### **Simply Fly - Air Ticket Booking System**

**Project Overview:**

A secure backend system for booking and managing flights, built using Java and Spring Boot. It supports three roles: **Admin**, **Flight Owner**, and **User**, all authenticated with JWT.

### **Core Features**

* **User**: Register, login, search flights, book tickets, view/update profile
* **Flight Owner**: Add/manage flights they own
* **Admin**: Full control – manage users, routes, flights, and bookings
* **JWT Authentication**: Role-based secure access
* **Swagger UI**: Interactive API testing

### **Tech Stack**

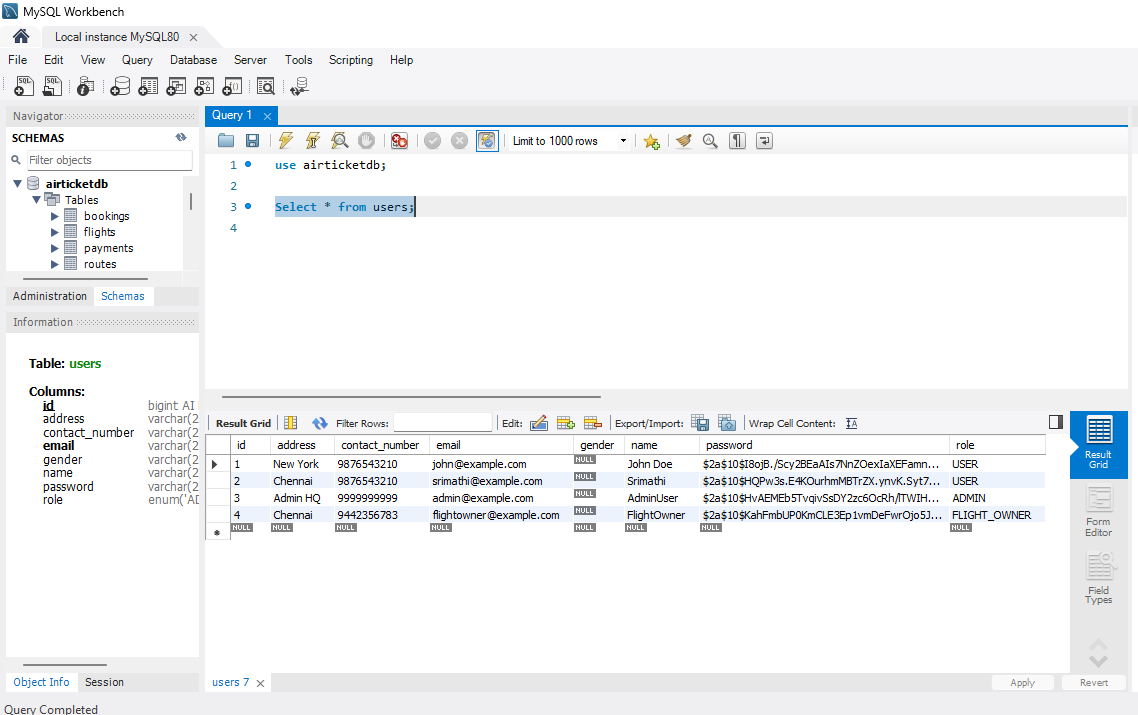
* Java 17, Spring Boot, Spring Security
* JWT (stateless auth), MySQL, JPA (Hibernate)
* Swagger (OpenAPI 3) for documentation

