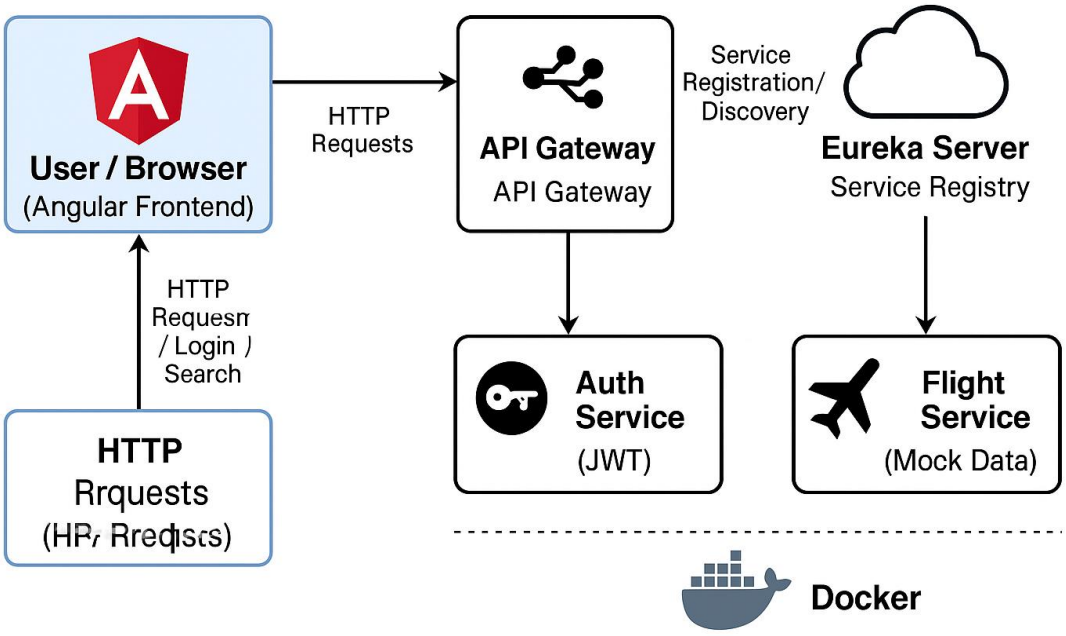
Important Screenshots

Flight Booking Application – Frontend using Angular:

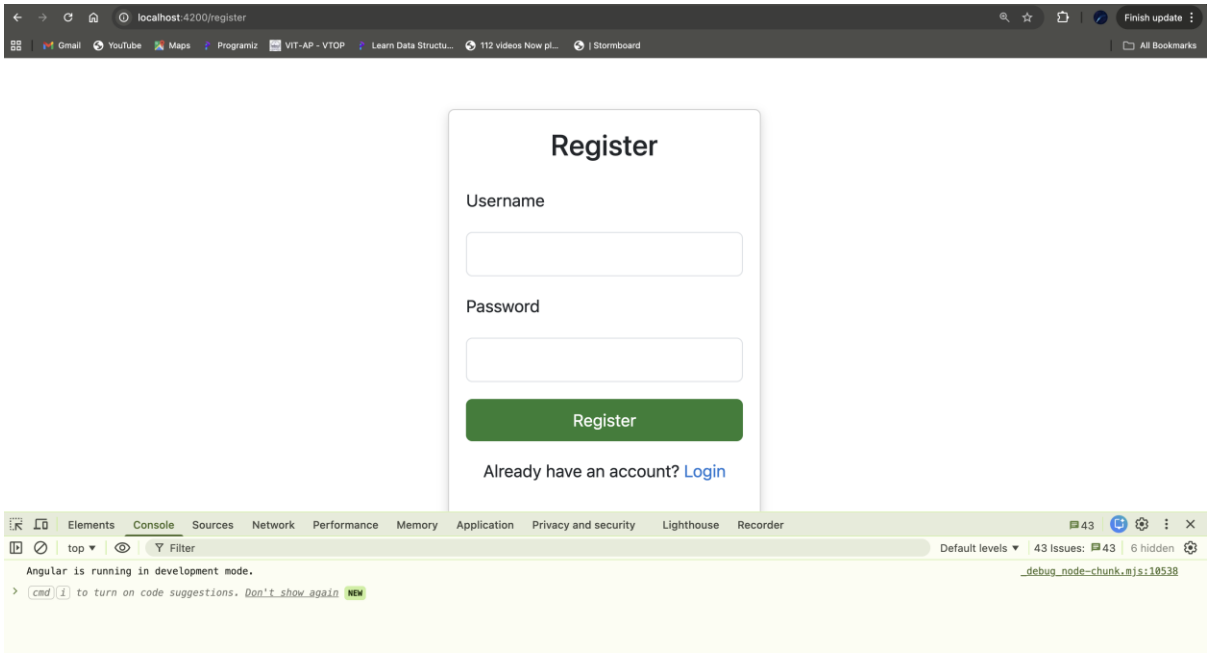
The application provides three main functionalities: User Registration, User Login, and Flight Search. Angular routing is used for navigation between components, and user-friendly forms with validations are implemented.

The application communicates with a backend API for authentication using **JWT (JSON Web Token)**.

