name VARCHAR(255) NOT NULL,
    number_of_passengers INT NOT NULL
);
```

```
{
  "type": "Boeing 747",
  "airlineName": "United Airlines",
  "numberOfPassengers": 400
}
```

```
{
  "type": "Airbus A320",
  "airlineName": "Delta Airlines",
  "numberOfPassengers": 180
}
```

Passenger-Aircraft Many-to-Many Table

```
CREATE TABLE passenger_aircraft (
    passenger_id BIGINT,
    aircraft_id BIGINT,
    PRIMARY KEY (passenger_id, aircraft_id),
    FOREIGN KEY (passenger_id) REFERENCES passengers(id) ON DELETE CASCADE,
    FOREIGN KEY (aircraft_id) REFERENCES aircraft(id) ON DELETE CASCADE
);
```

Aircraft-Airports Many-to-Many Table

```
CREATE TABLE aircraft_airports (
    aircraft_id BIGINT,
    airport_id BIGINT,
    PRIMARY KEY (aircraft_id, airport_id),
    FOREIGN KEY (aircraft_id) REFERENCES aircraft(id) ON DELETE CASCADE,
    FOREIGN KEY (airport_id) REFERENCES airports(id) ON DELETE CASCADE
);
```

Entity-Relationship Diagram (ERD)

```
+--------------+     +--------------+     +--------------+     +--------------+
|   Cities     | 1 -- |   Airports   |  --  |  Aircraft    | --  M | Passengers  |
+--------------+     +--------------+     +--------------+     +--------------+
       |                    |                  |                      |
       |                    |                  |                      |
       |                    |                  |                      |
       1                    M                  M                      M
```