### **Key Packages**

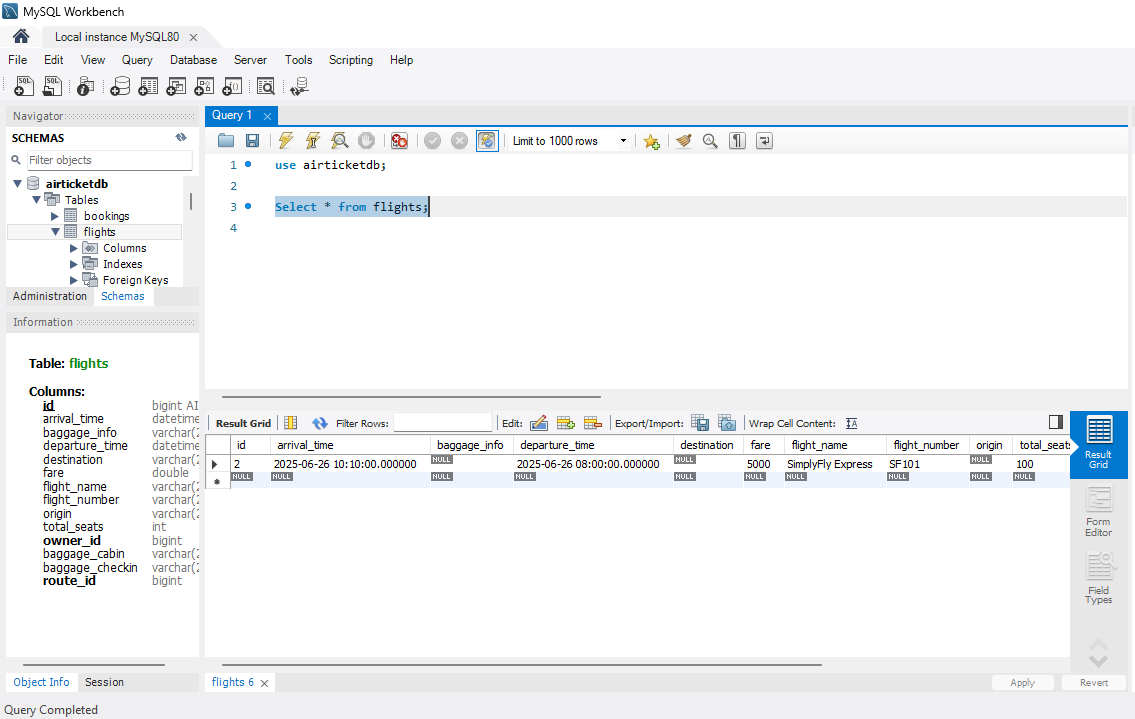
1. controller – API endpoints
2. dto – Request & response data structures
3. entity – Database models (User, Flight, Booking, etc.)
4. service – service interface
5. service implementation - Business logic
6. repository – Data access (JPA)
7. security – JWT utilities & filters
8. config – App & Swagger configuration
9. exception - custom exceptions and global exception handler

**Database Schema:**

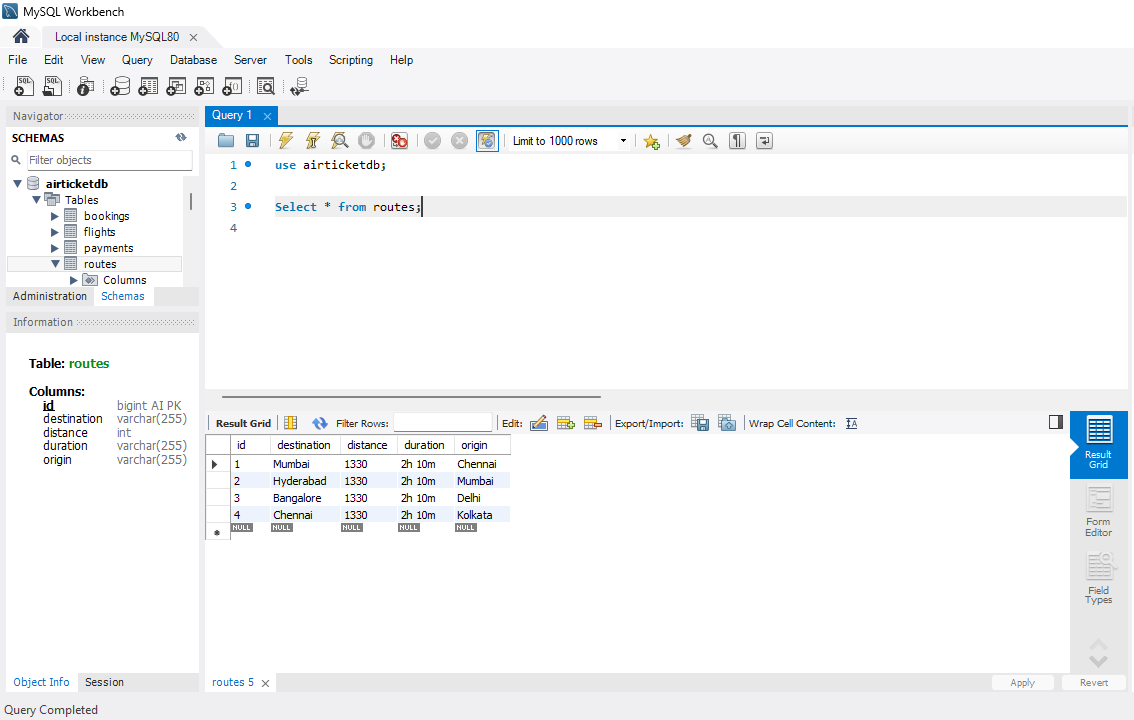
User Table:



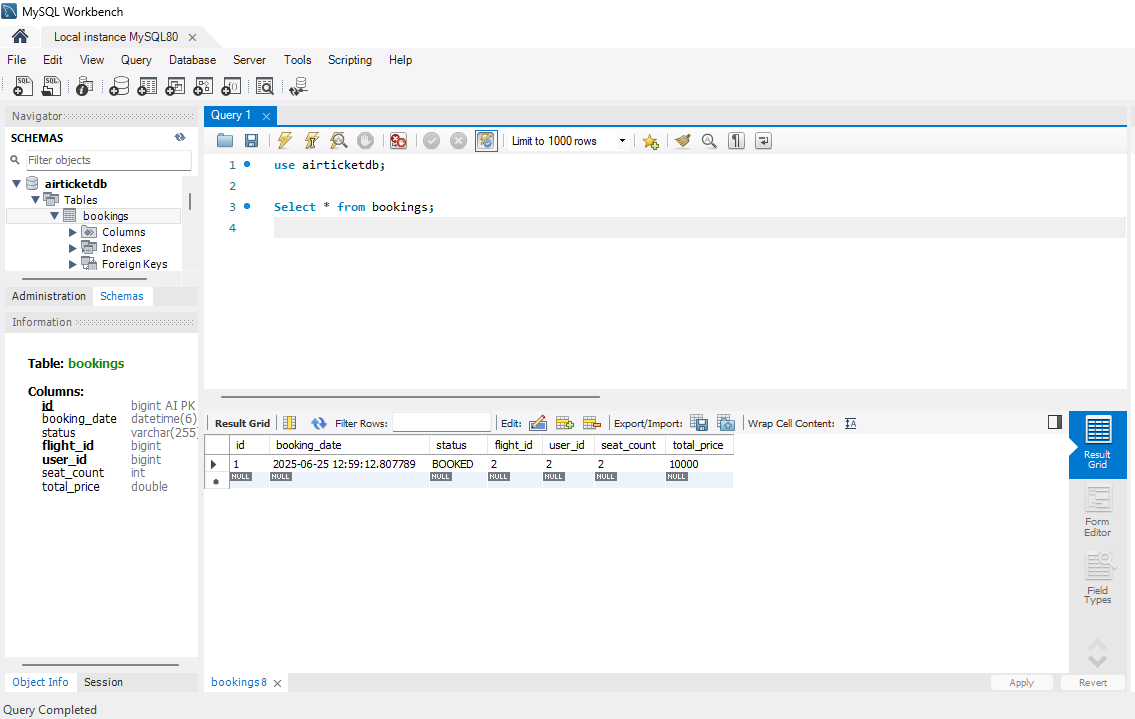
Flights Table:



Routes Table:



Bookings Table:



**Role-wise Features:**

|  |  |
| --- | --- |
| **ROLE** | **CAPABILITIES** |
| Admin | Manage users, routes, flights |
| Flight Owner | Add/manage their own flights |
| User | Search flights, book tickets, view profile |

### **Admin**

Admins have full access across the system for monitoring and management.