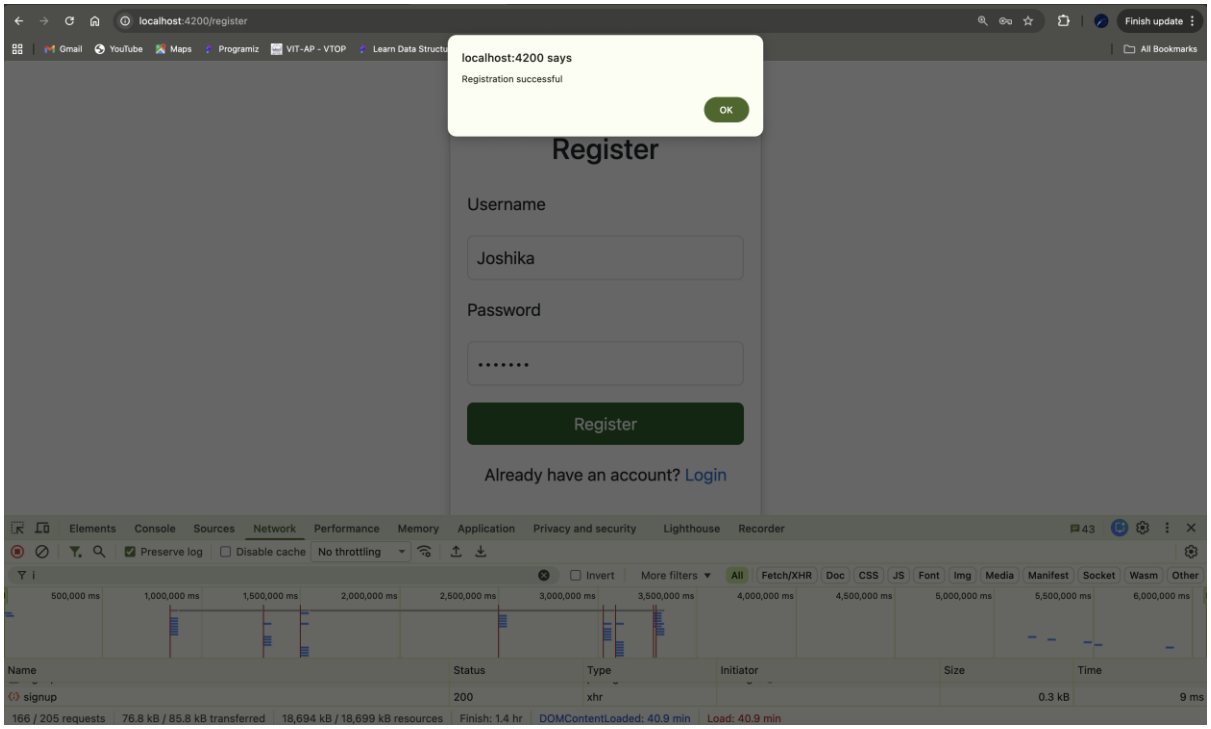
Flight Booking Application Architecture:



Register Page:



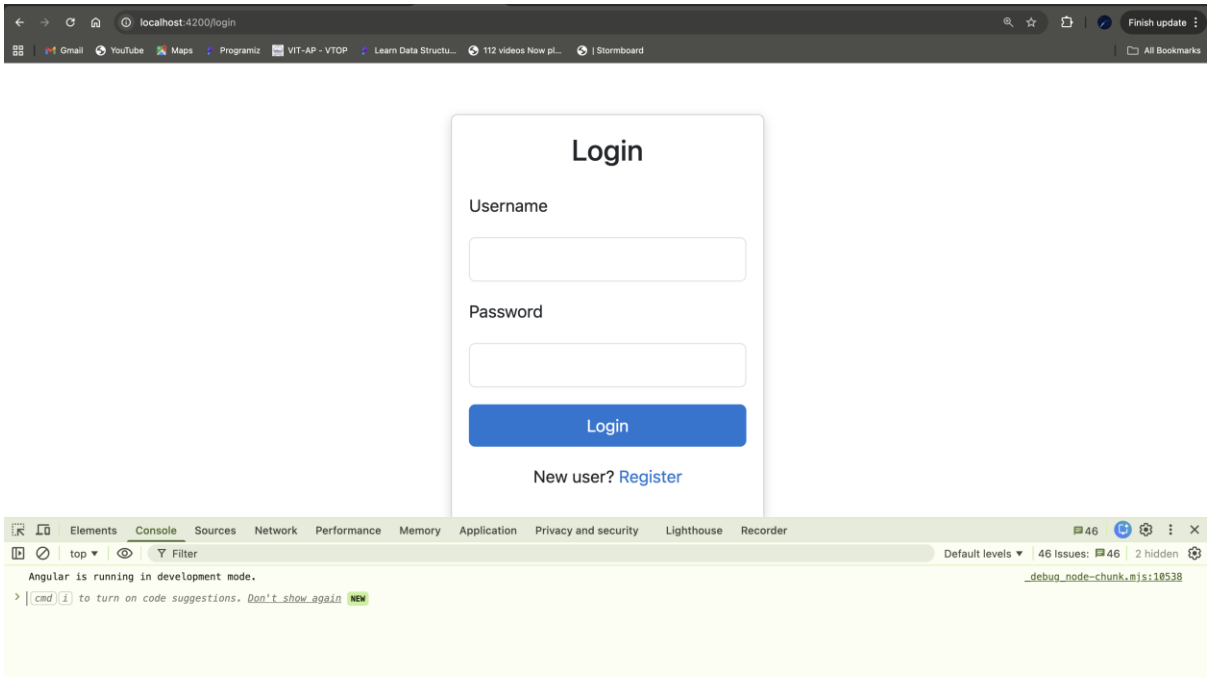
This is the register page. Now we'll see response after adding username and password.



