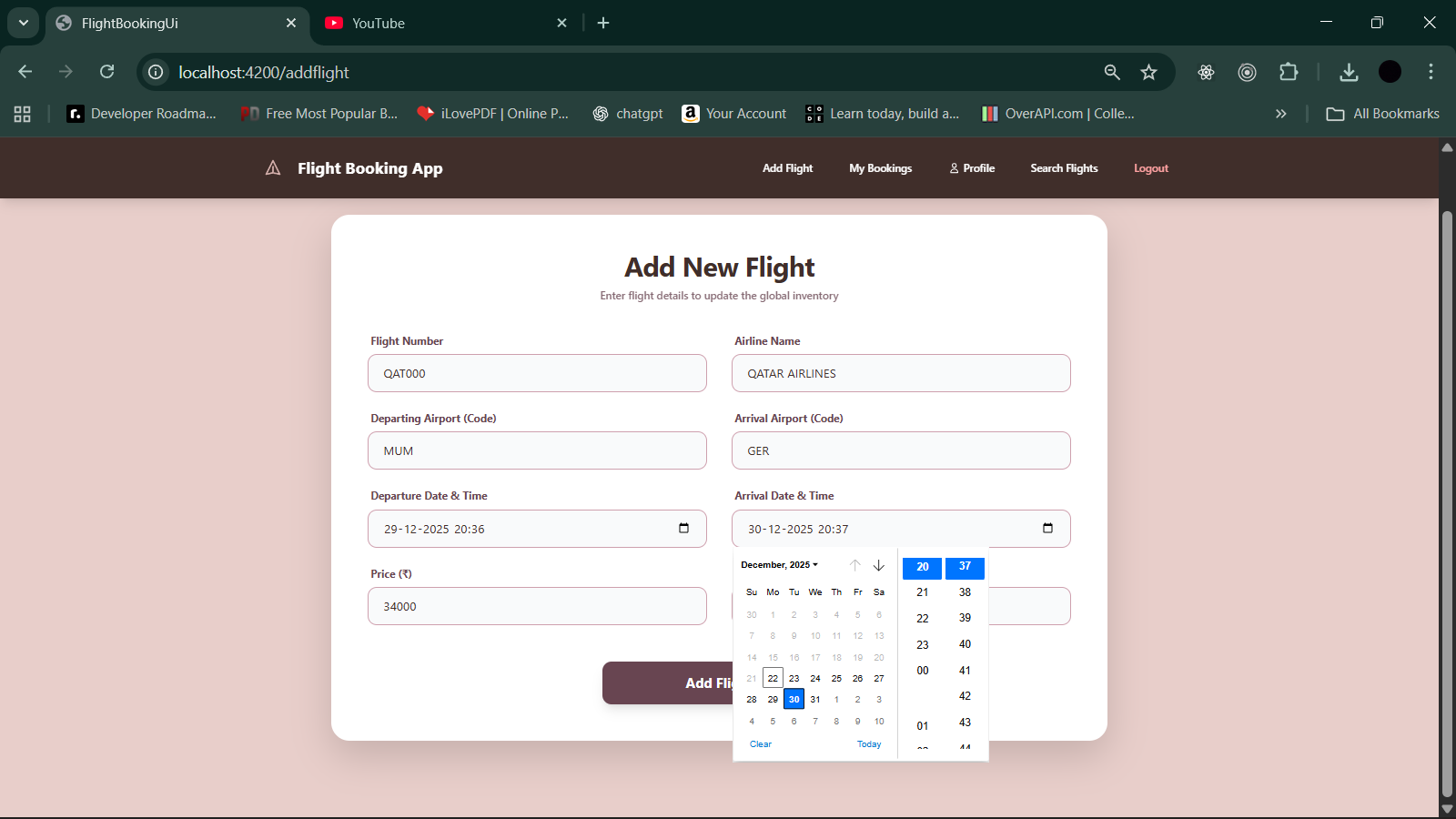
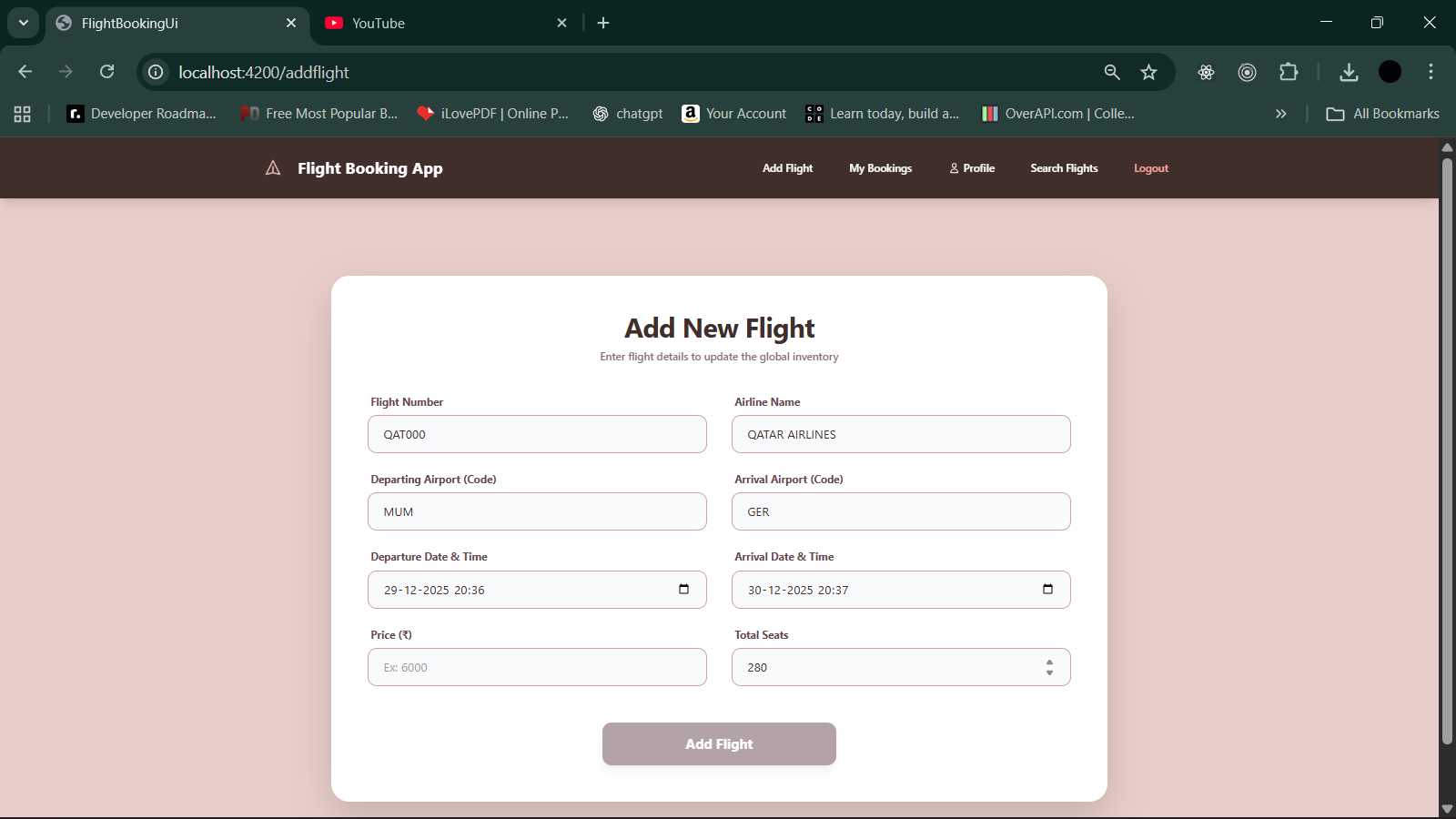
**Flight Booking  
Microservices App**

FRONTEND : Angular & Tailwind CSS

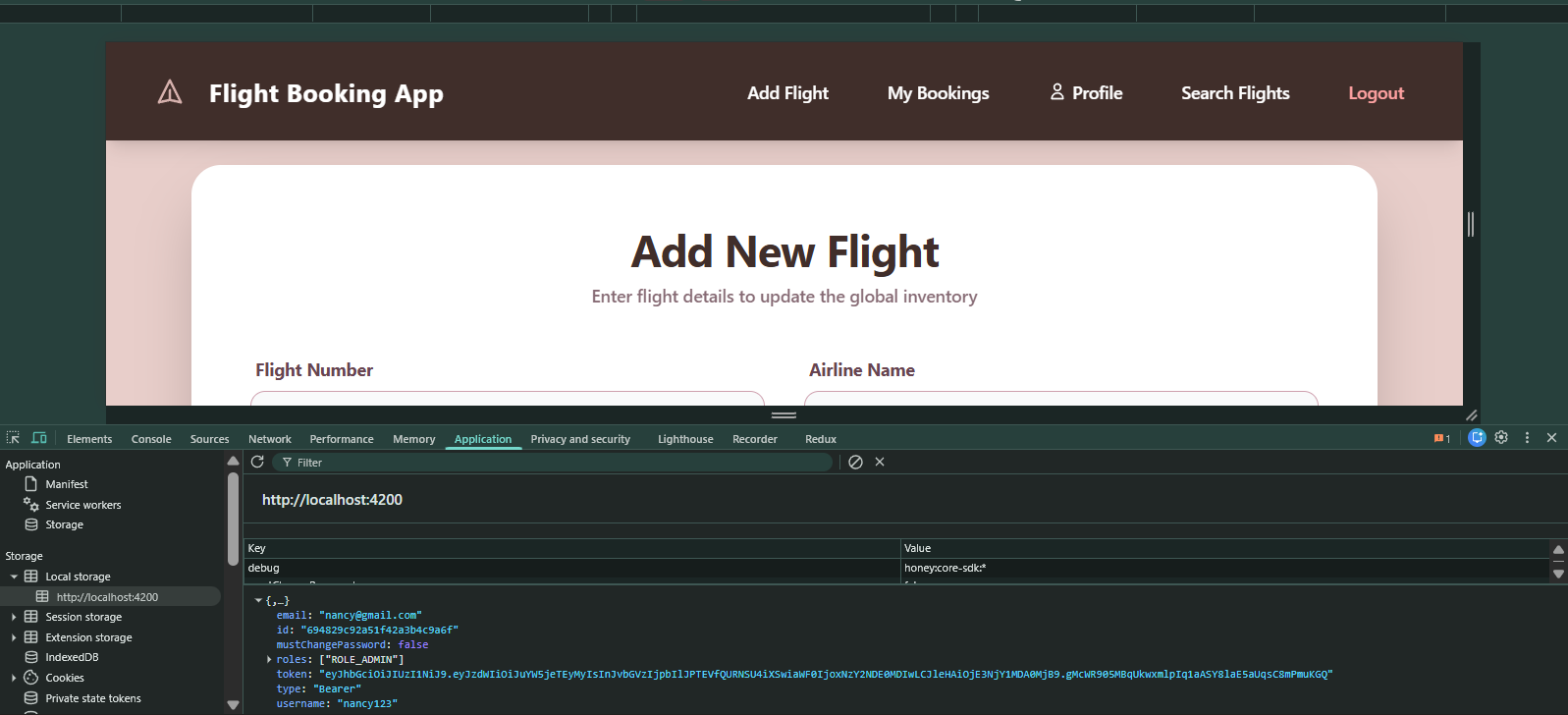
Add Flights

* Disabled past dates
* Disabled button until all required fields filled
* RBAC - only admin should be able to add flights





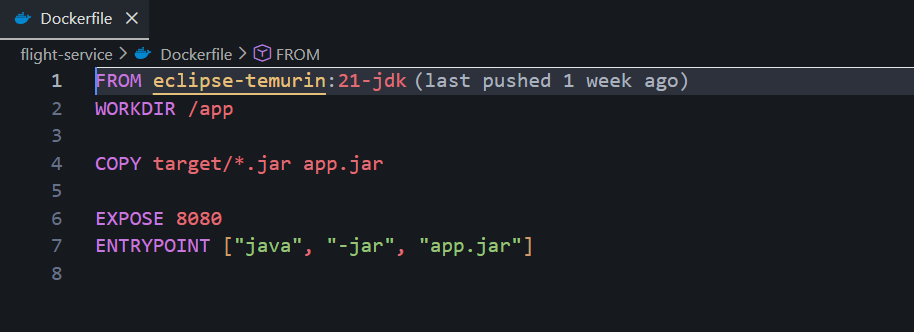
* Add flight option enabled only after admin login



Optimised to run all services from local :

* 2 properties files
* Implemented : start services.cmd



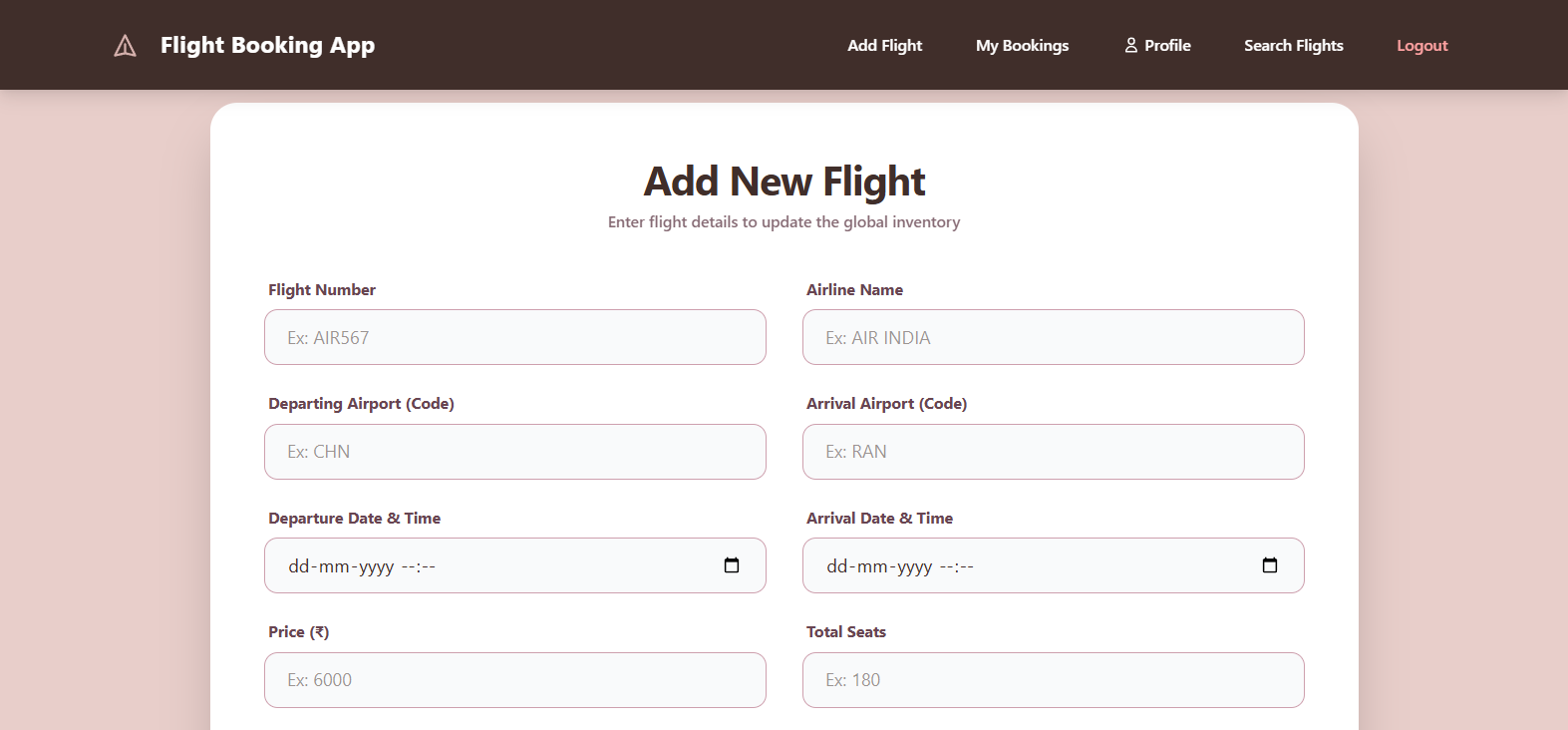


Post admin login :-

* Enabled add flight
* Attached wireframe png in github repo

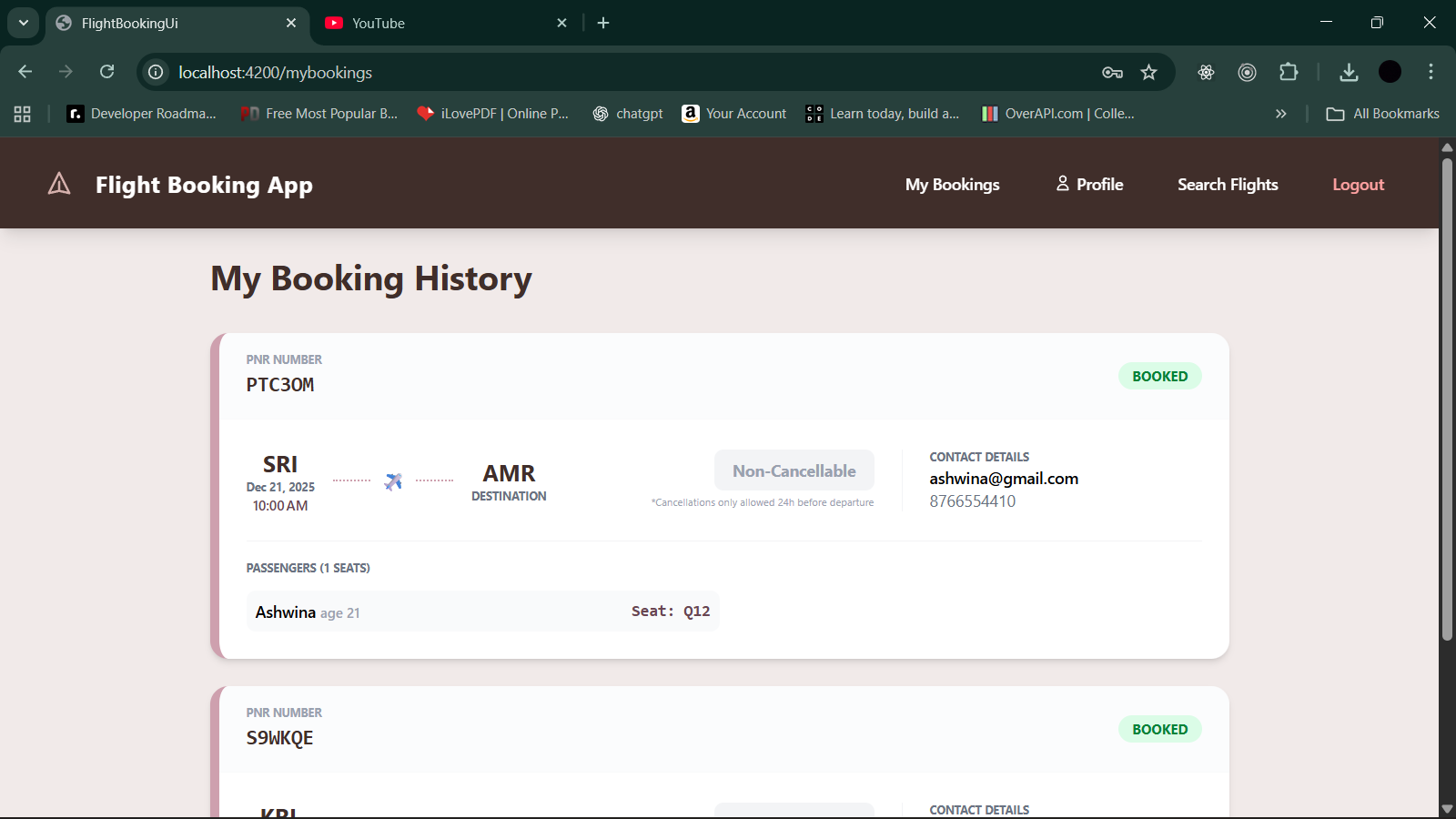
<https://github.com/NaynaBisht/FlightBooking_Microservices_App/blob/main/add-flight%20wireframe.png>