* **User Management**
  + GET /api/users → View all users
  + GET /api/users/{id} → View specific user
  + DELETE /api/users/{id} → Delete user
* **Flight Management**
  + POST /api/flights → Add new flights
  + PUT /api/flights/{id} → Update any flight
  + DELETE /api/flights/{id} → Delete any flight
  + GET /api/flights → View all flights
  + GET /api/flights/{id} → View flight details
  + POST /api/flights/search → Search flights
* **Booking Management**
  + GET /api/bookings/all → View all bookings
  + DELETE /api/bookings/{id} → Cancel any booking

### **User**

Regular users can book flights and manage their own profile.

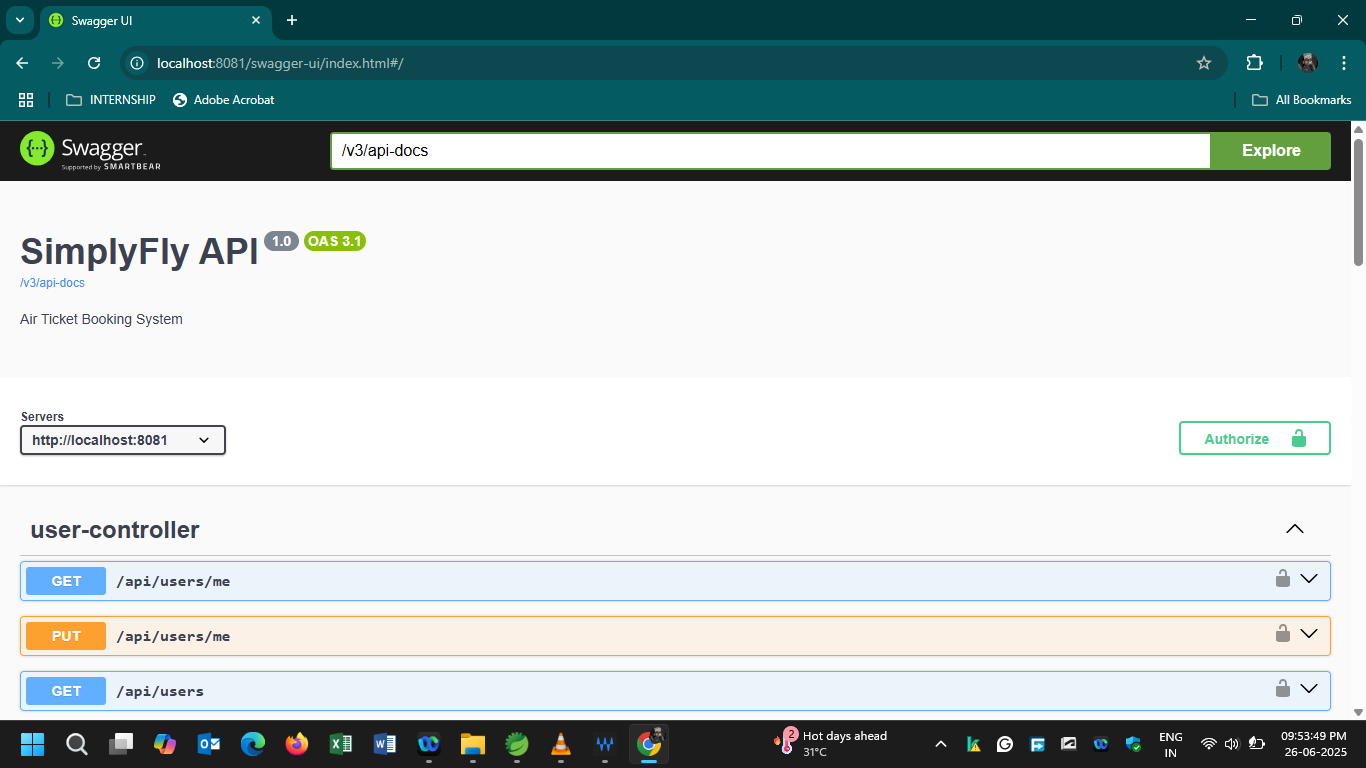
* **Authentication**
  + POST /api/auth/register → Register
  + POST /api/auth/login → Login
* **Profile**
  + GET /api/users/me → View own profile
  + PUT /api/users/me → Update own profile
* **Flight Viewing**
  + GET /api/flights → View available flights
  + GET /api/flights/{id} → View flight detail
  + POST /api/flights/search → Search by route and date
* **Booking**
  + POST /api/bookings → Book a flight
  + GET /api/bookings/my → View own bookings
* **Seat**
  + GET /api/seats/flight/{flightId} → View available seats on a flight