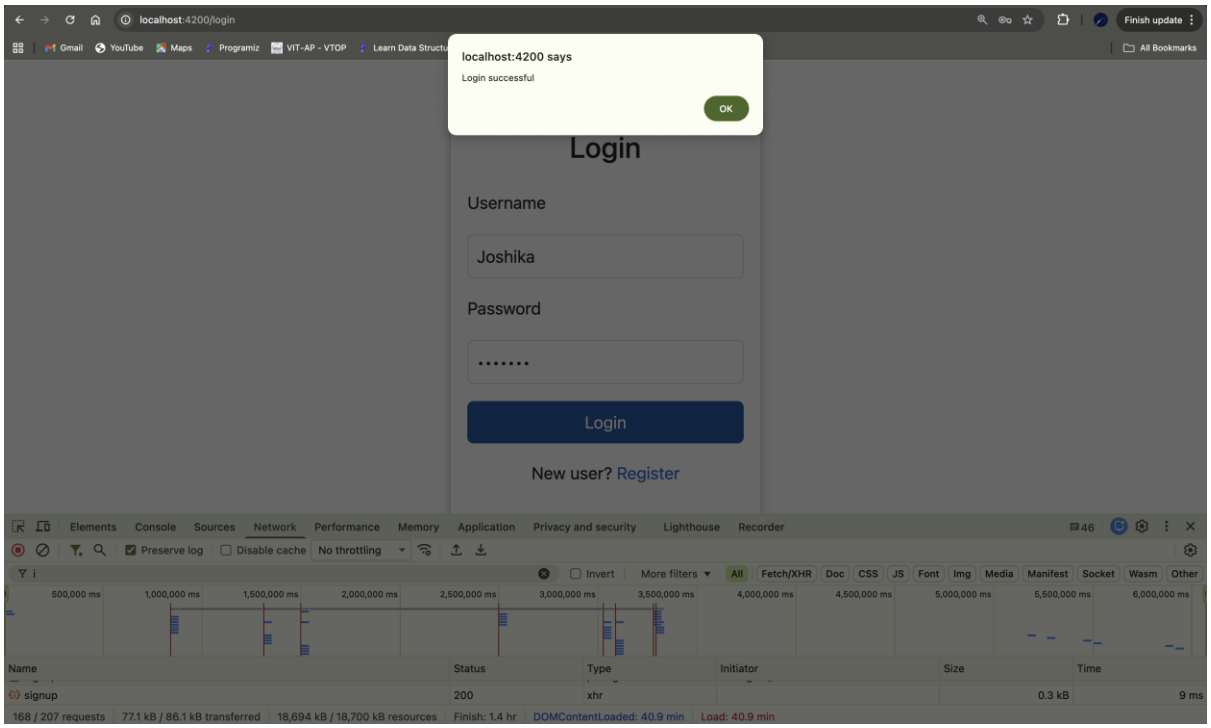
We can see the post status here. It shows that the backend is connected successfully.

The post request shows 200OK!

Login Page:



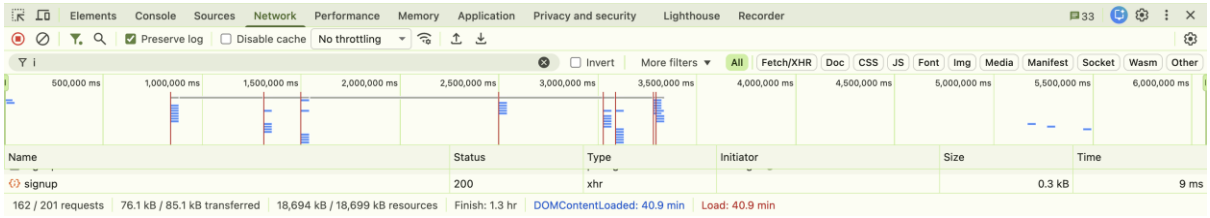
This is the Login page. Now we'll see response after adding username and password.

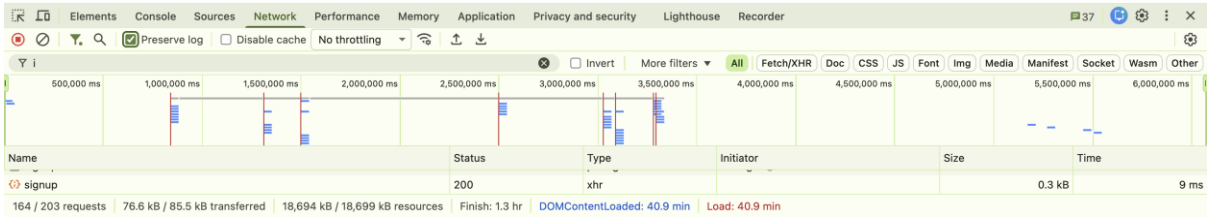


We can see the post status here. It shows the backend is connected successfully.