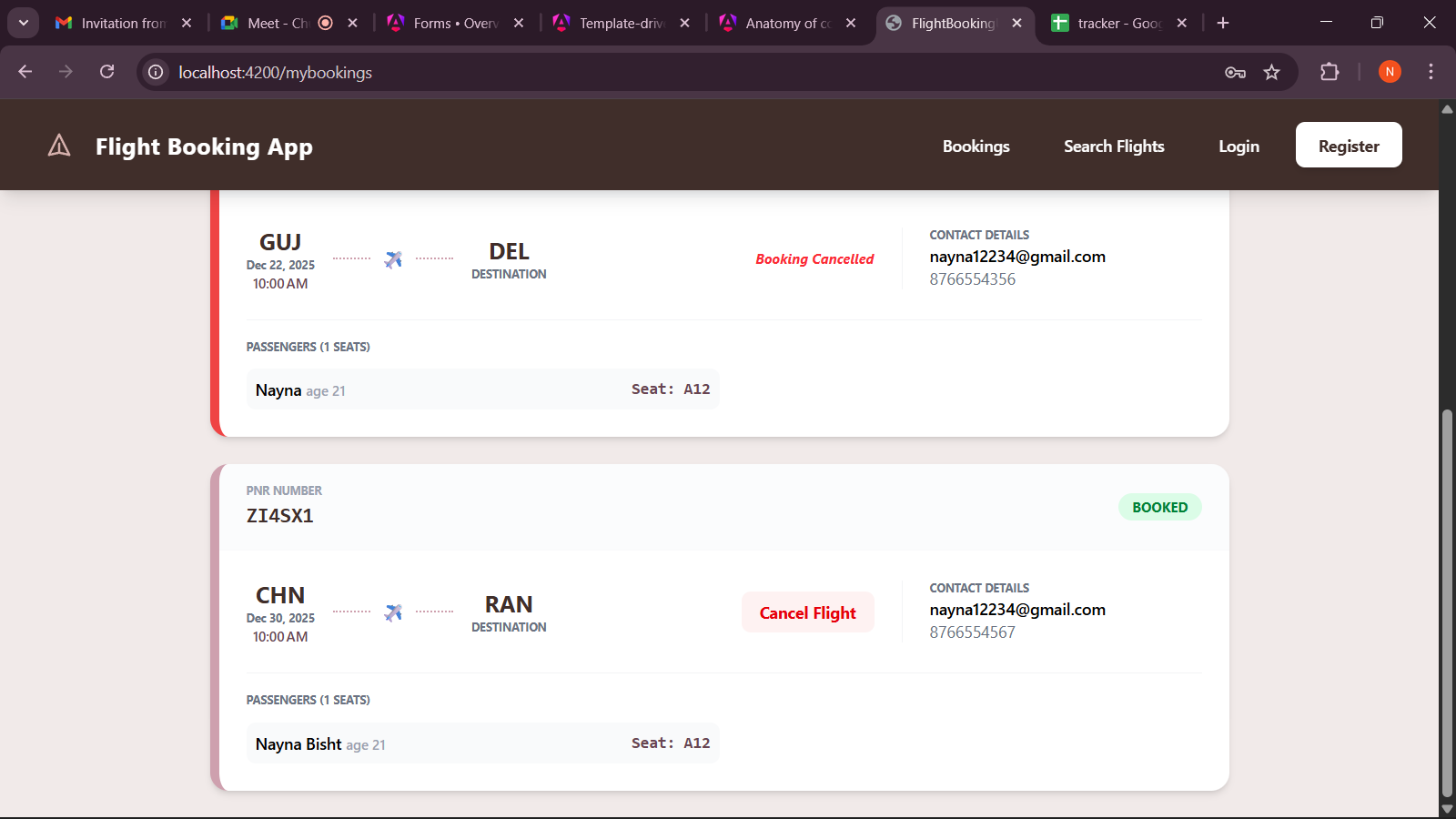


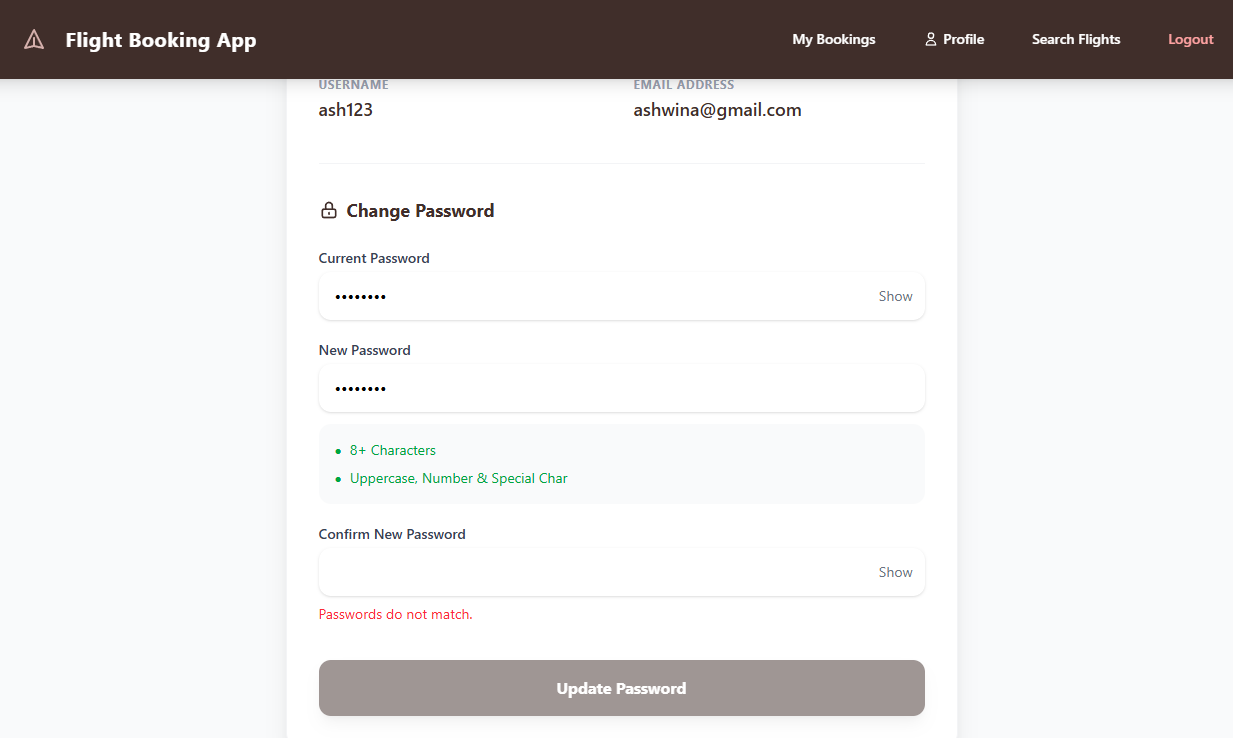


Post user login : –

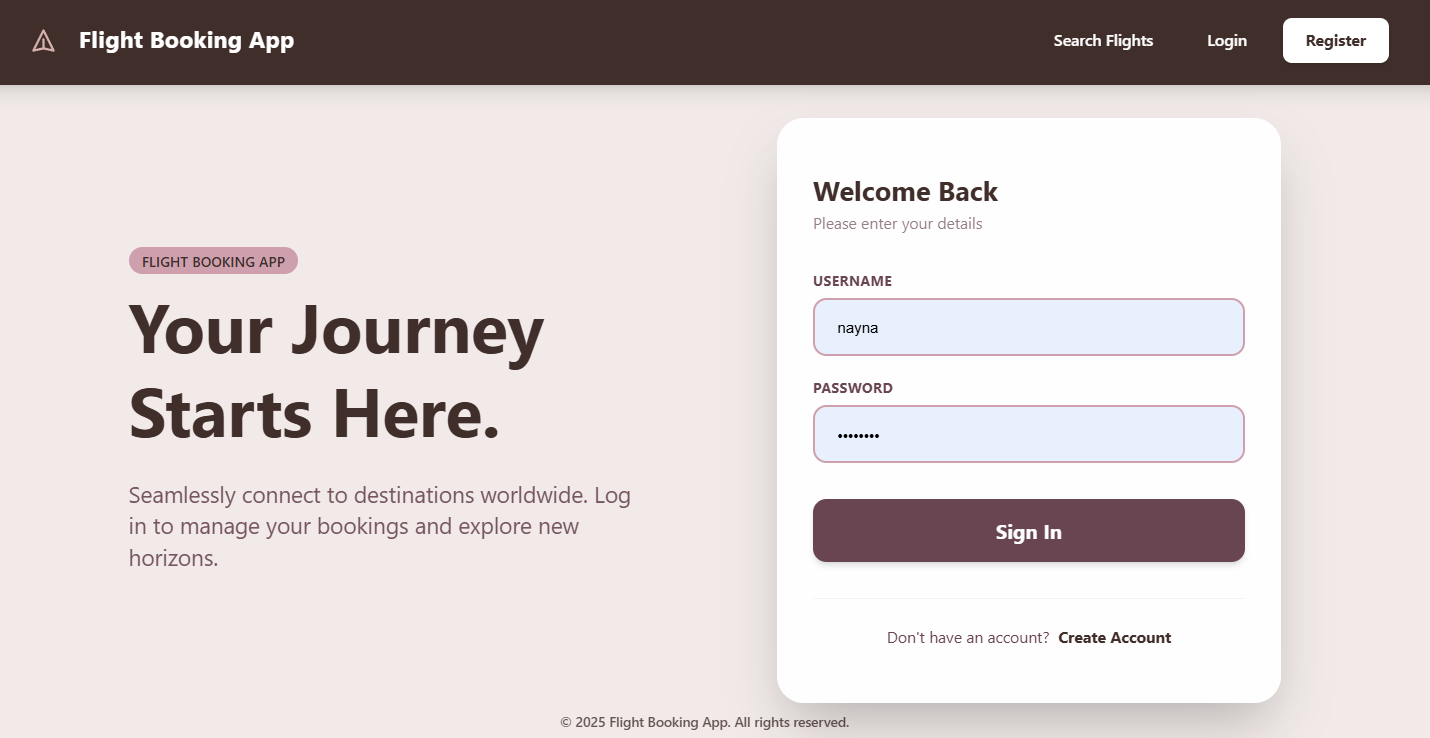
* booking history enabled
* cancel booking buttons
* non cancellable within 24 hrs
* update password in profile

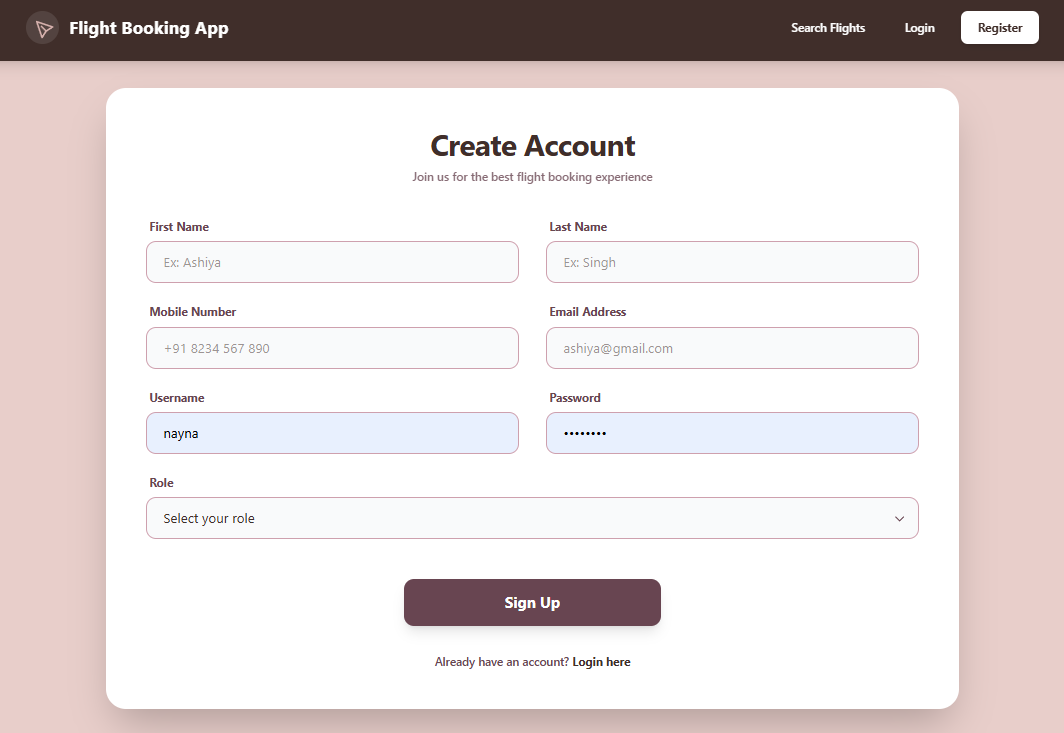






SignIn & Signup Page :

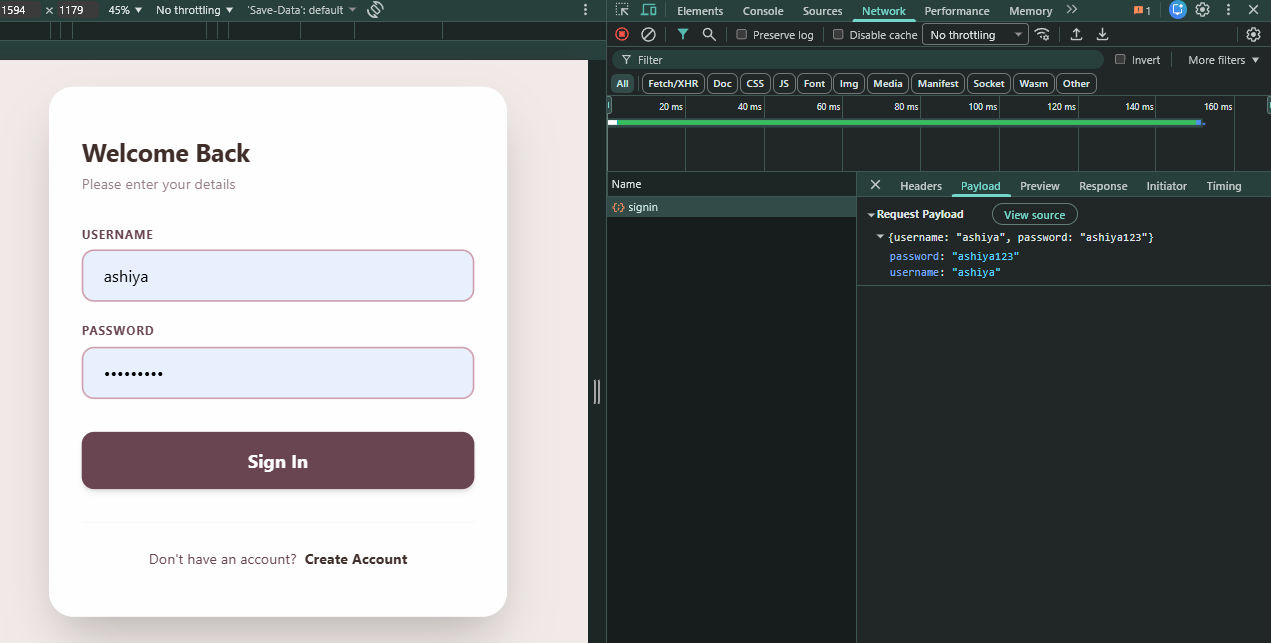




Register User/Admin :