### **Flight Owner**

Flight owners can manage their own flights.

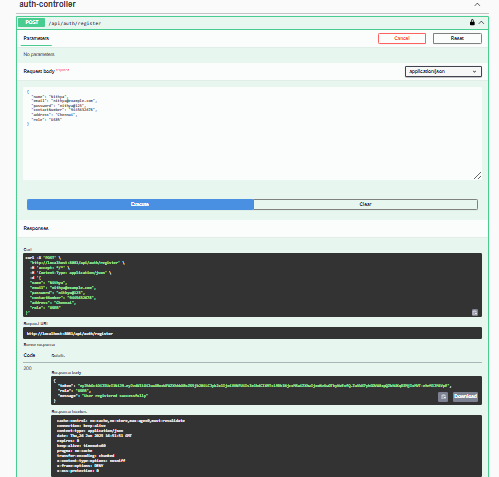
* **Flight Management**
  + POST /api/flights → Add flights they own
  + PUT /api/flights/{id} → Update flights they own
  + DELETE /api/flights/{id} → Delete flights they own
  + GET /api/flights → View all their flights
  + GET /api/flights/{id} → View details
* **Seat View**
  + GET /api/seats/flight/{flightId} → View seat map for owned flights

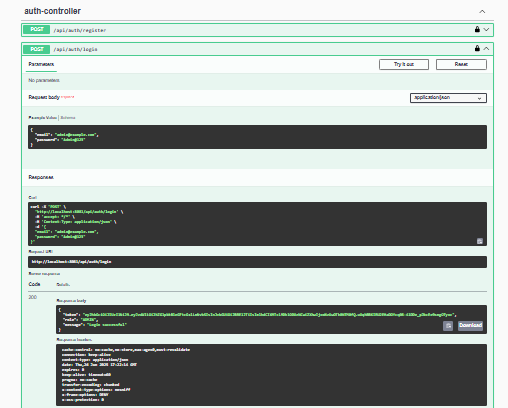
**Screenshots - Swagger UI:**

****

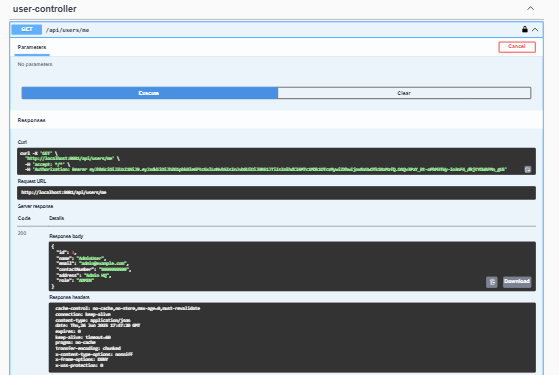
****

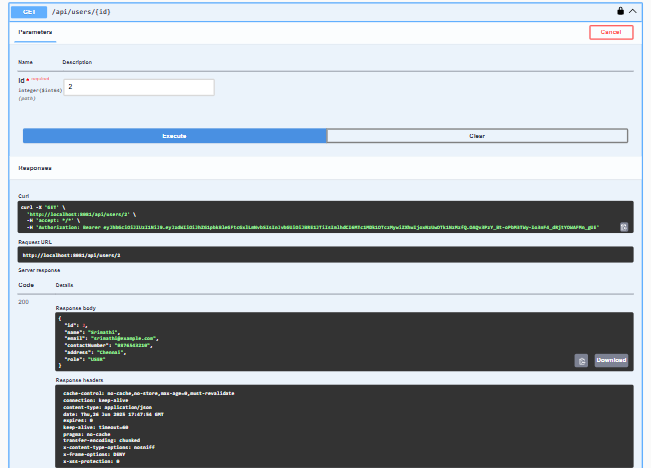
**Auth Controller (Register & Login):**

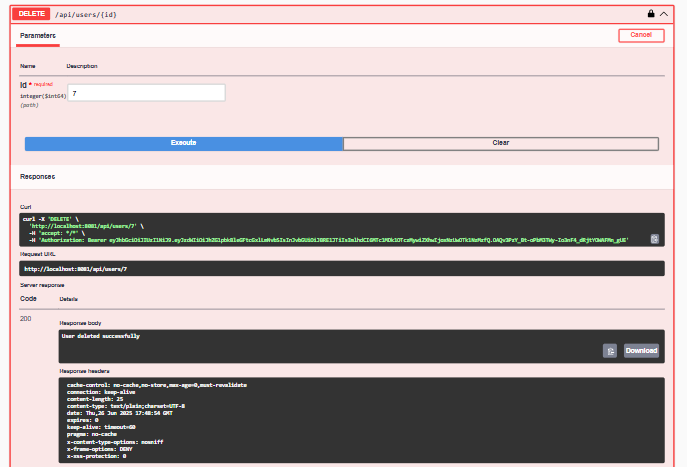
****

****

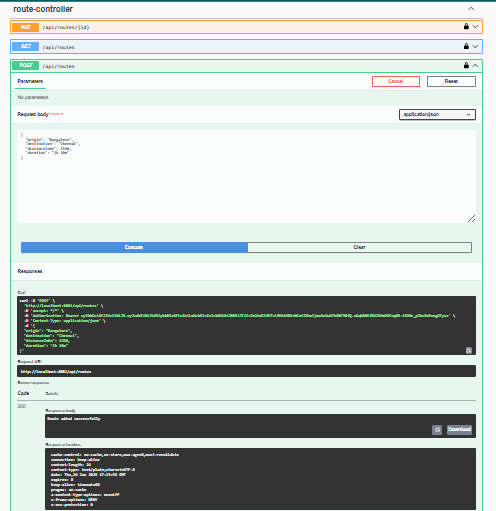
**User Controller – managing user information**