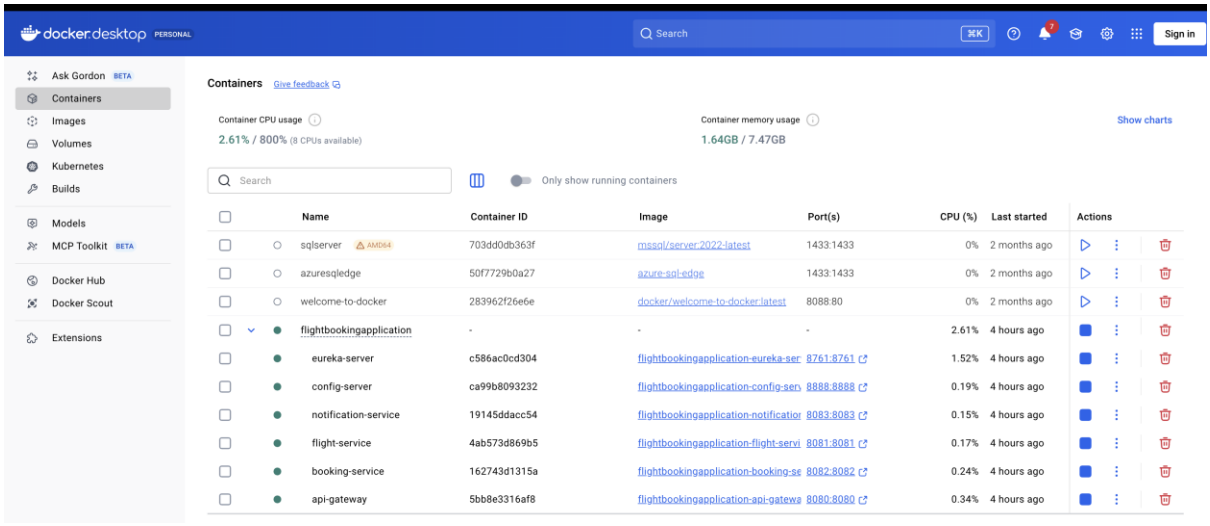
The post request shows 200OK!

Backend Connection Proof:

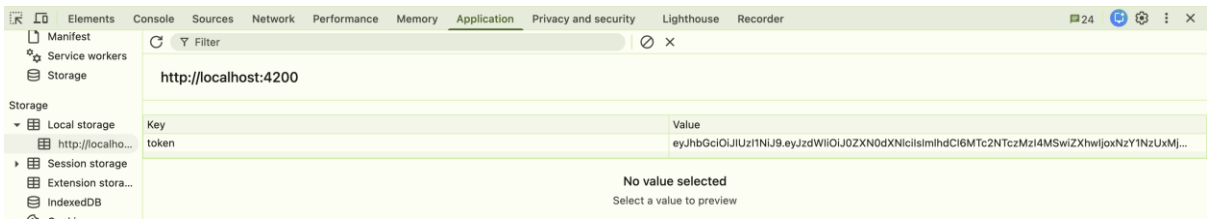




The requests used in backend.