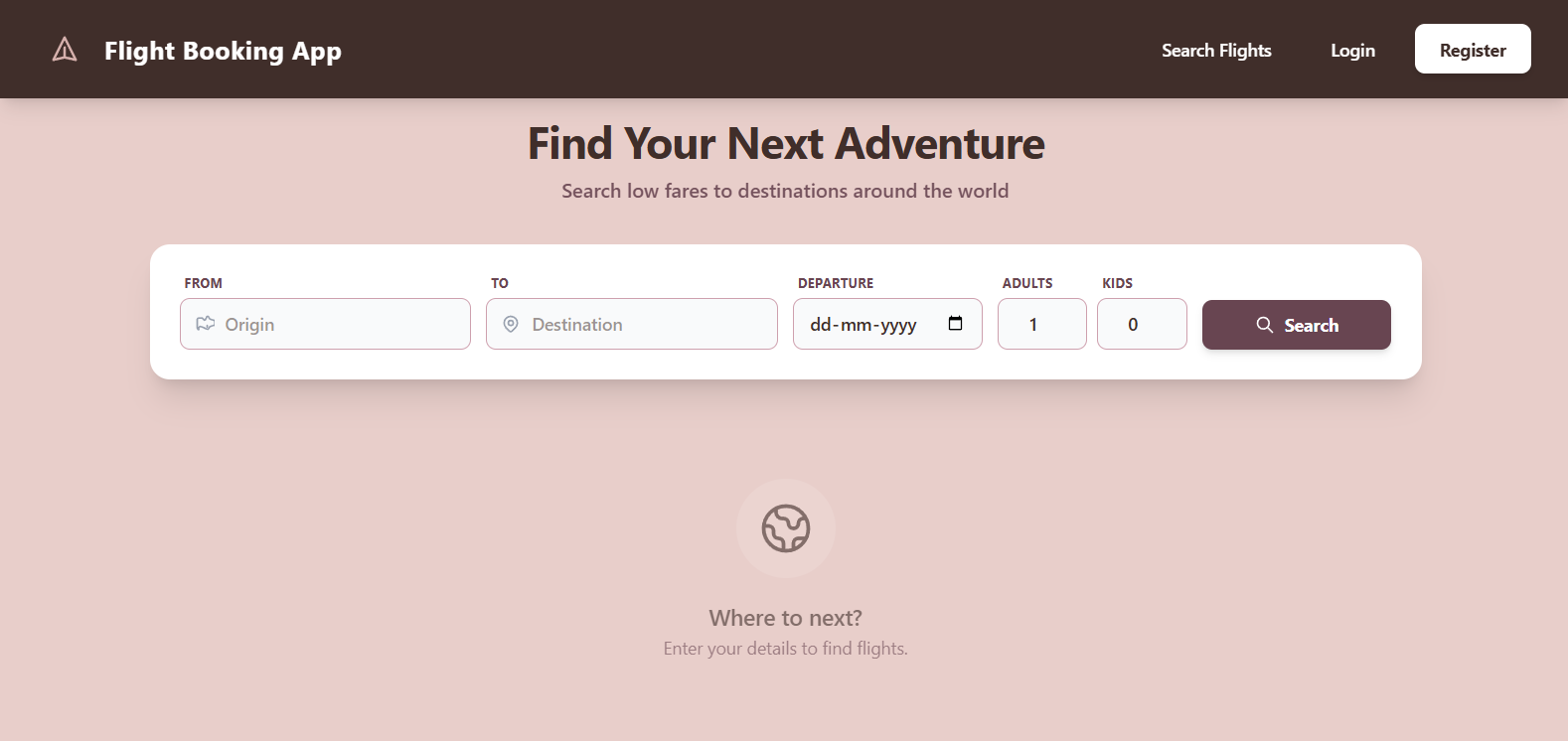
Login User/Admin successful :

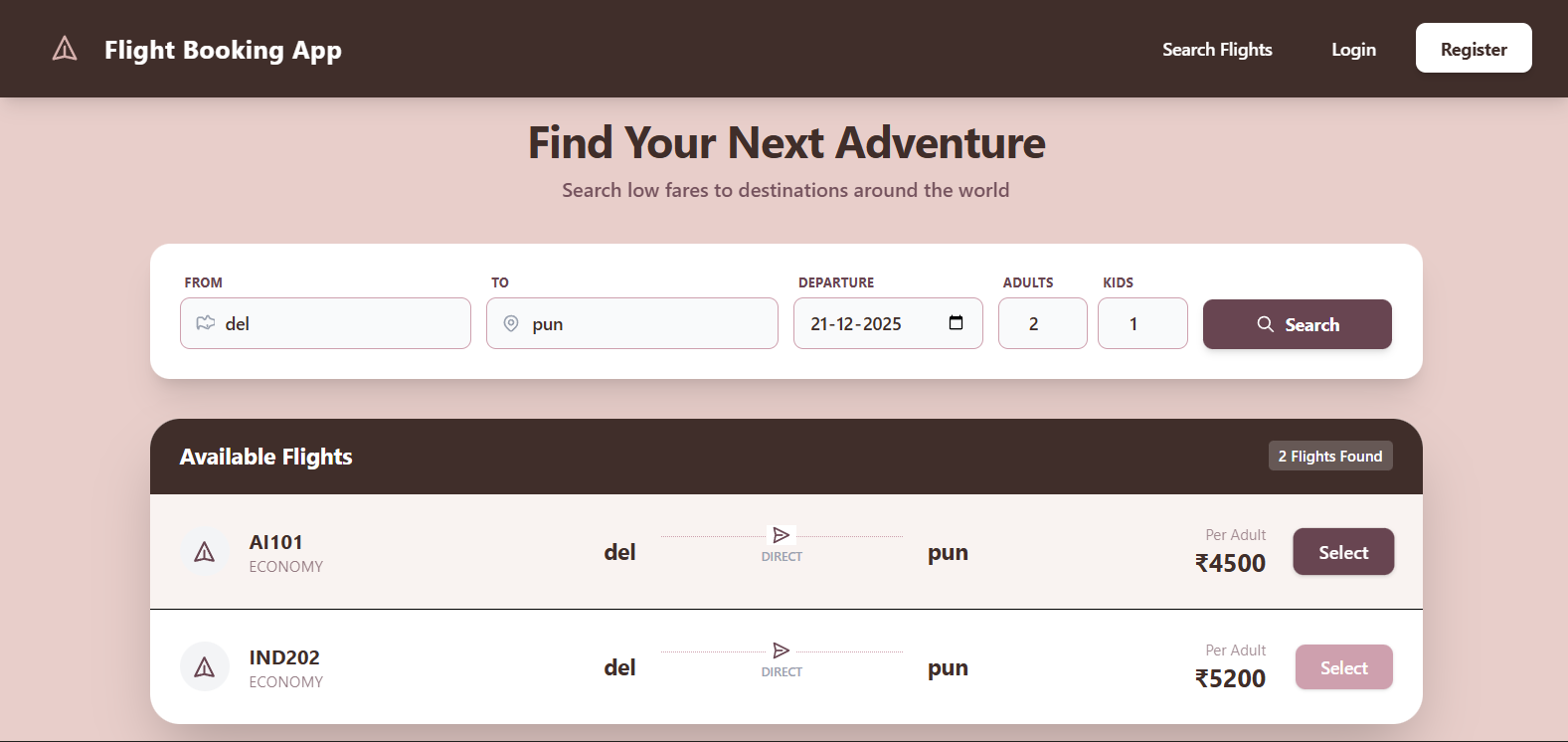


JWT Token Stored in local storage :

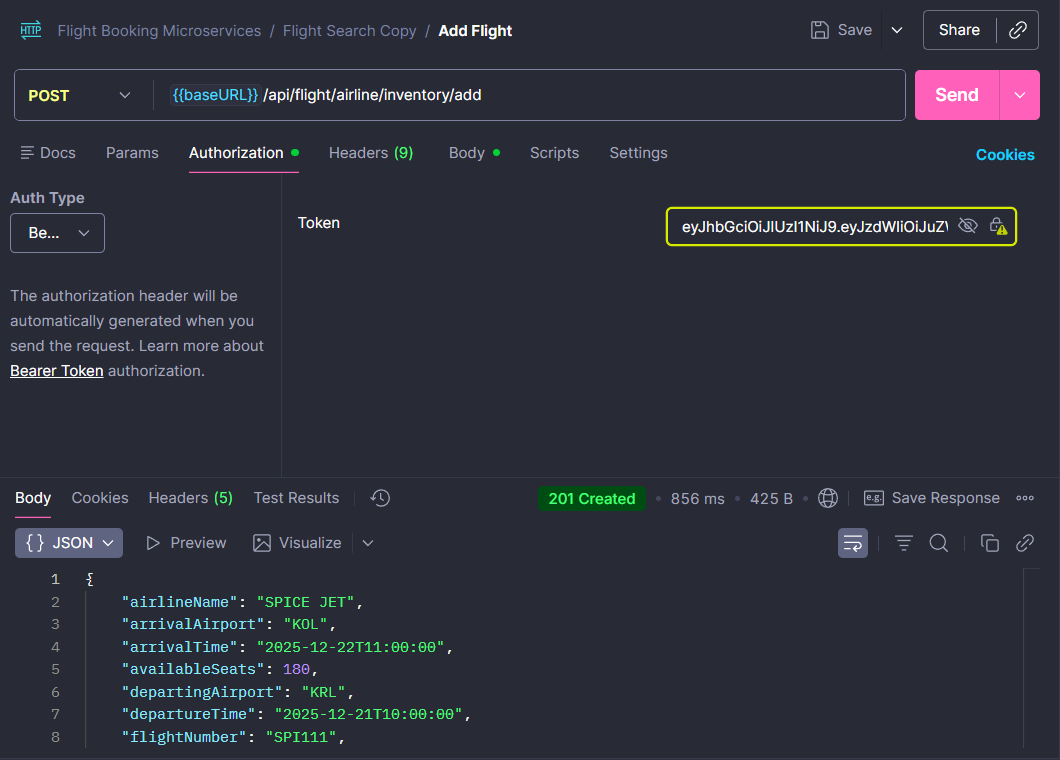


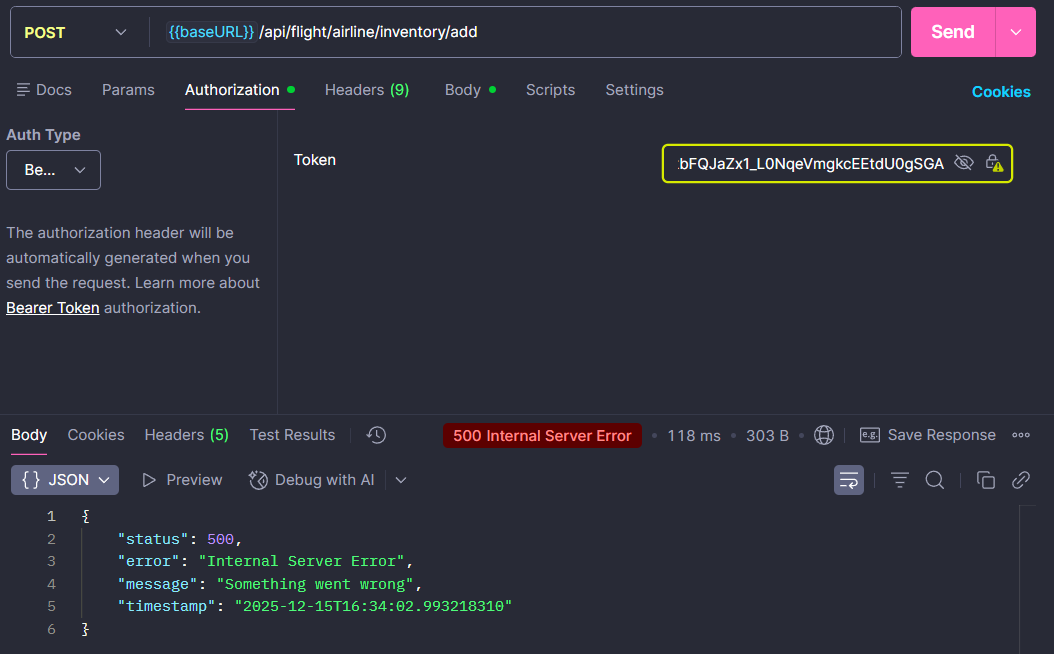
Flight Search Page :

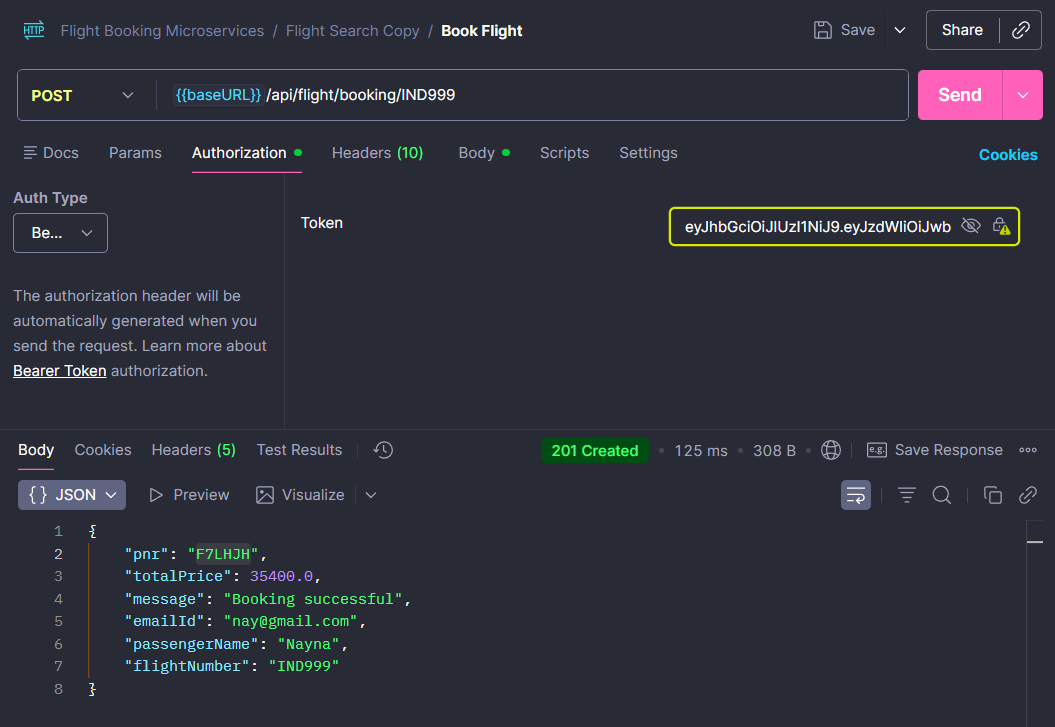


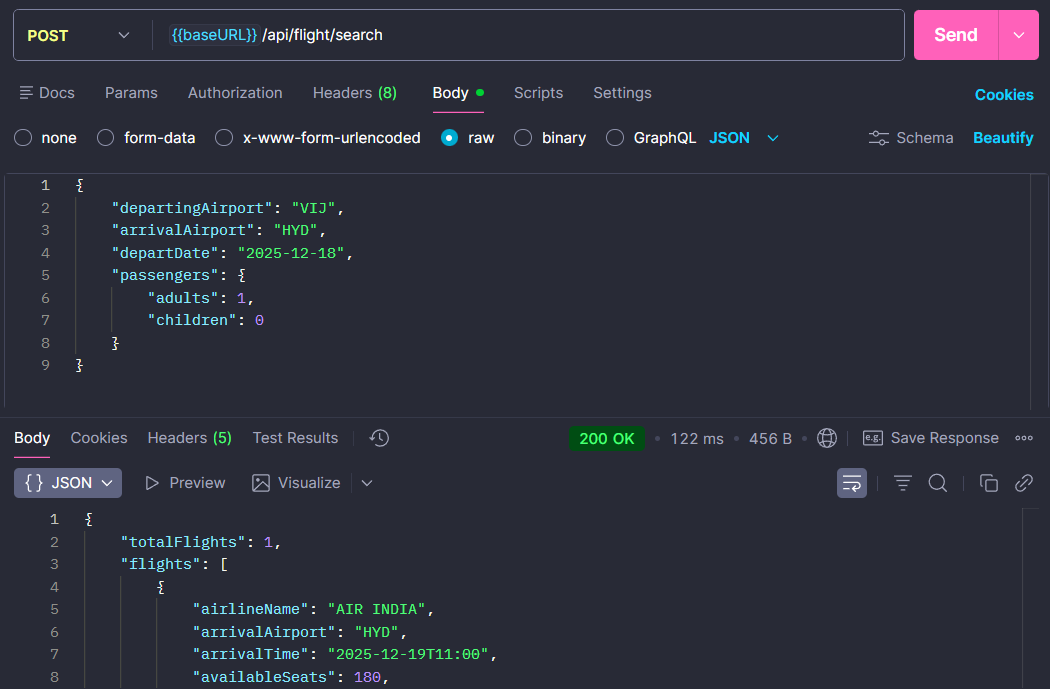


Admin token required to add flight – private url :