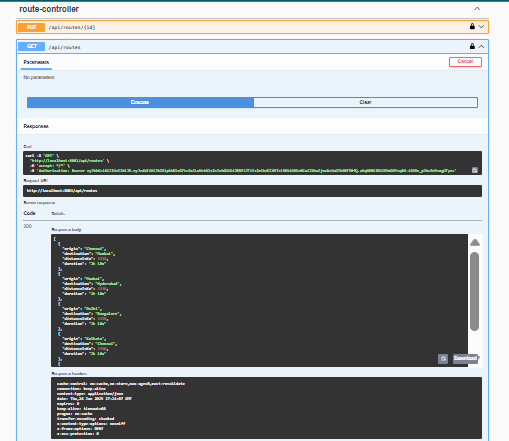
****

****

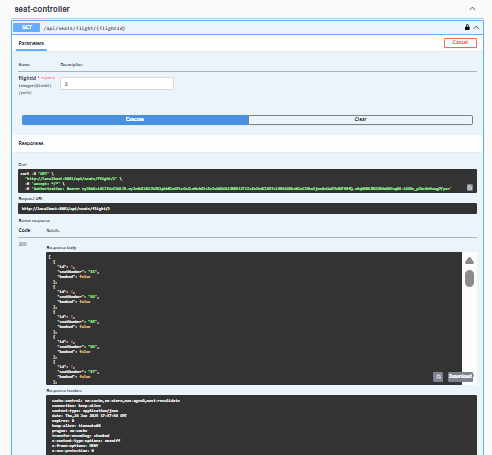
****

**Route Controller – creating and managing routes**

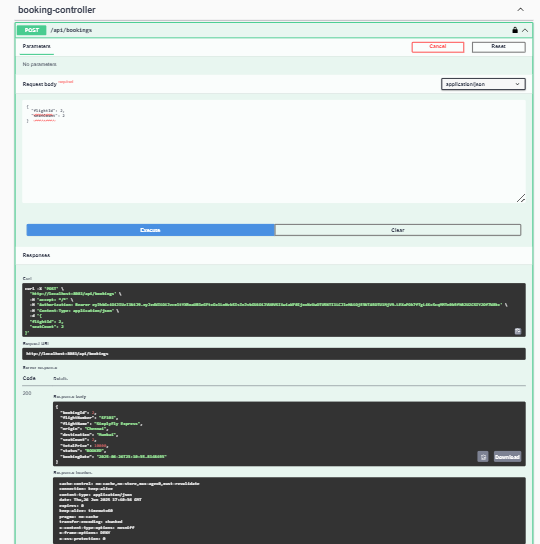
****

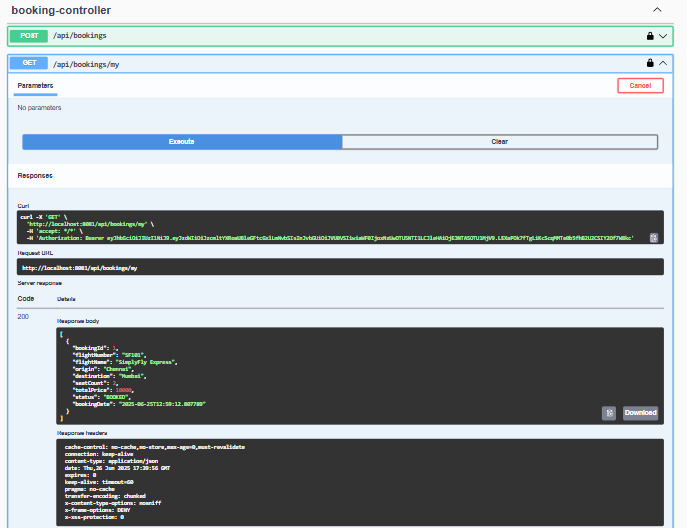
****

**Seat Controller – check seat availability**

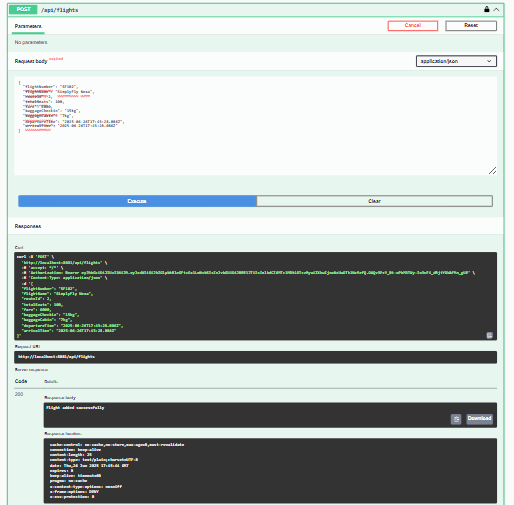
****

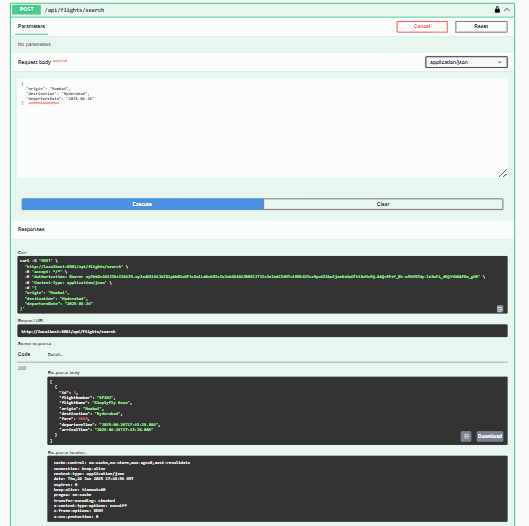
**Booking Controller – Book and manage flights**