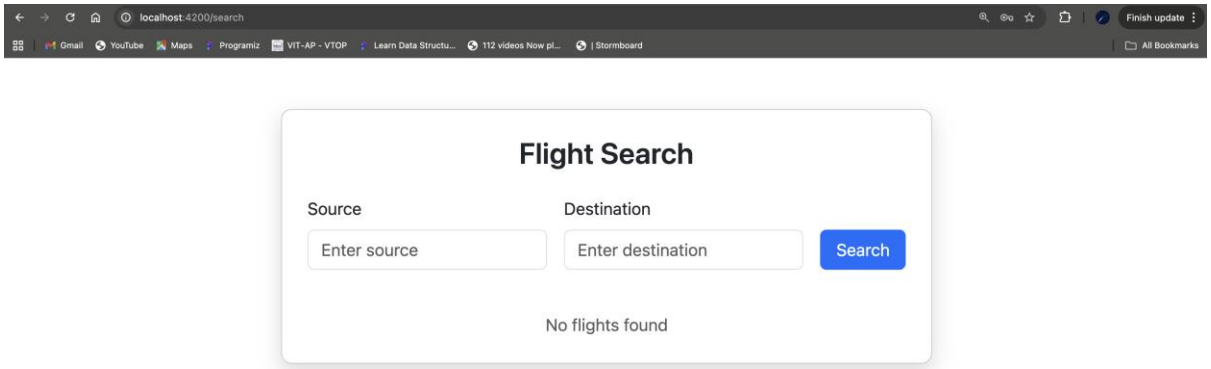


Docker running all the services

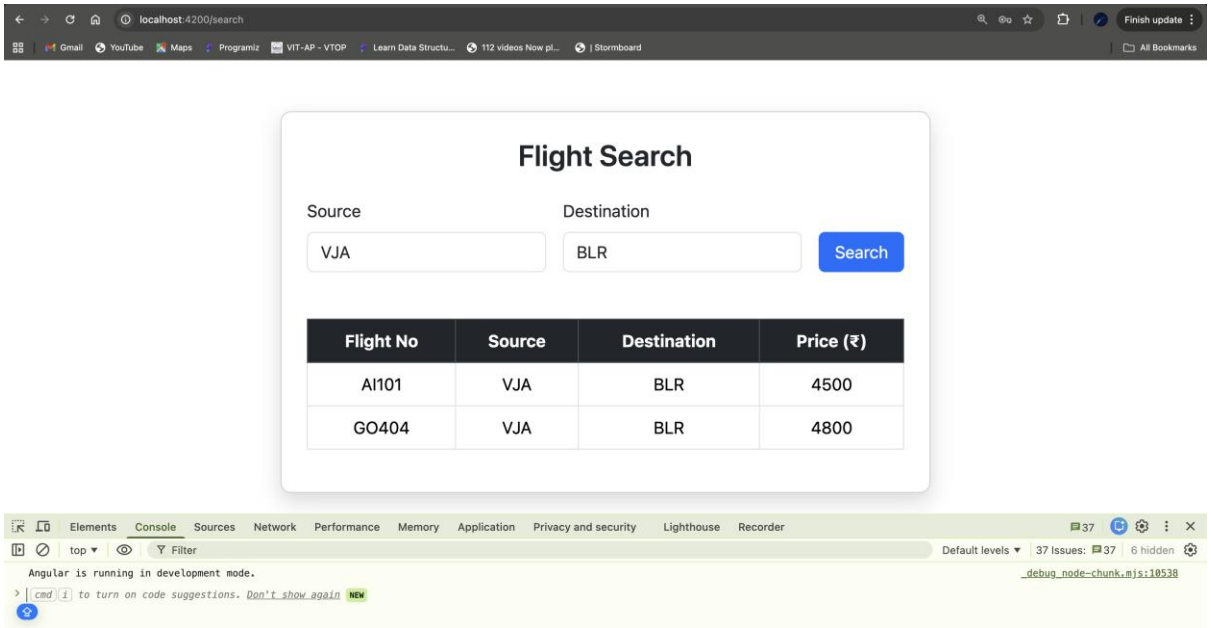


Token generated after Login.

Flight Search Page:



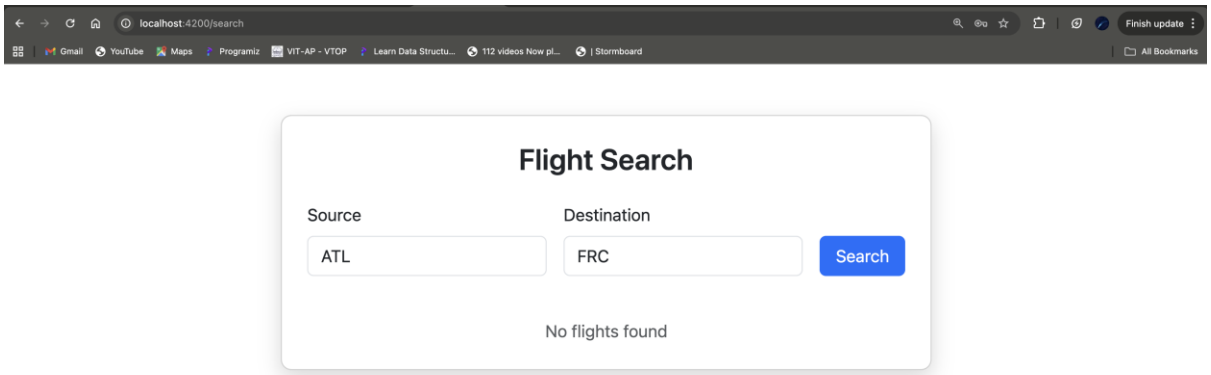
This is the Flight search page. Now we'll see response after adding src and dest.

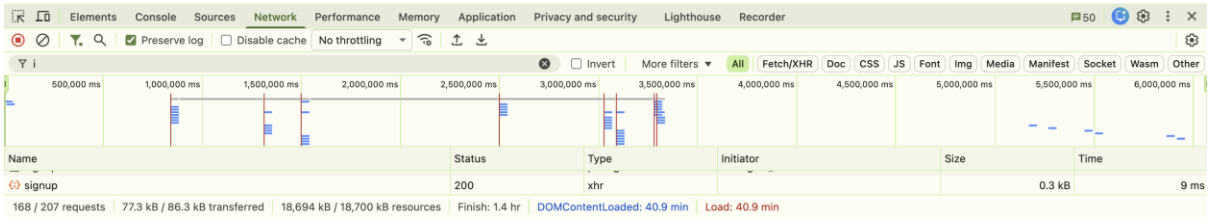


It shows available flights from source to destination.

EDGE CASES:

If flight not found.