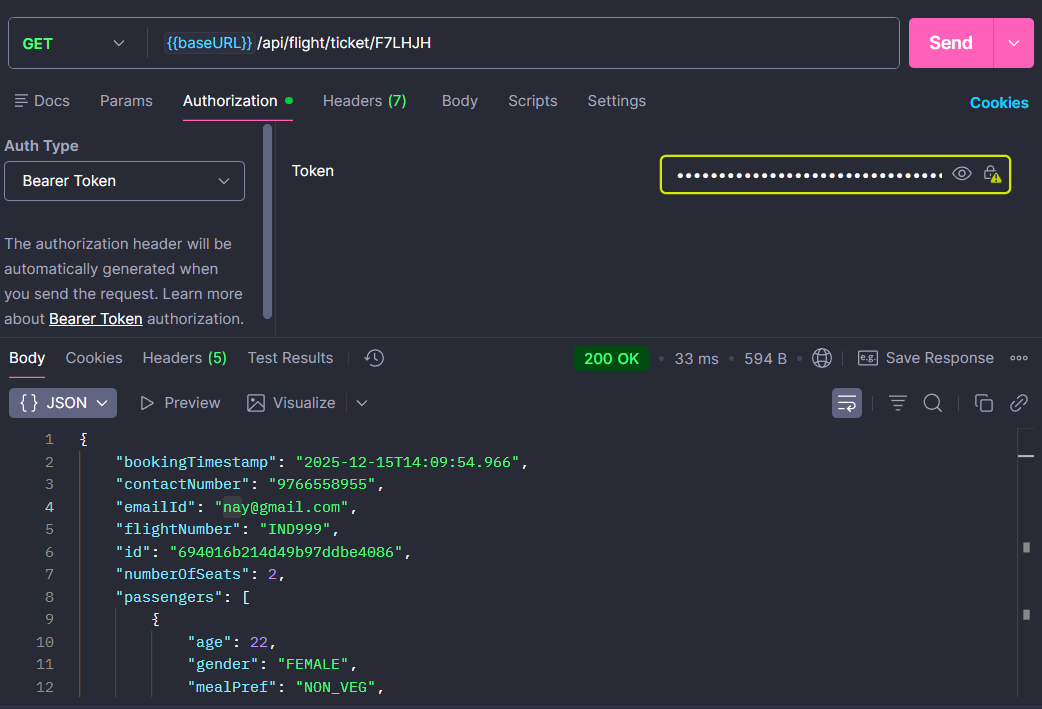


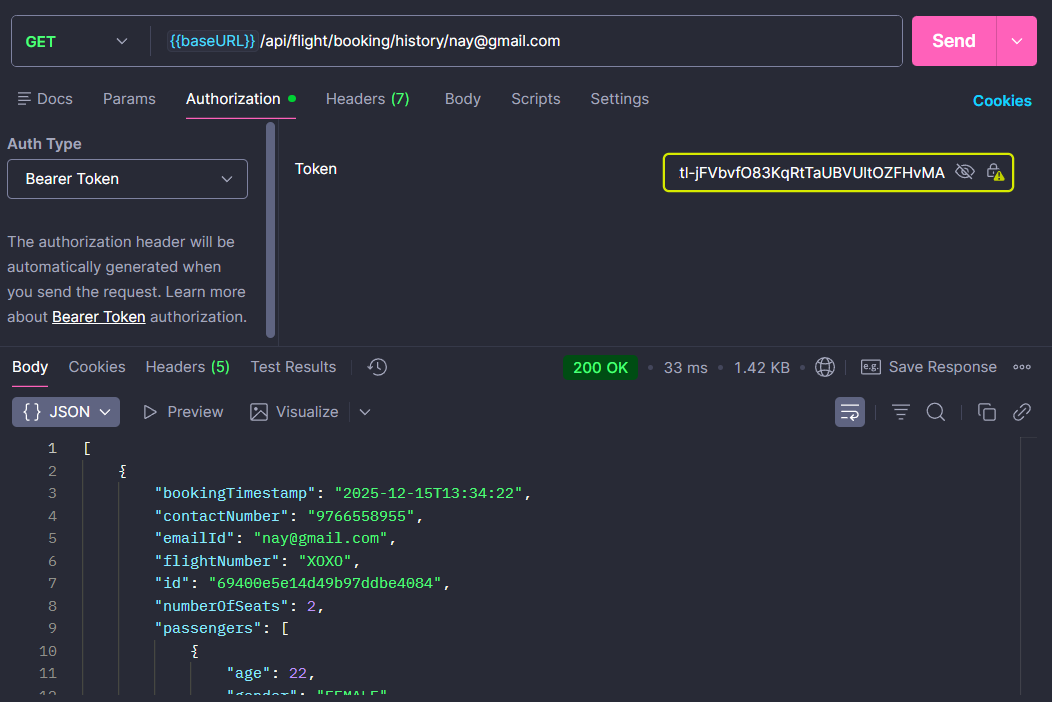
Failed – when used user token :  
 User token required for flight booking – private url :



Search flights – public url endpoint  


User token required to fetch details :



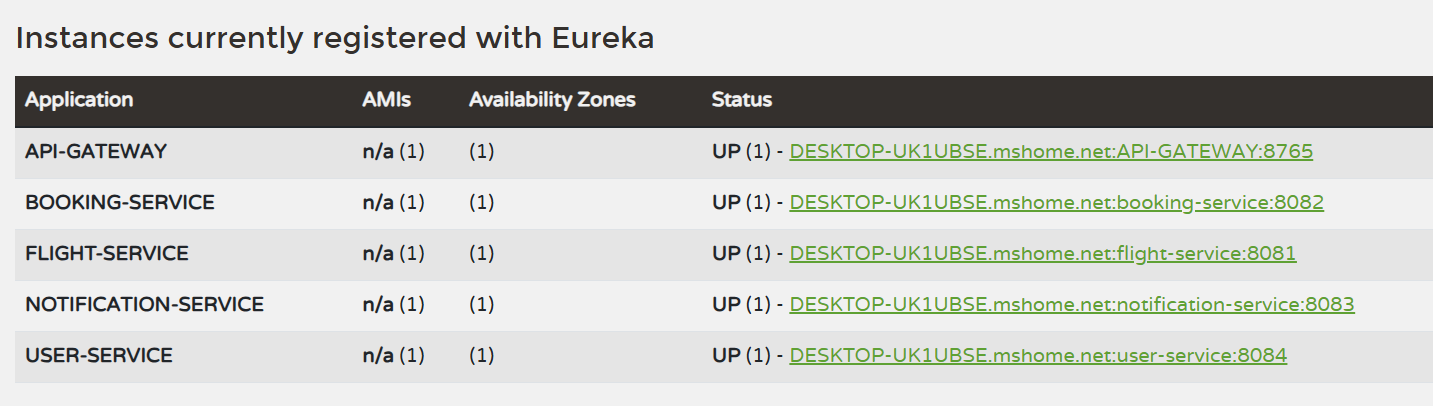


Microservices Implemented

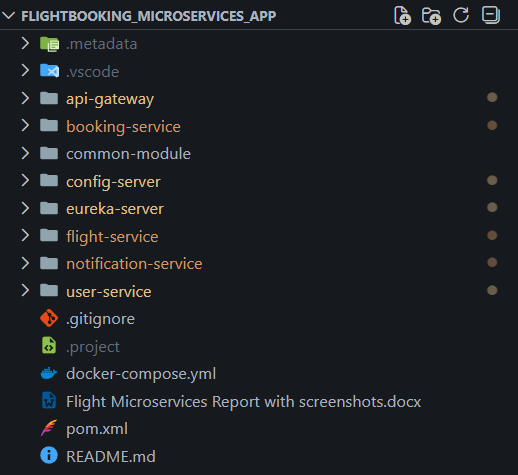
* Flight Service – Manages flight operations and search functionality
* Booking Service – Handles reservations, cancellations, and passenger details
* Notification Service – Sends email notifications via RabbitMQ
* API Gateway – Central entry point for routing and load balancing
* Eureka Server – Service discovery and registration
* Config Server – Centralized configuration management
* Common Module – Shared DTOs and exception handling

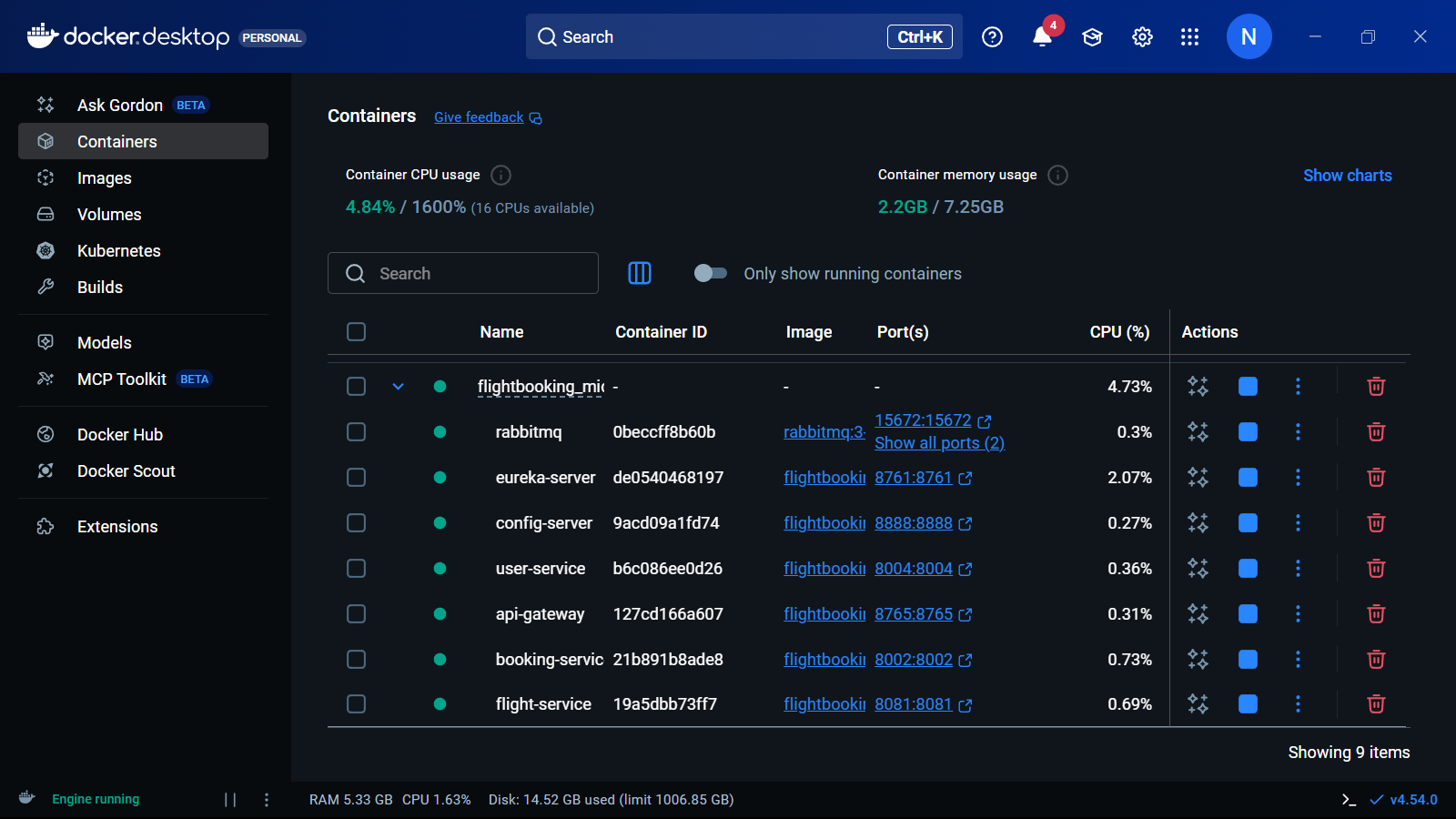
Core Integrations & Features

Eureka Server Setup :

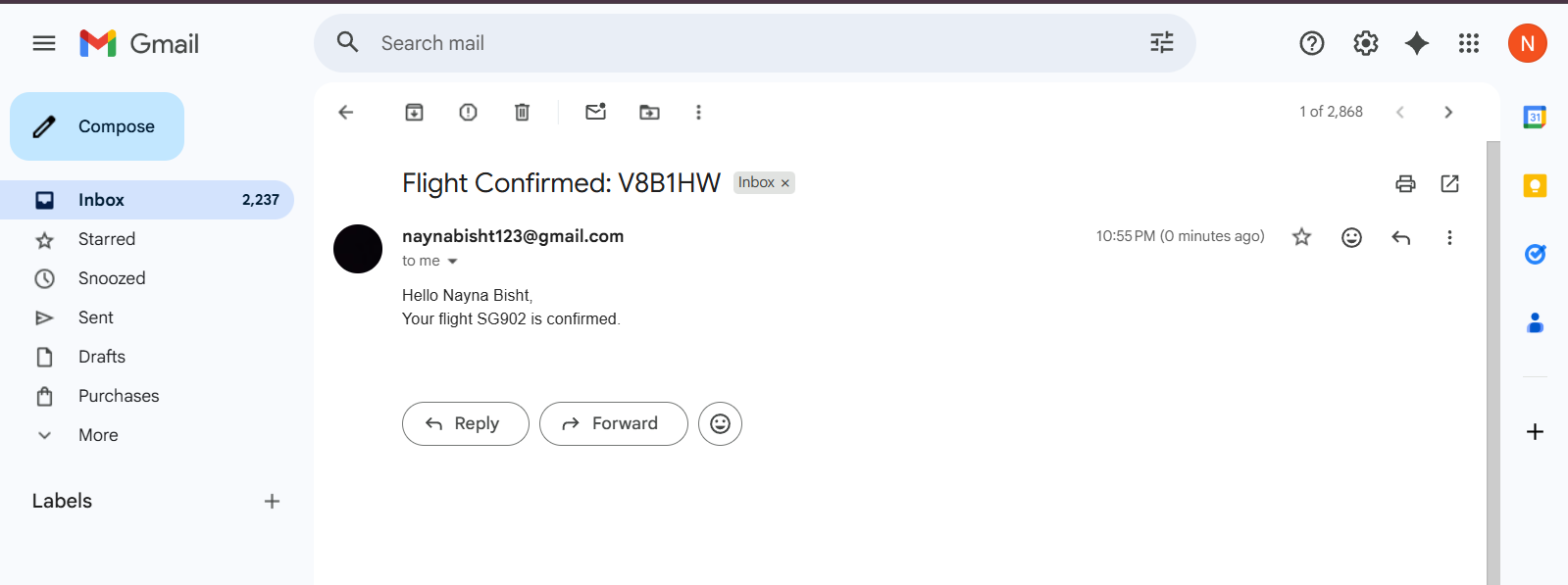


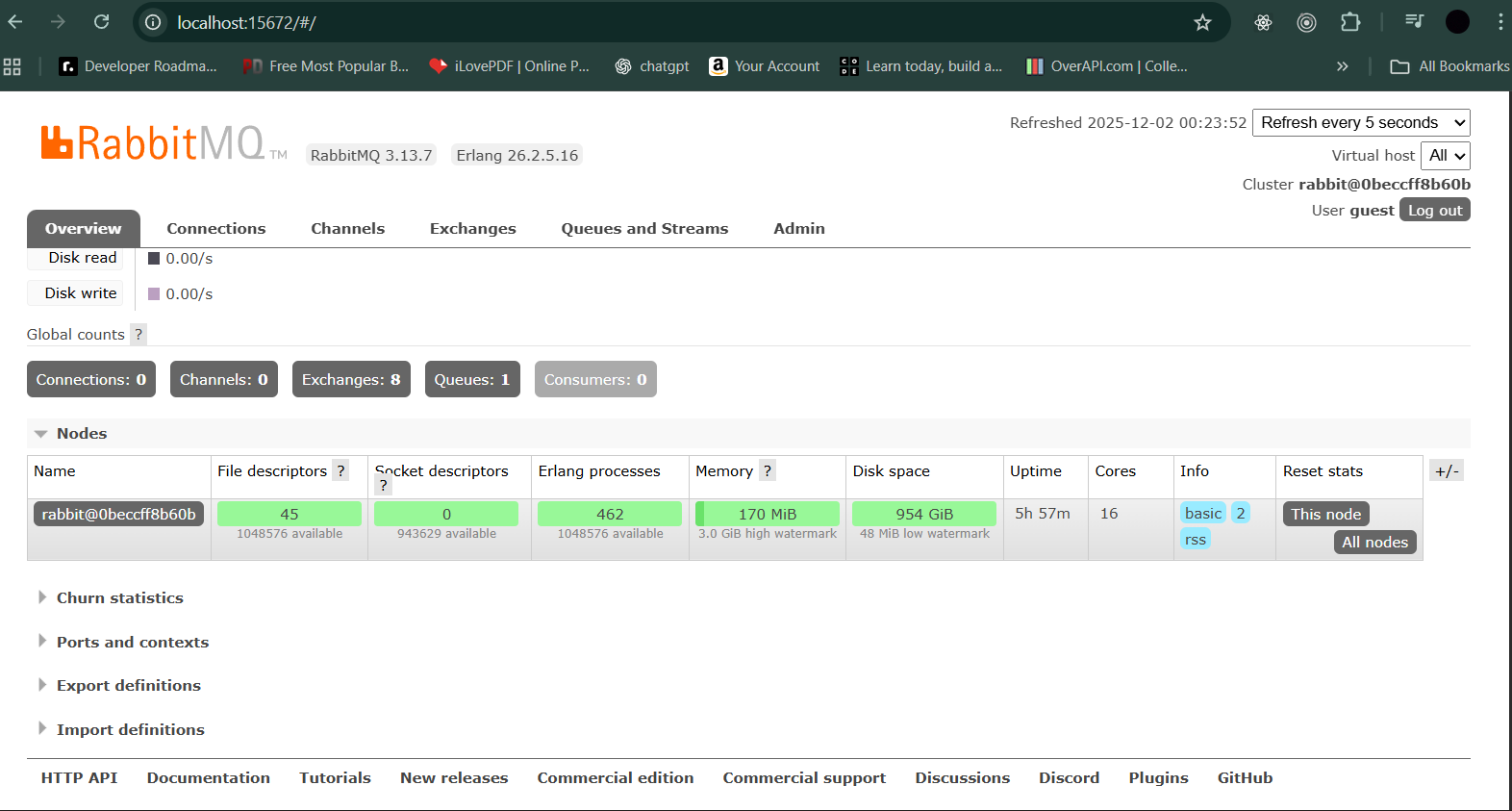
Docker Containers – single docker compose file:



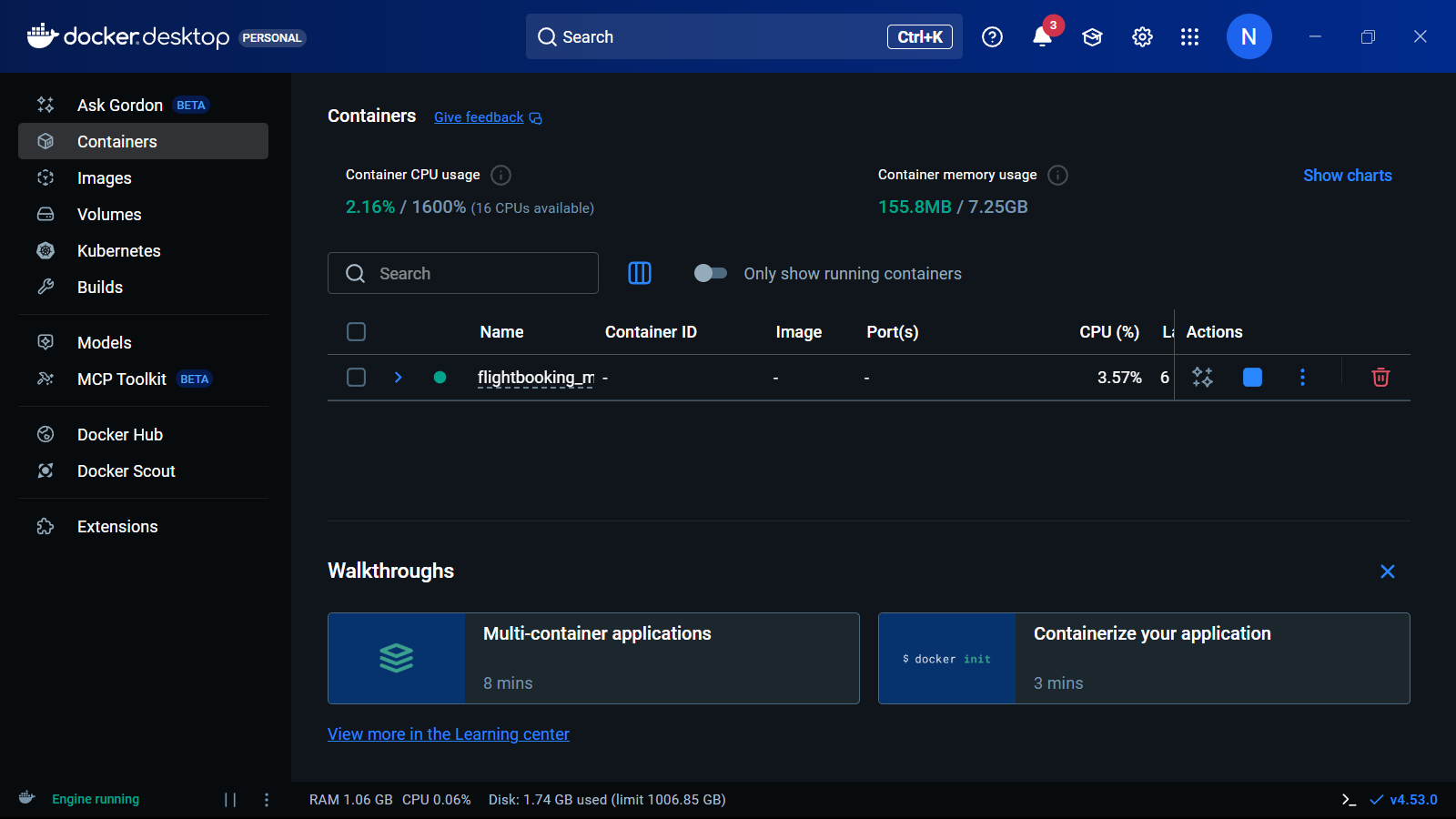


\*\*Docker Logs File Added in repo\*\*

RabbitMQ Message Broker – Processes asynchronous email notifications  


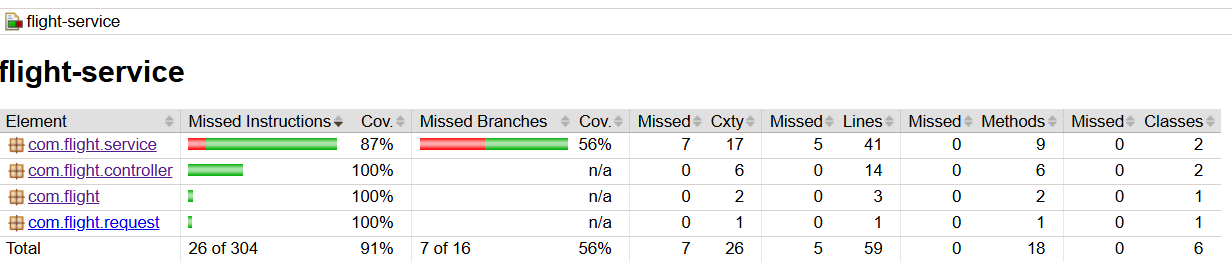


Docker Container for RabbitMQ – Containerized message broker setup

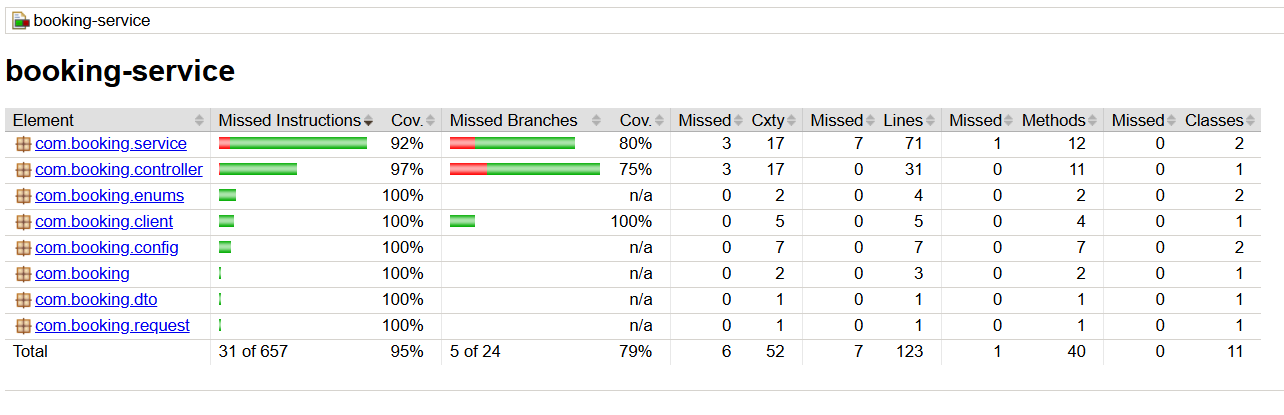


Jacoco Code Coverage Report – Test coverage analysis

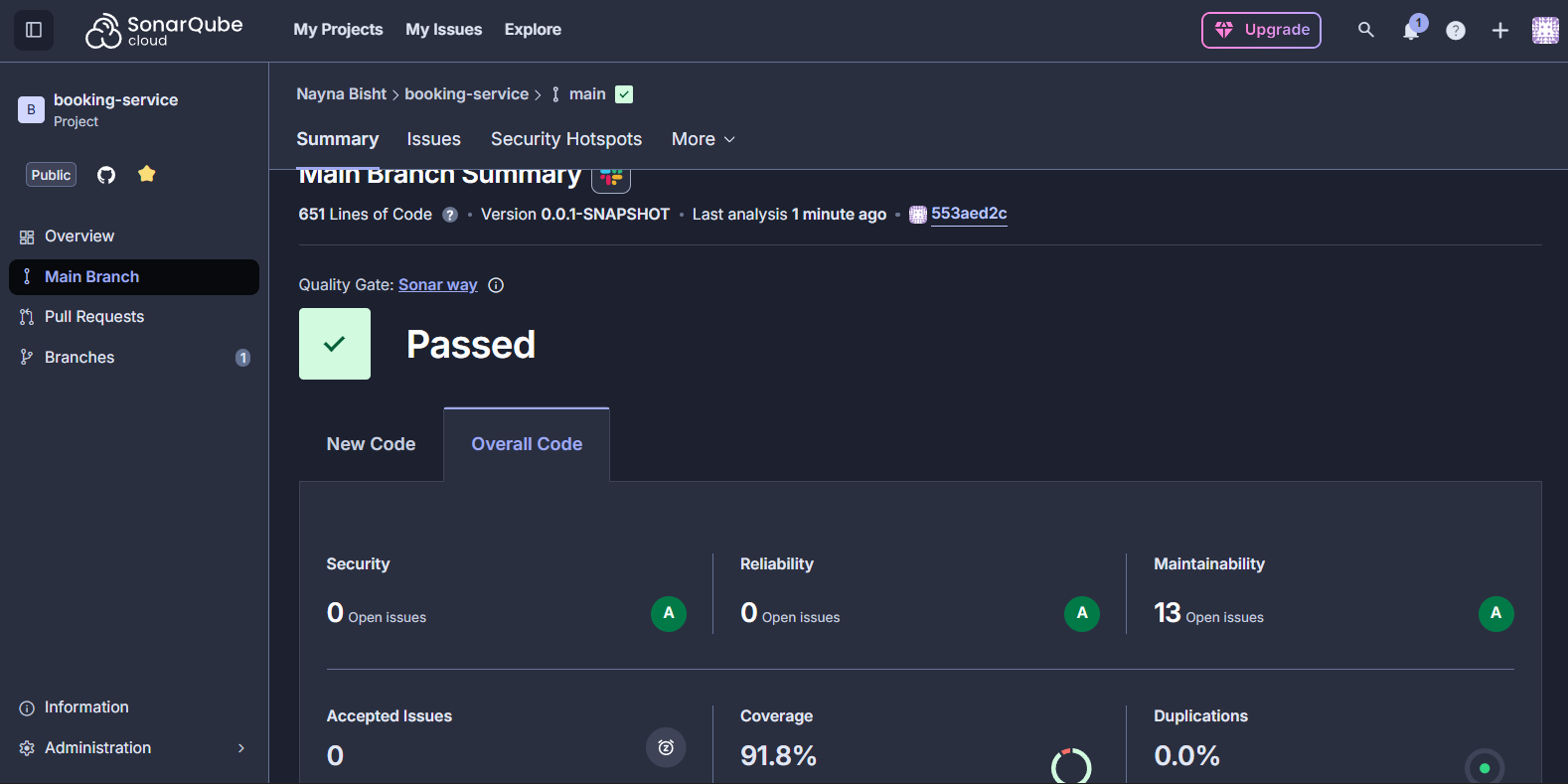
Flight Service :-



Booking Service:-

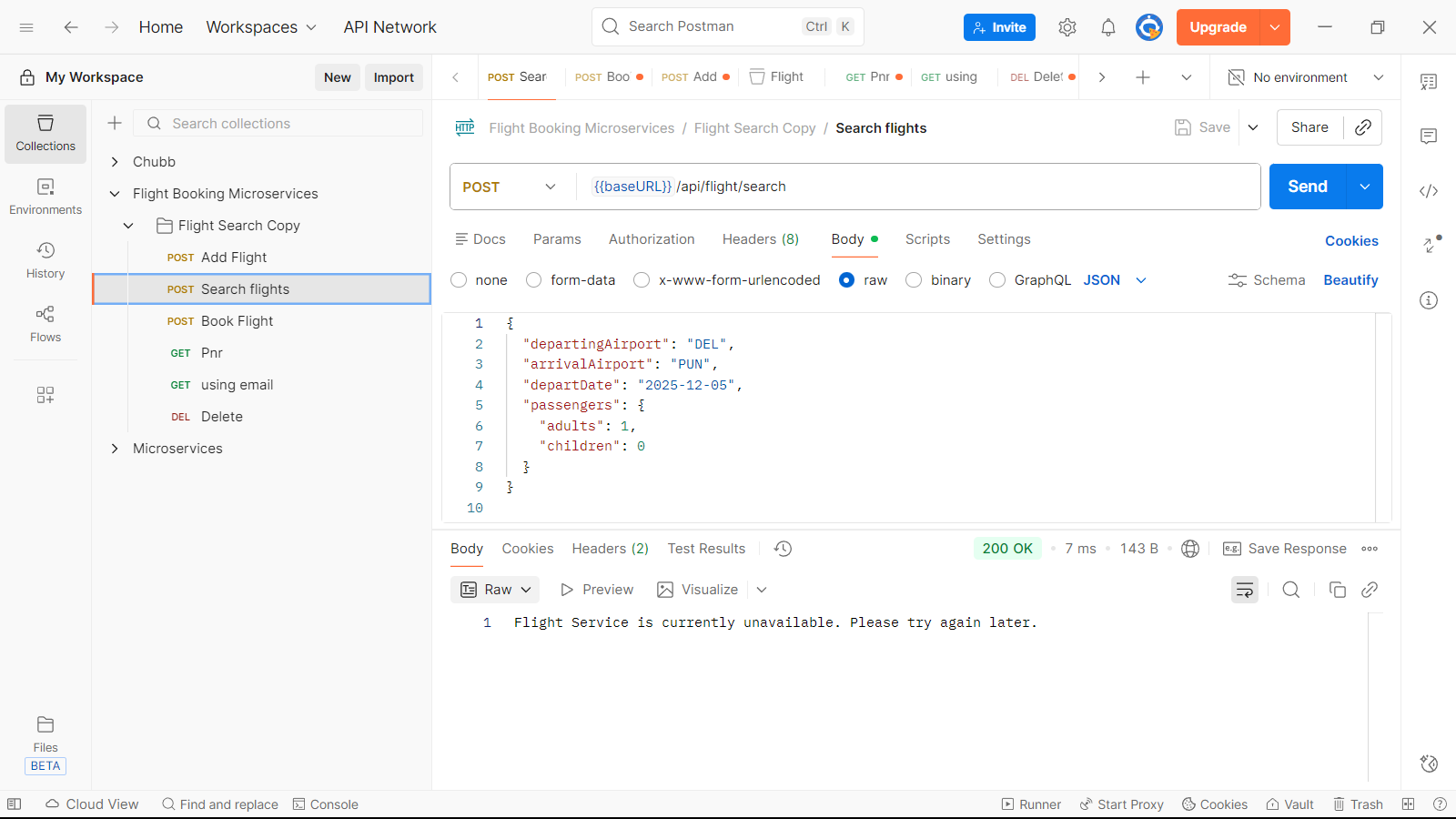


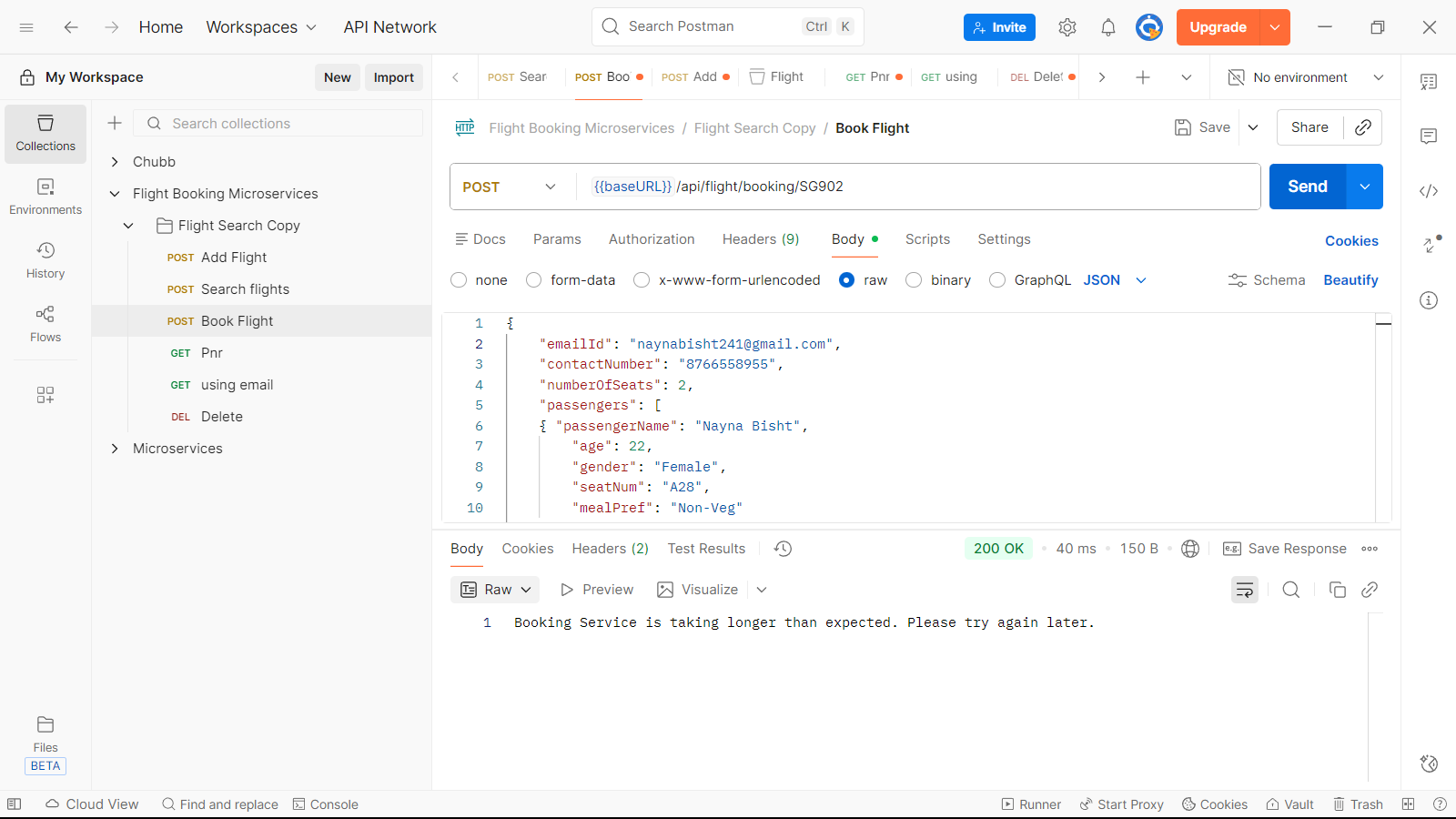
SonarQube Quality & Coverage Report – Code quality and maintainability checks