****

****

**Flight Controller – Create and manage flight information:**

****

****

**Sample JSON Requests and Responses:**

### 1. **Register User**

Request:

**POST** /api/auth/register

json

{

"name": "Srimathi",

"email": "srimathi@example.com",

"password": "sri@123",

"contactNumber": "9876543210",

"address": "Chennai",

"role": "USER"

}

Response:

json

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmltYXRoaUBleGFtcGxlLmNvbSIsInJvbGUiOiJVU0VSIiwiaWF0IjoxNzUxMDAwMDAwLCJleHAiOjE3NTEzNjAwMDB9.hNXXEXAMPLEjwtToken123",

"role": "USER",

"message": "User registered successfully"

}

### 2. **Login User**

**POST** /api/auth/login

**Request:**

json

{

"email": "srimathi@example.com",

"password": "sri@123"

}

Response:

Json

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmltYXRoaUBleGFtcGxlLmNvbSIsInJvbGUiOiJVU0VSIiwiaWF0IjoxNzUxMDAwMDAwLCJleHAiOjE3NTEzNjAwMDB9.hNXXEXAMPLEjwtToken123",

"role": "USER",

"message": "Login successful"

}

### 3. **Register Admin**

**POST** /api/auth/register

Request:

json

{

"name": "Admin",

"email": "admin@example.com",

"password": "Admin@123",

"contactNumber": "9876543211",

"address": "Bangalore",

"role": "ADMIN"

}

Response:

Json

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmltYXRoaUBleGFtcGxlLmNvbSIsInJvbGUiOiJVU0VSIiwiaWF0IjoxNzUxMDAwMDAwLCJleHAiOjE3NTEzNjAwMDB9.hNXXEXAMPLEjwtToken123",

"role": "ADMIN",

"message": "User registered successfully"

}

### 4. **Add Flight (by Admin or Flight Owner)**

**POST** /api/flights

Request:

json

{

"flightNumber": "SF101",

"flightName": "SimplyFly Express",

"routeId": 1,

"totalSeats": 100,

"fare": 5000.00,

"baggageCheckin": "15kg",

"baggageCabin": "7kg",

"departureTime": "2025-06-26T08:00:00",

"arrivalTime": "2025-06-26T10:10:00"

}

Response:

Json

json

{

"message": "Flight added successfully with 100 seats"

}

### 5. **Add Route (by Admin)**

**POST** /api/routes

Request:

json

{

"origin": "Chennai",

"destination": "Mumbai",

"distanceInKm": 1200,

"duration": "2h 30m"

}

Response:

Json

{

"message": "Route successfully added: Chennai -> Mumbai"

}

### 6. **Search Flights**

**POST** /api/search

Request:

json

{

"origin": "Chennai",

"destination": "Mumbai",

"departureDate": "2025-06-26"

}

Response:

json

[

{

"id": 1,

"flightNumber": "SF101",

"flightName": "SimplyFly Express",

"origin": "Chennai",

"destination": "Mumbai",

"fare": 5000.0,

"departureTime": "2025-06-26T08:00:00",

"arrivalTime": "2025-06-26T10:10:00"

}

]

7. **Book a Flight**

**POST** /api/bookings

Request:

json

{

"flightId": 1,

"seatCount": 2

}

Response:

json

{

"message": "Flight booked successfully",

"bookingId": 101,

"totalPrice": 10000.0,

"status": "BOOKED"

}

### 8. **Update User Profile**

**PUT** /api/users/me

Request:

json

{

"name": "Srimathi Raju",

"contactNumber": "9876543210",

"address": "Chennai - 600001"

}

Response:

Json

{

"id": 2,

"name": "Srimathi Raju",

"email": "srimathi@example.com",

"contactNumber": "9876543210",

"address": "Chennai - 600001",

"role": "USER"

}