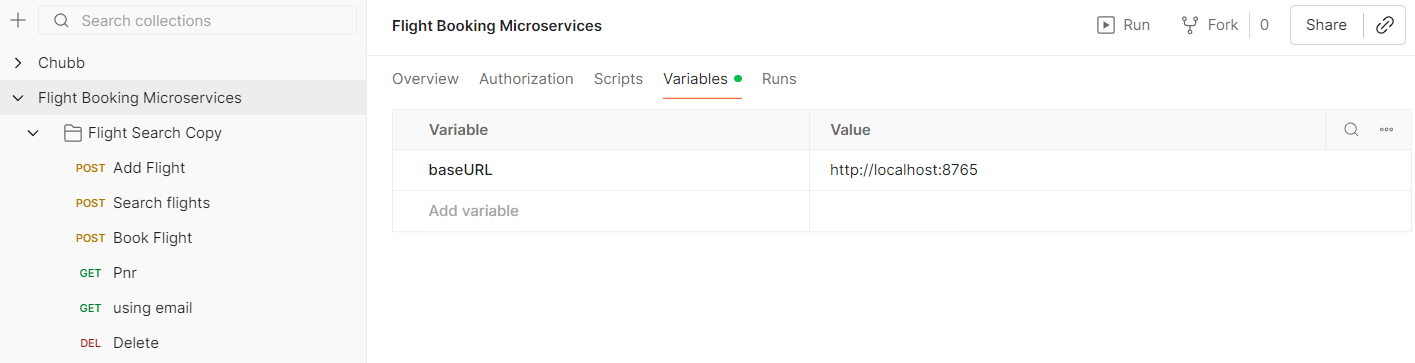
Circuit Breaker (Resilience4j)

Flight Service :



Booking Service :  


Api Gateway : baseURL - <http://localhost:8765>

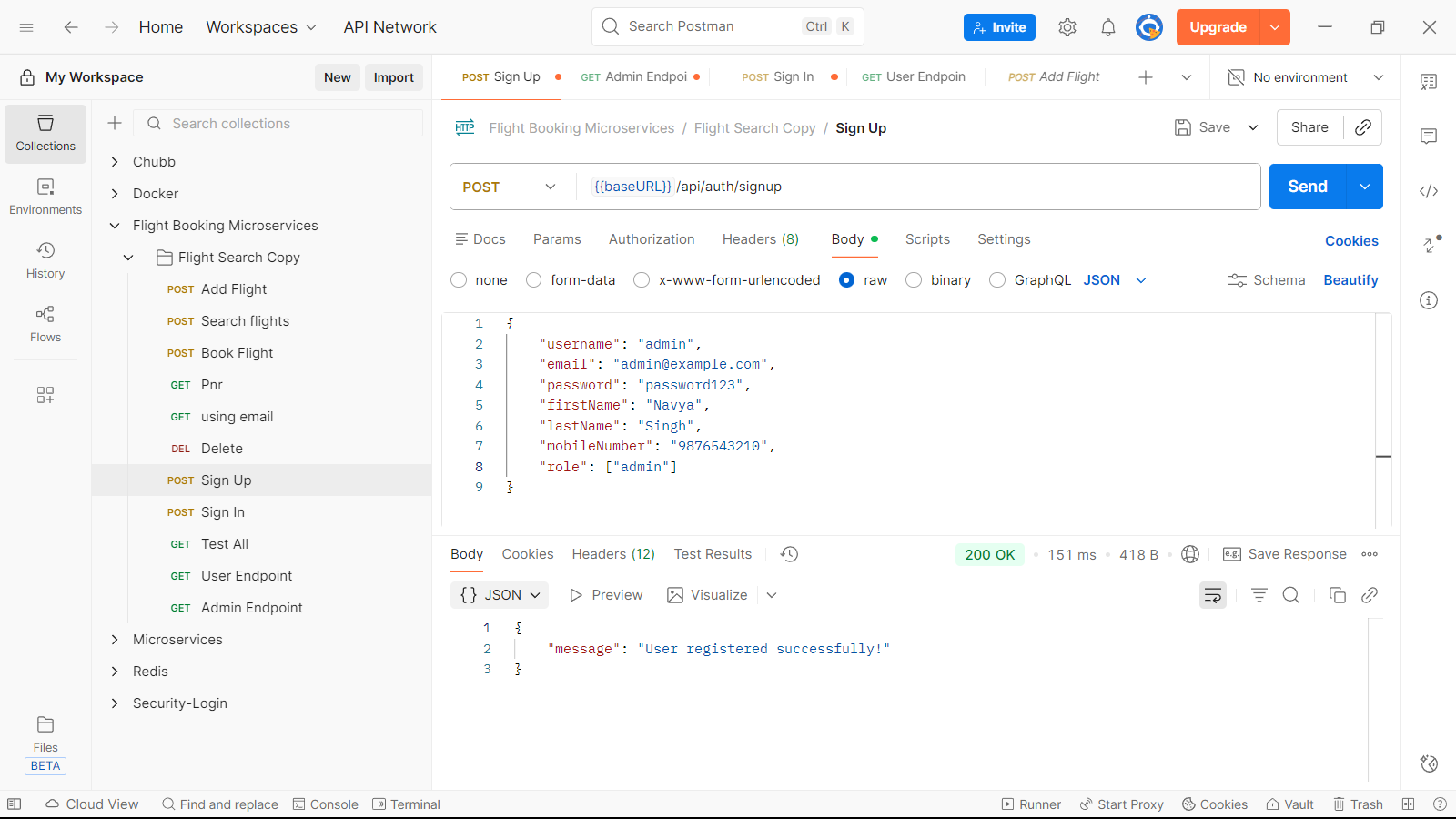


Functional Features

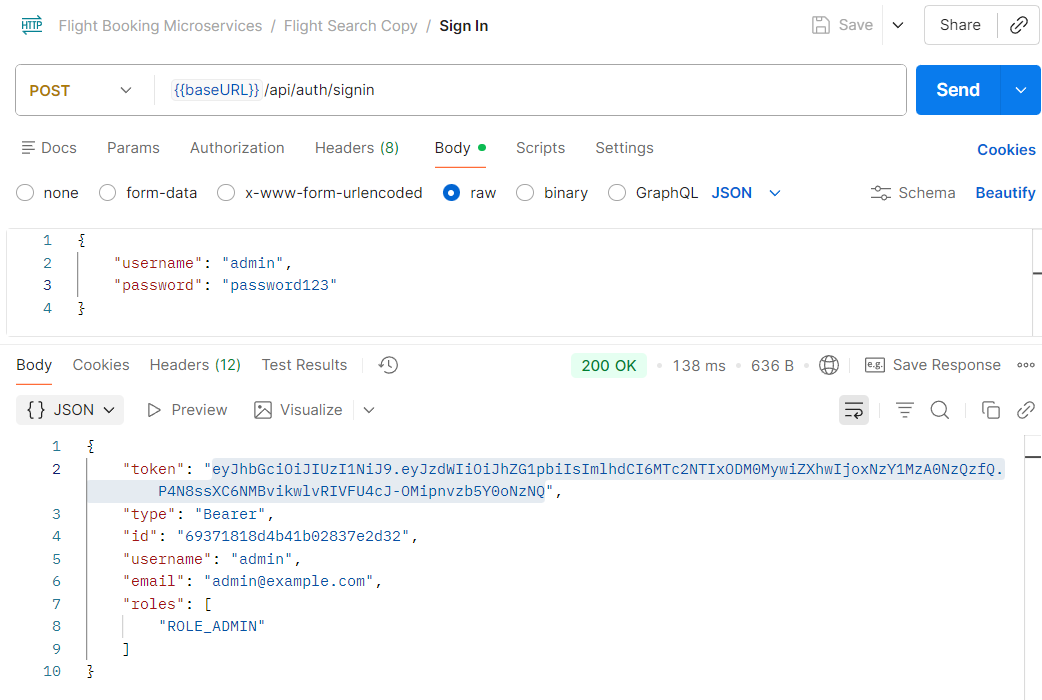
* Add Flight – Admin ability to create flights
* Search Flight – Query flights based on route and date
* Book Flight – End-to-end booking flow with validation
* Cancel Booking – Modify booking status and release resources
* Negative Testing Scenarios – Validation of bad inputs & error flows

Authentication & Security

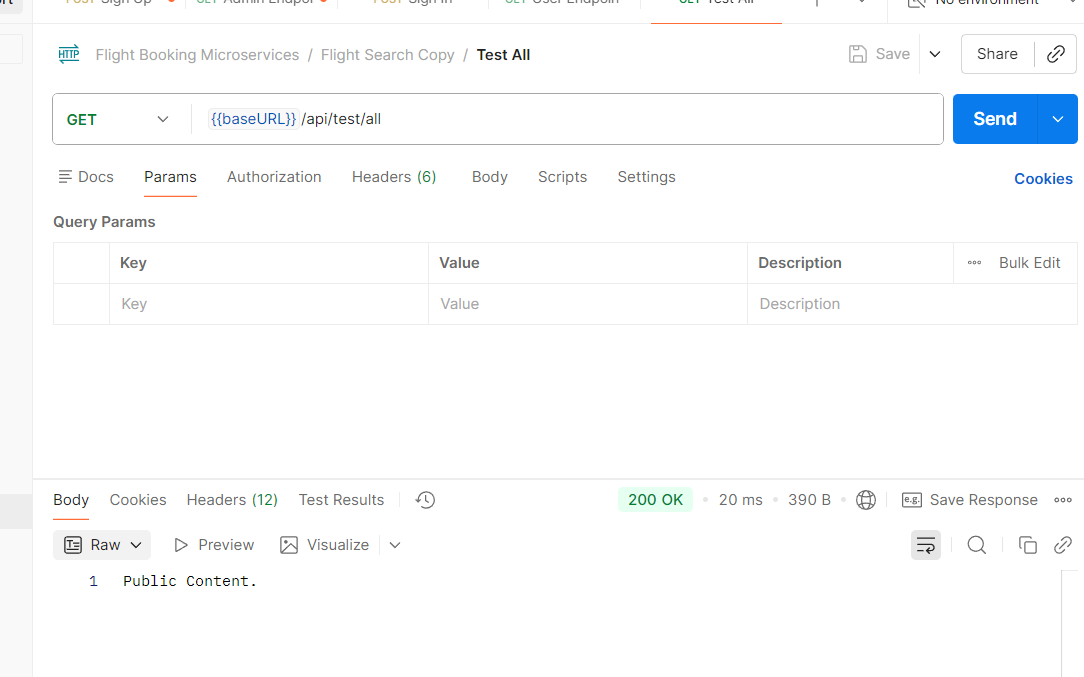
User Signup :



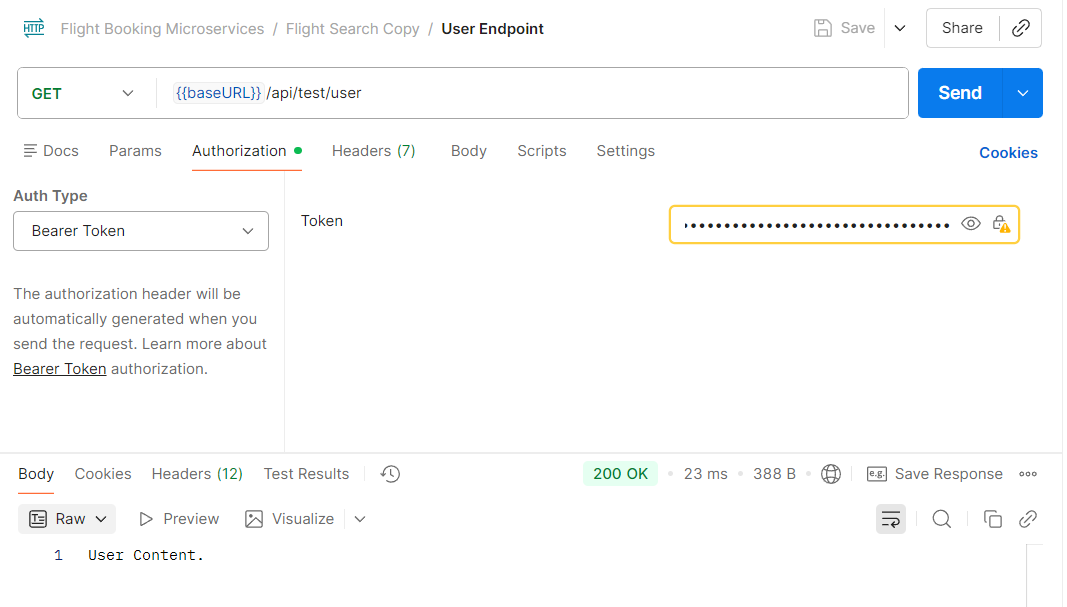
User Signin :



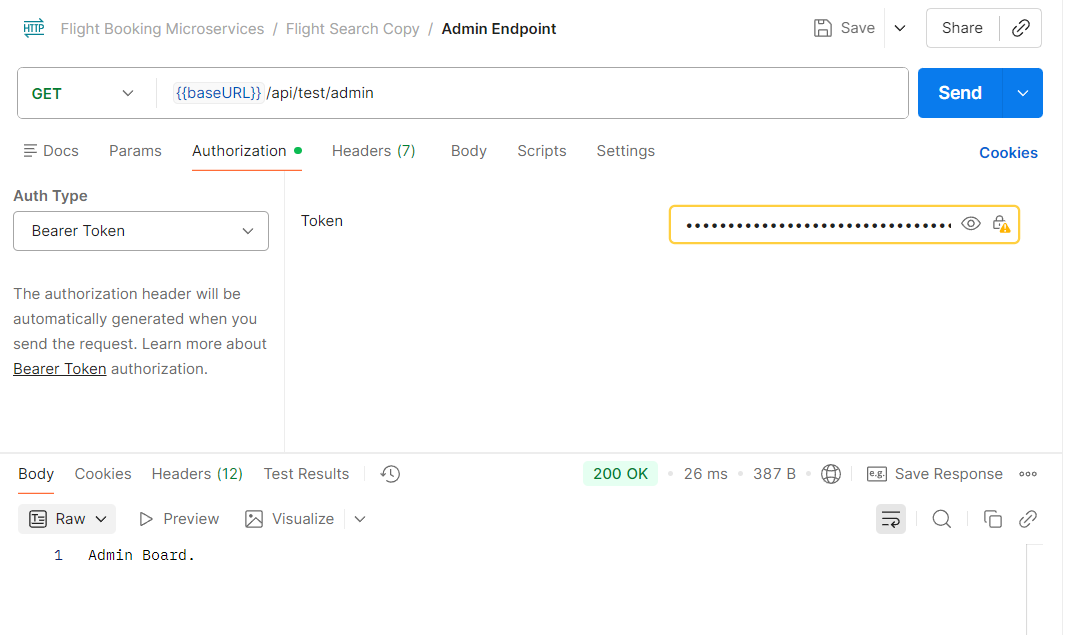
Test all :



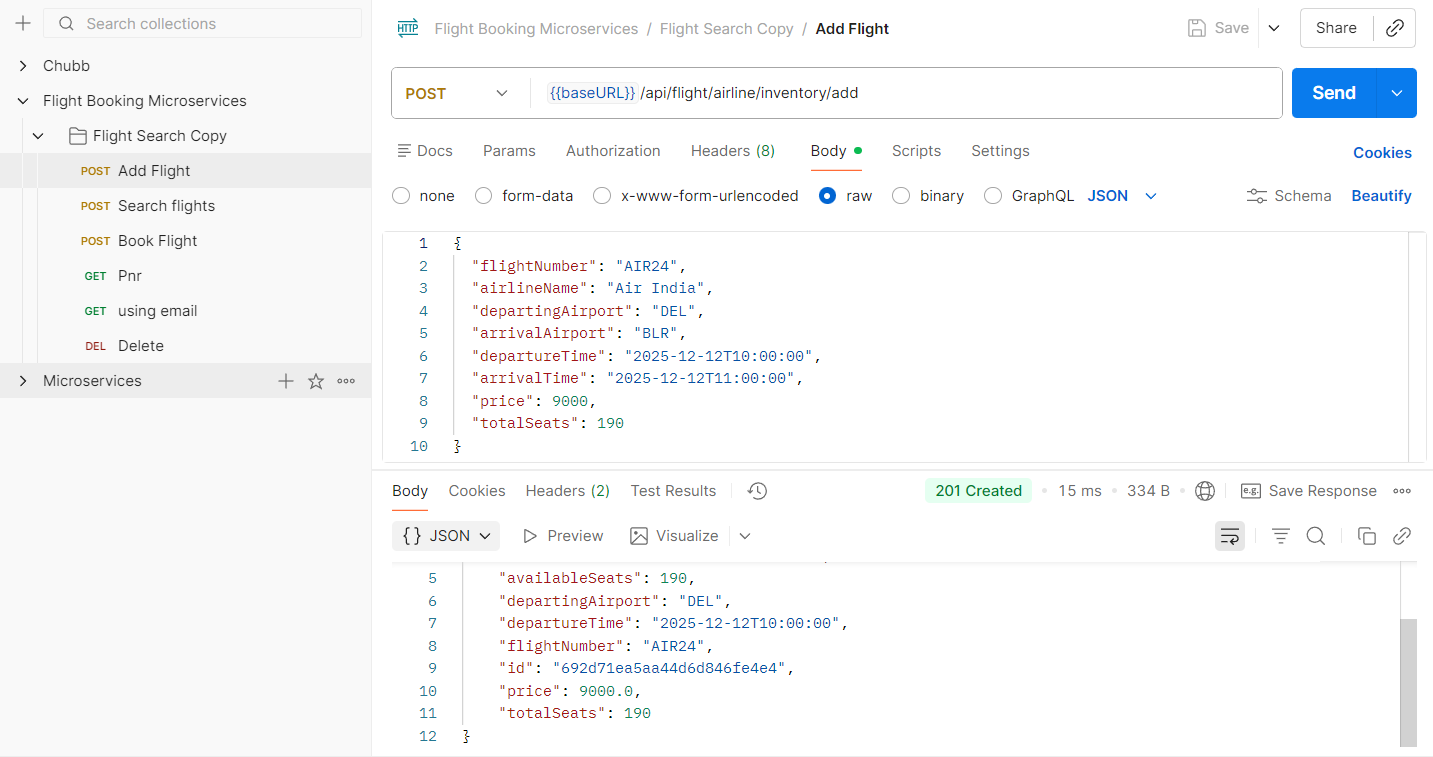
Test user with jwt token :



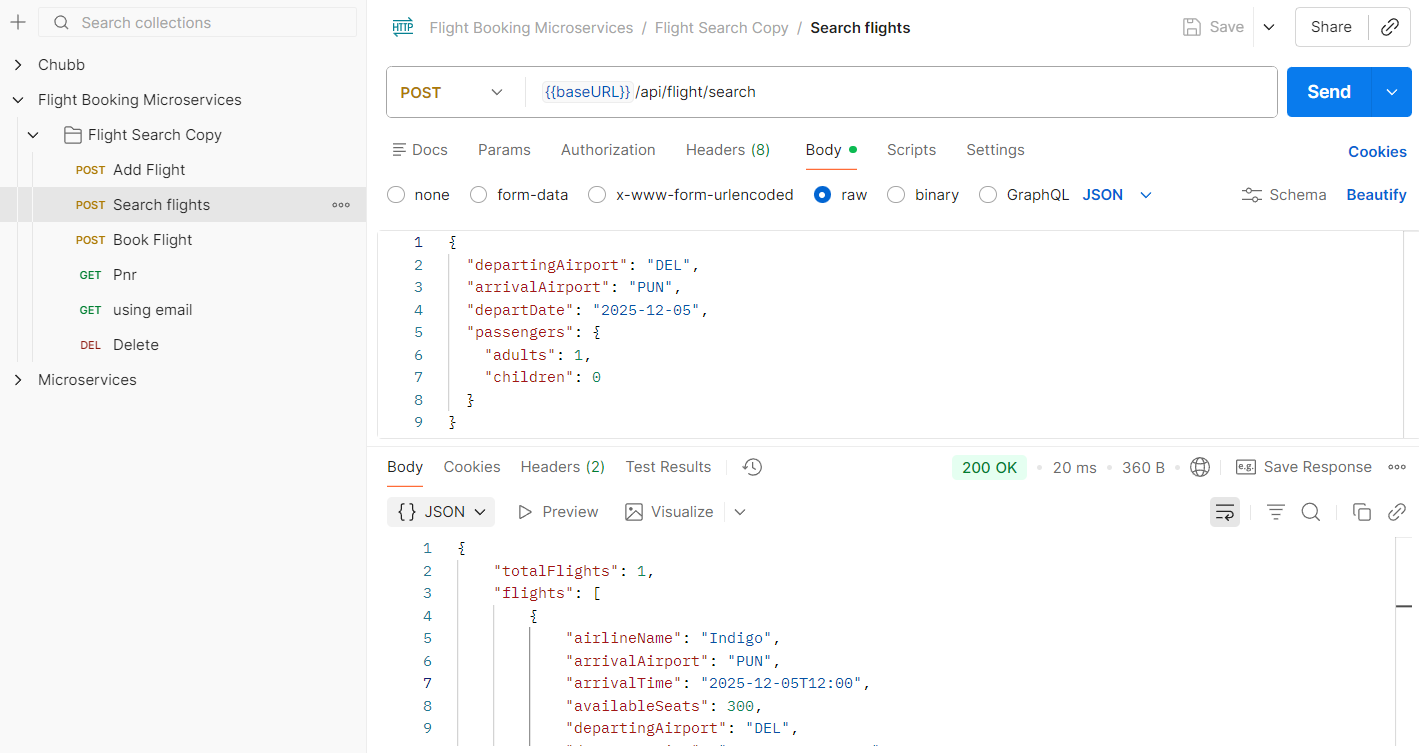
Test user with jwt token :



Add Flight (using baseURL) :



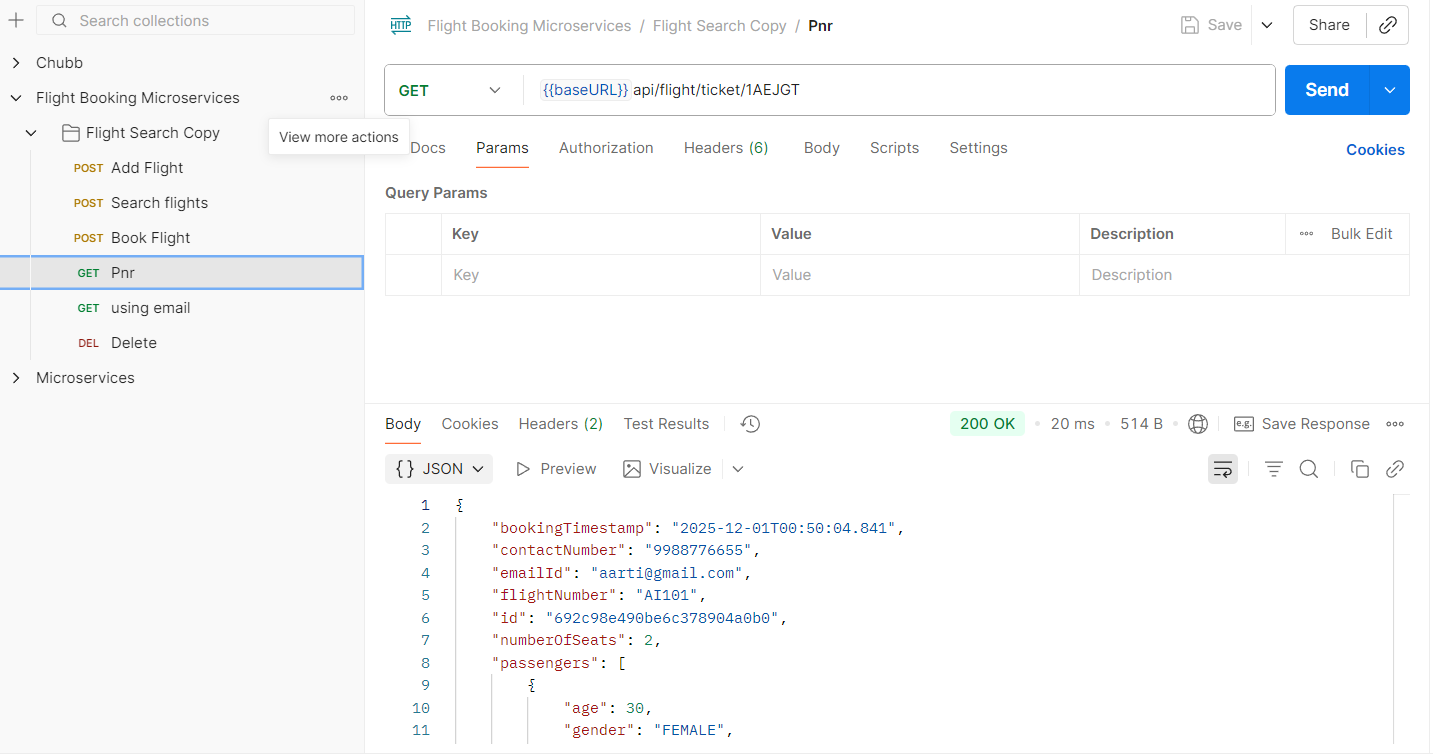
Search Flight



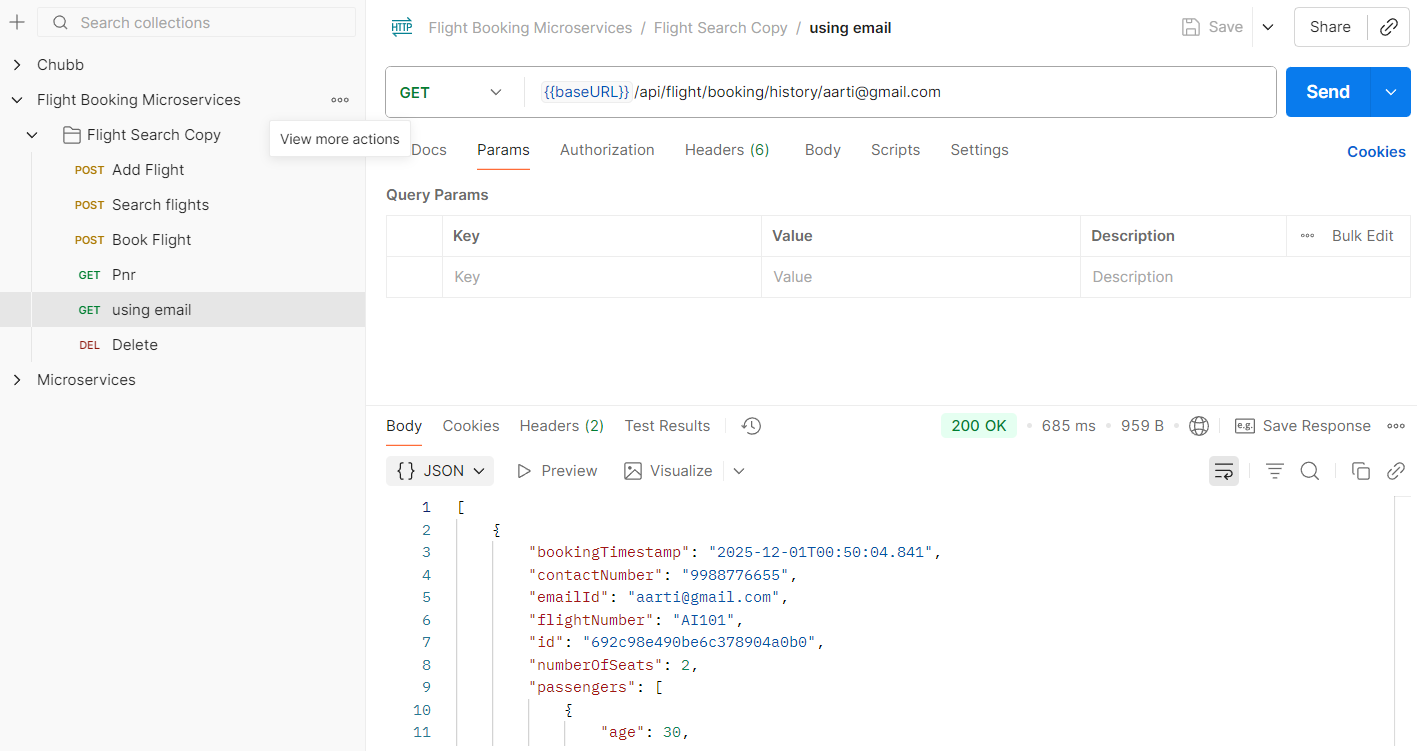
Book Flight



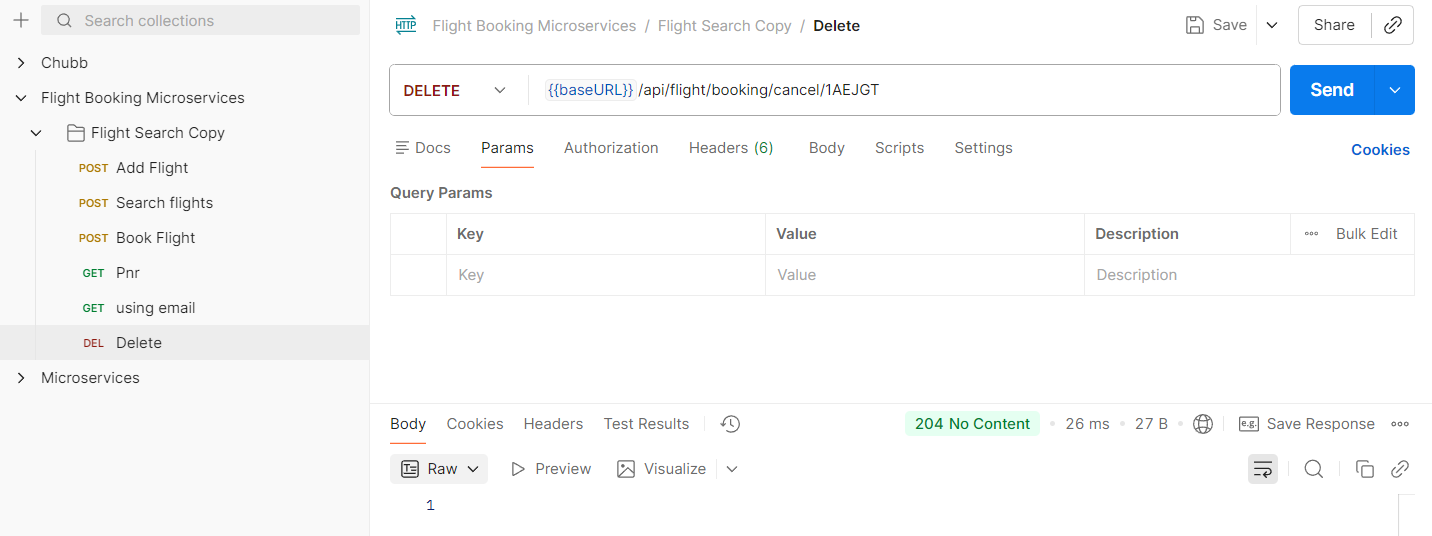
Get Flight Details by Pnr number



Get Flight Details by Email id

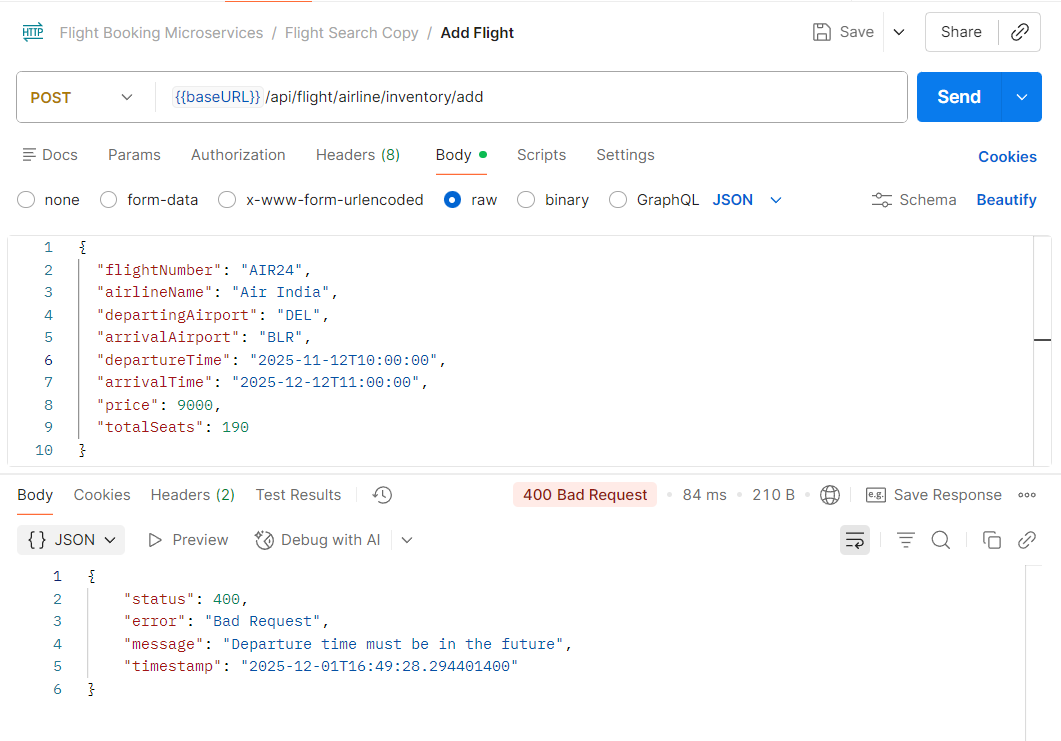


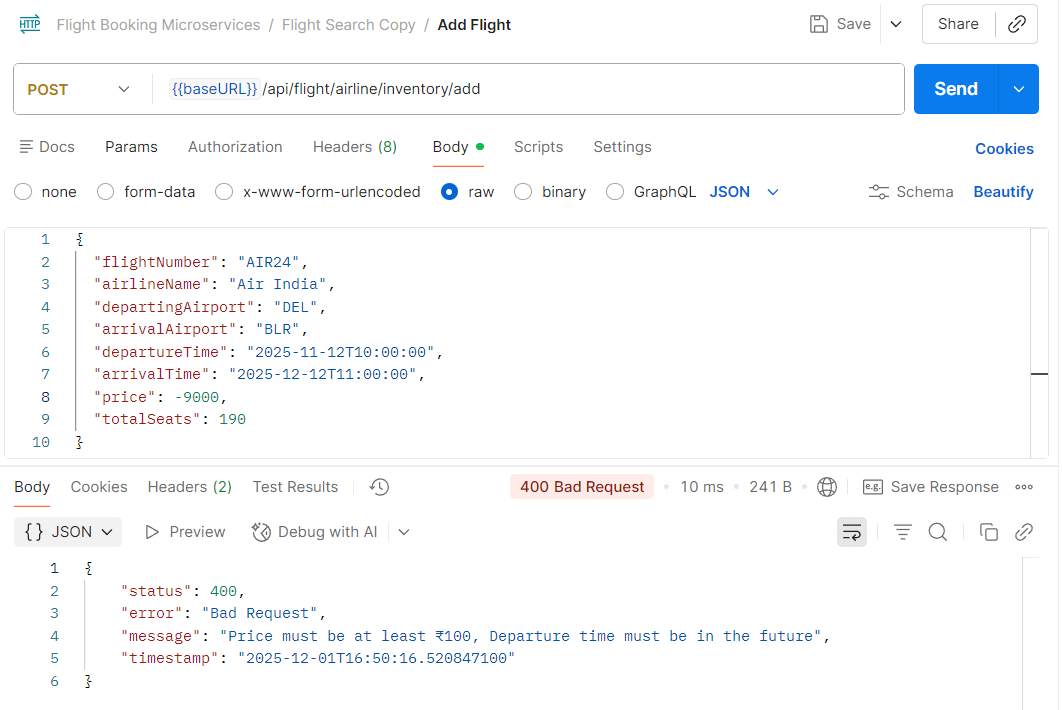
Cancel Booking



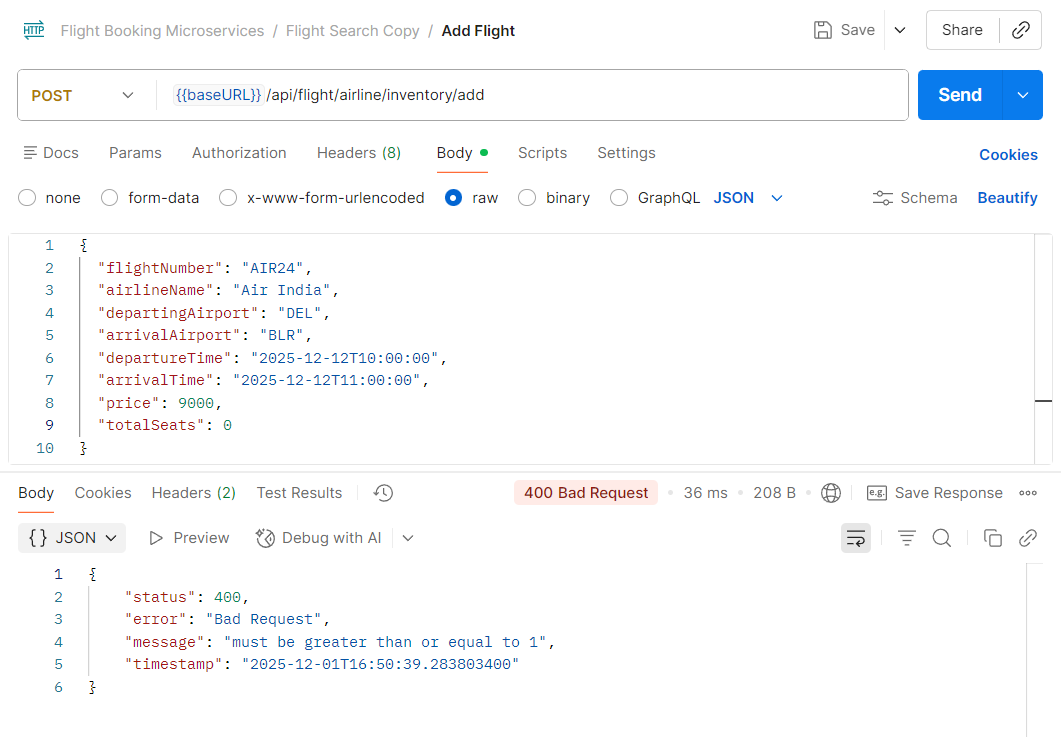
Negative testing with http status codes:

1. Wrong departure date



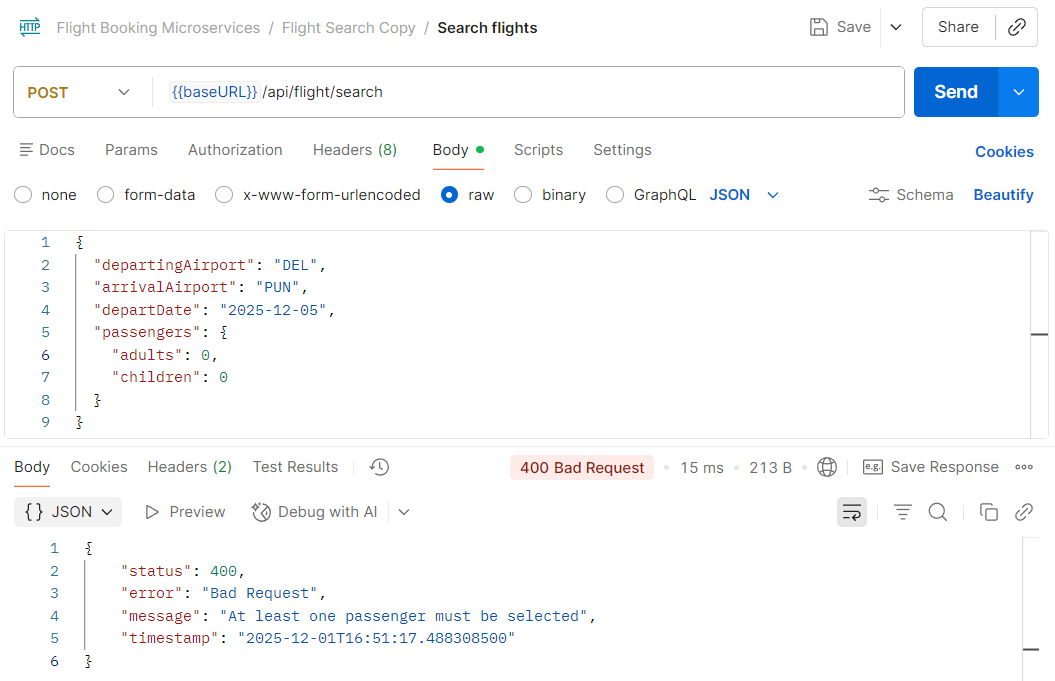
2. Negative price

3. 0 total seats

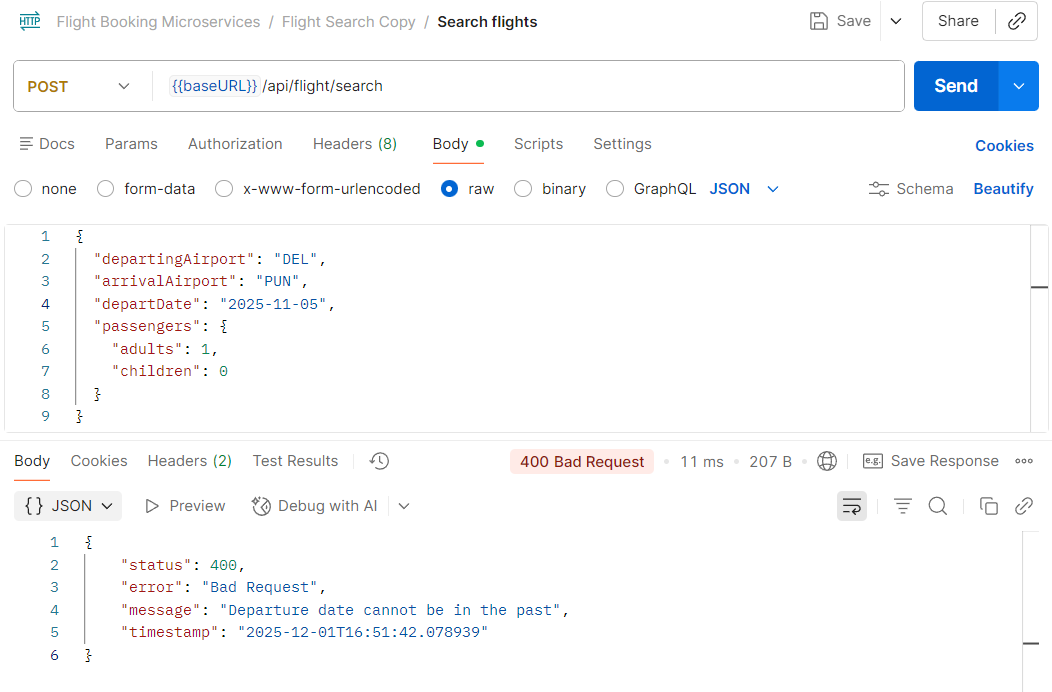


Negative testing:

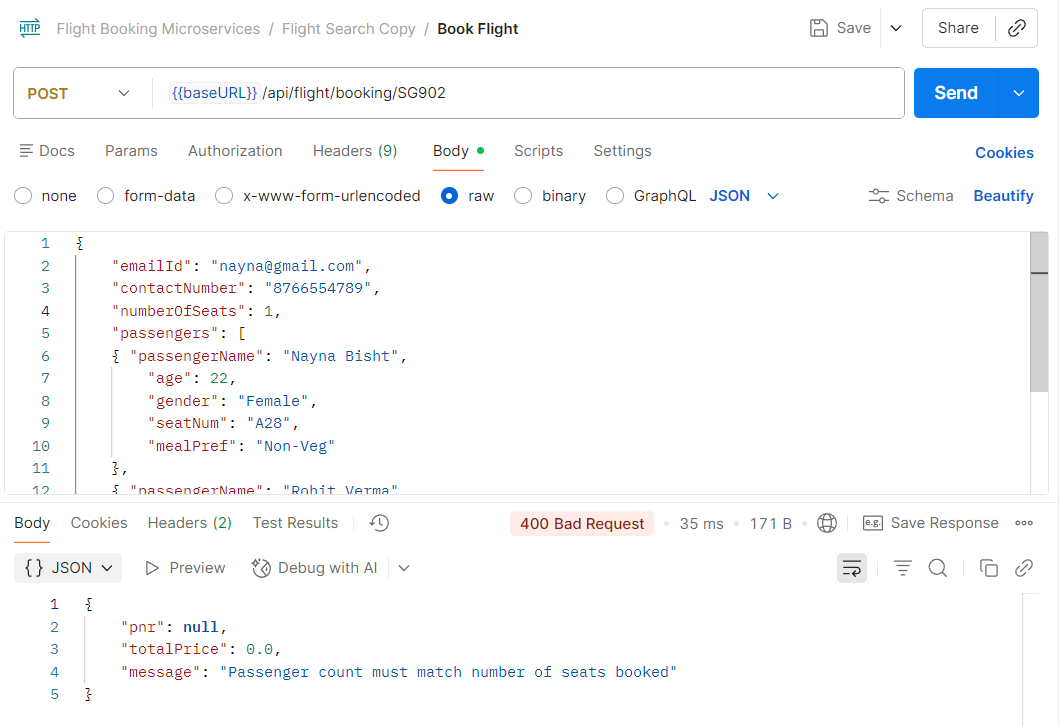
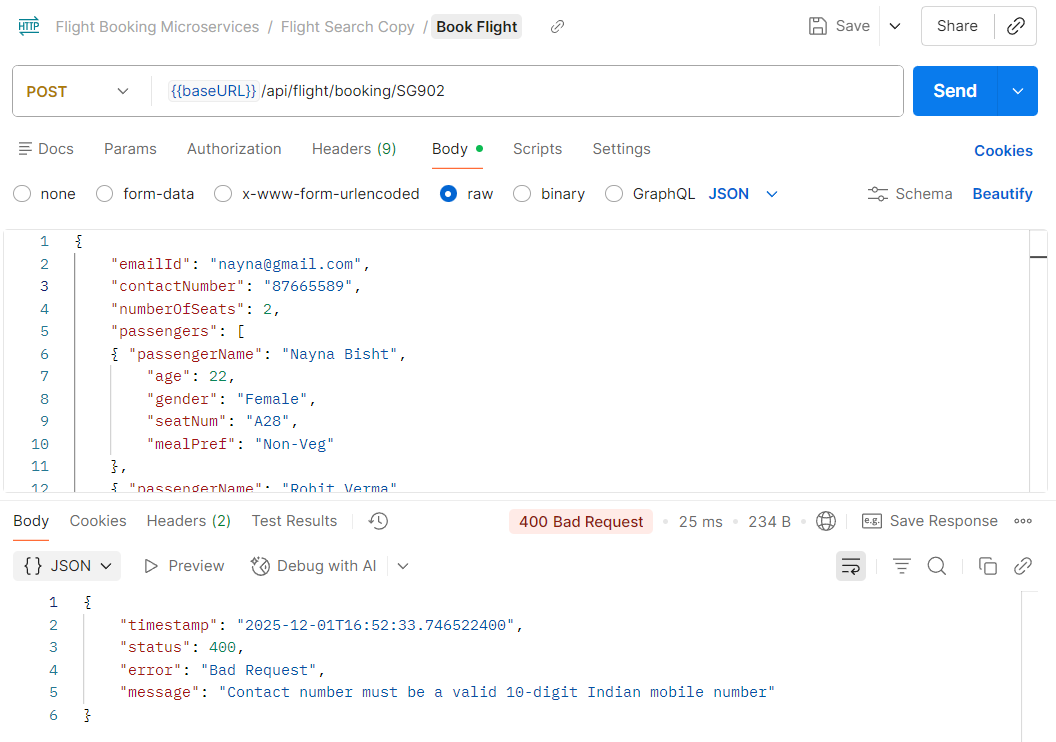
1. No passengers

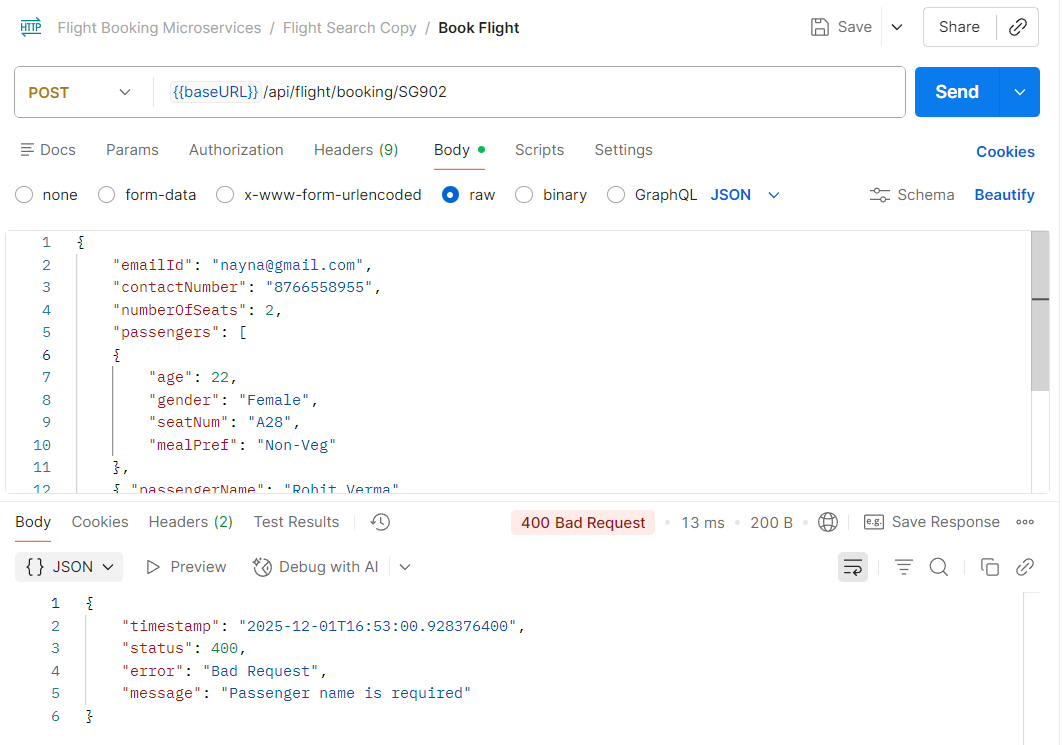


2. Wrong Departure date



Negative Testing :  
1.Passenger Count != no of passenger details filled

2. Wrong mobile number

3. Passenger Name missing

4. Email and contact number missing to book the flight