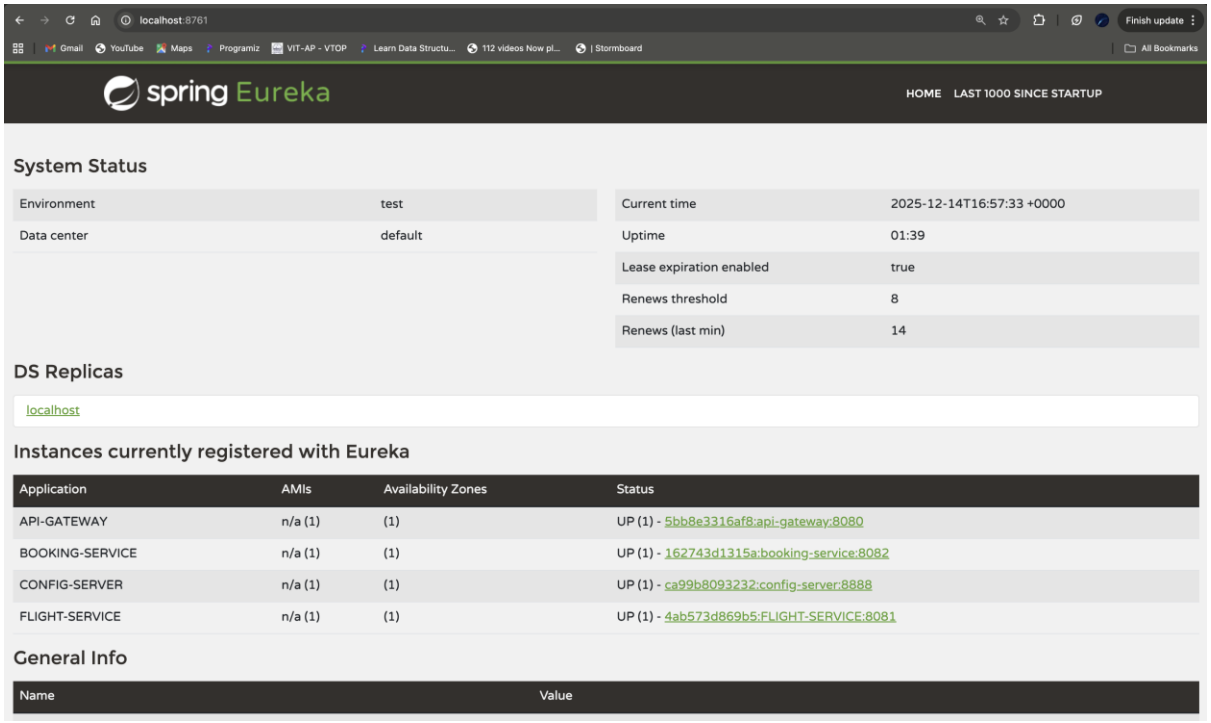


CODE ADDED IN BACKEND FOR INTEGRATION:

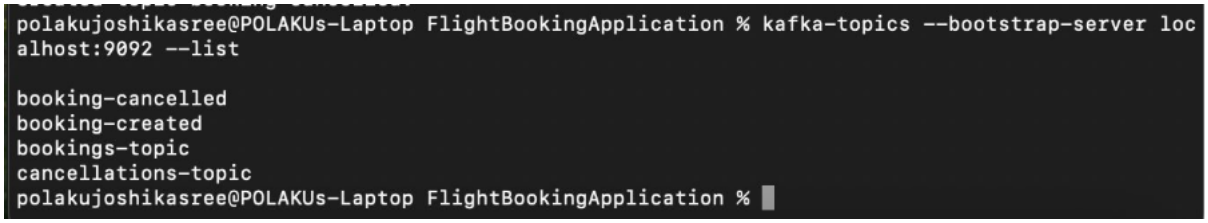
```
package com.apigateway.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.reactive.CorsWebFilter;
import org.springframework.web.cors.reactive.UrlBasedCorsConfigurationSource;

@Configuration
public class CorsConfig {
    @Bean
    public CorsWebFilter corsWebFilter() {
        CorsConfiguration config = new CorsConfiguration();
        config.addAllowedOrigin("http://localhost:4200");
        config.addAllowedMethod("*");
        config.addAllowedHeader("*");
        config.setAllowCredentials(true);
        UrlBasedCorsConfigurationSource source =
                new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", config);
        return new CorsWebFilter(source);
    }
}
```

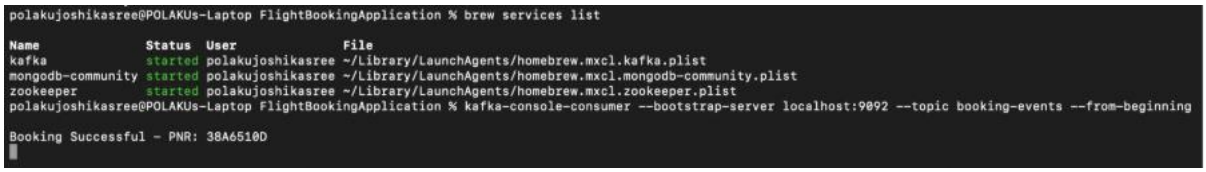
Working Eureka:



KAFKA:

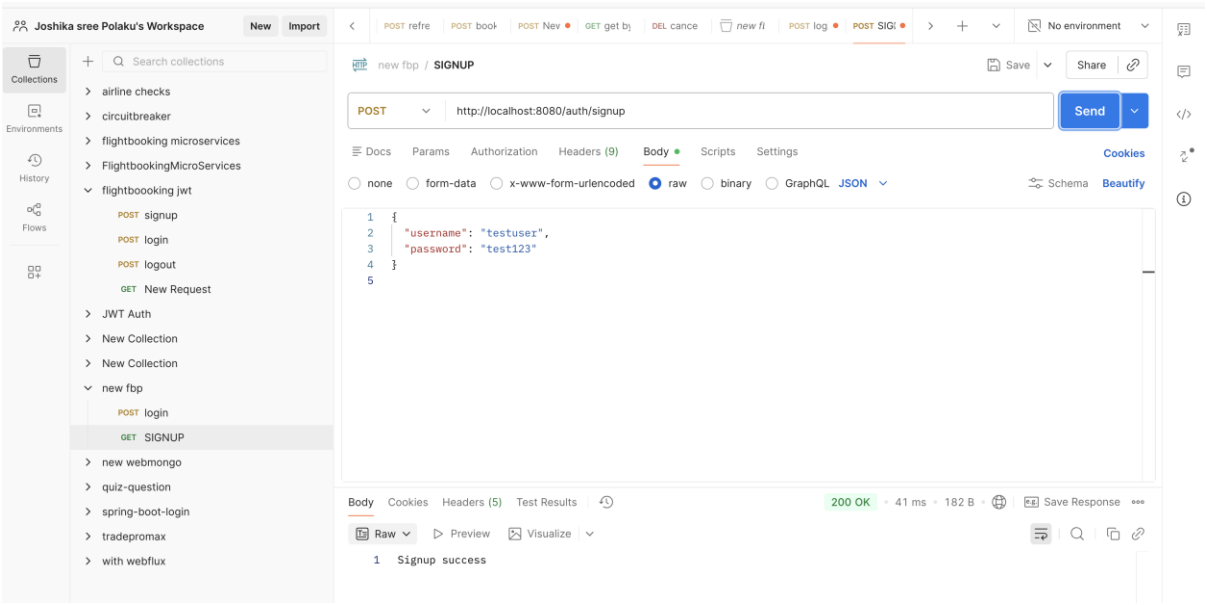


Notification:

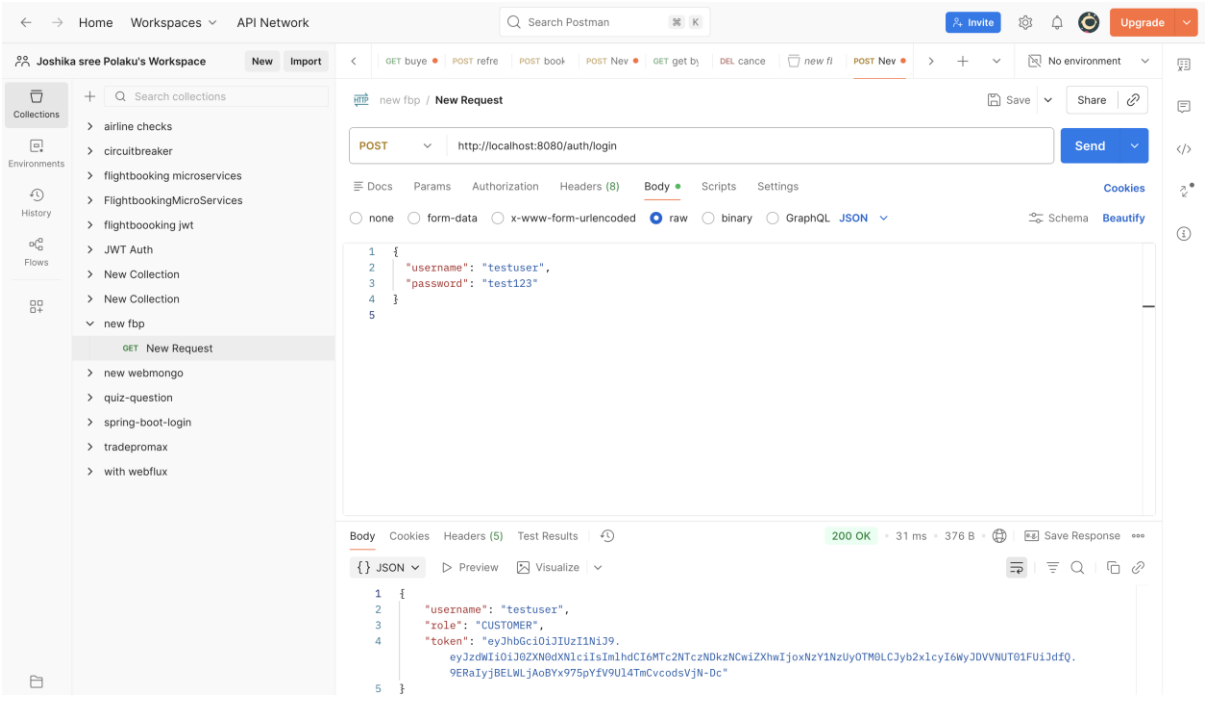


POSTMAN:

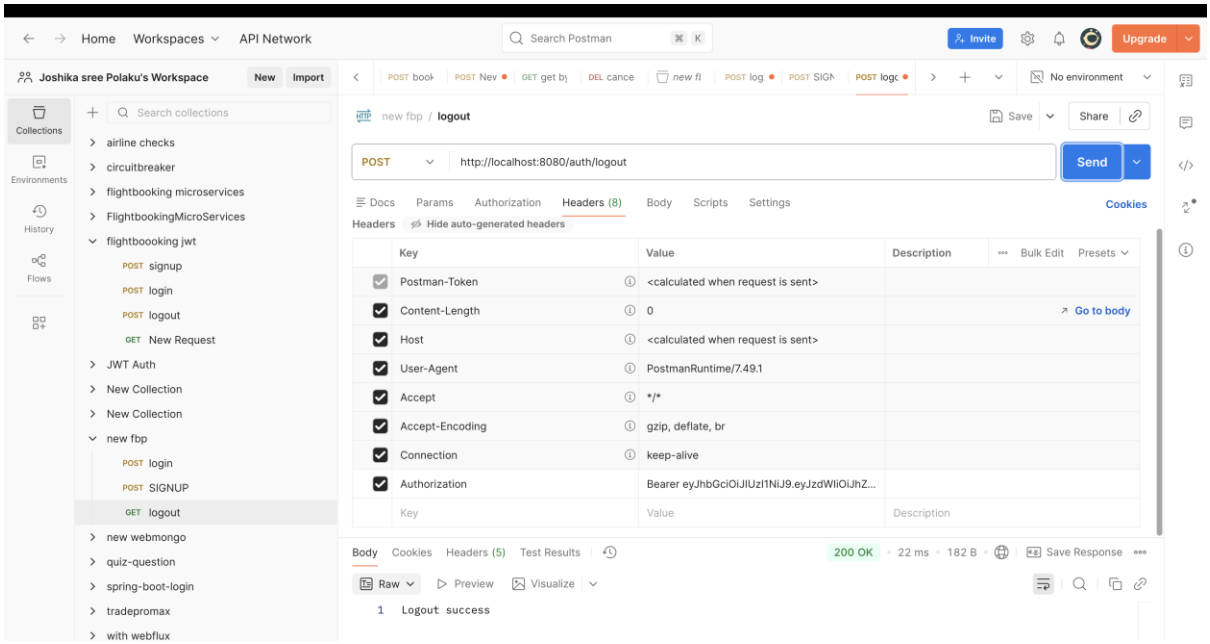
SignUp:



Login:



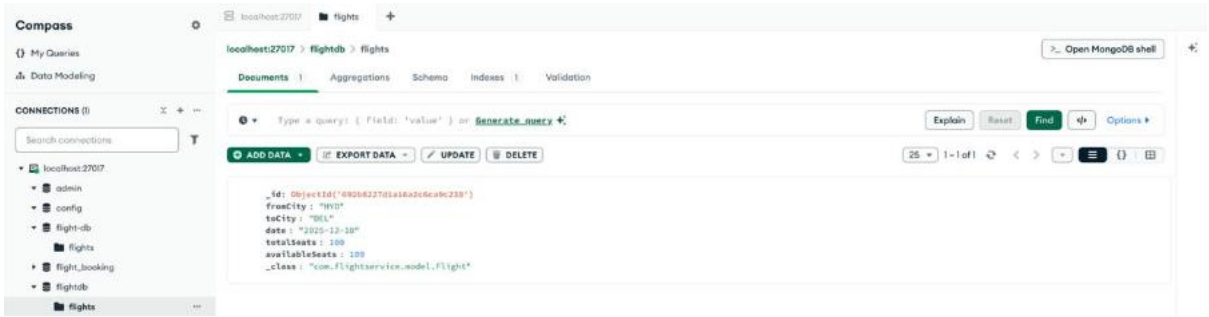
Logout:



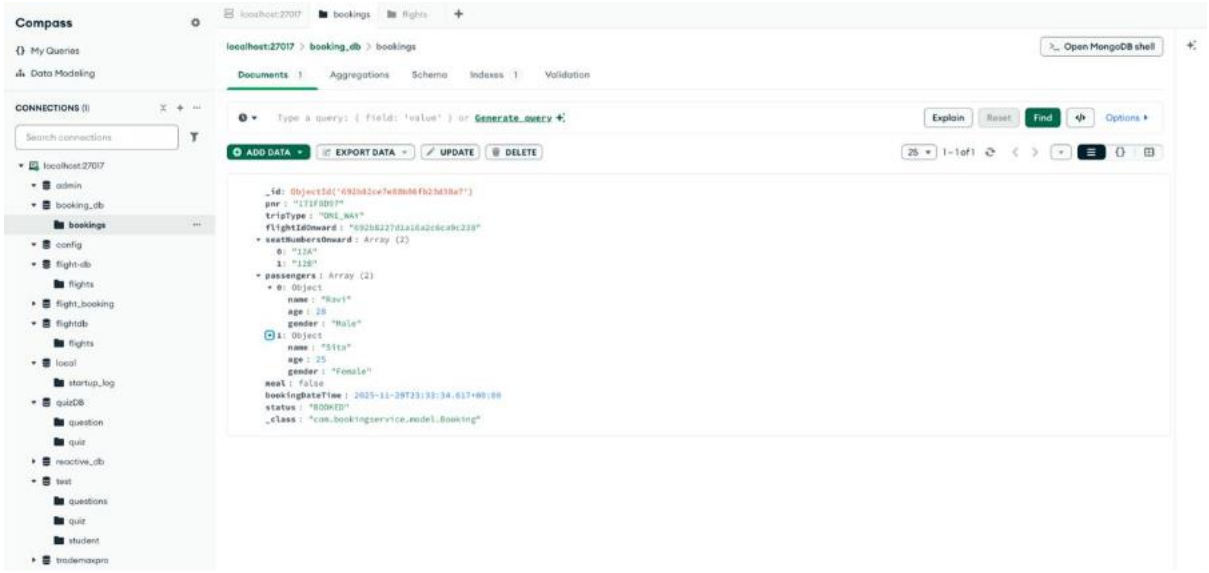
MongoDB:(2 databases)

booking_db	20.48 kB	1	1
config	12.29 kB	0	2
flightdb	20.48 kB	1	1

FlightDB:

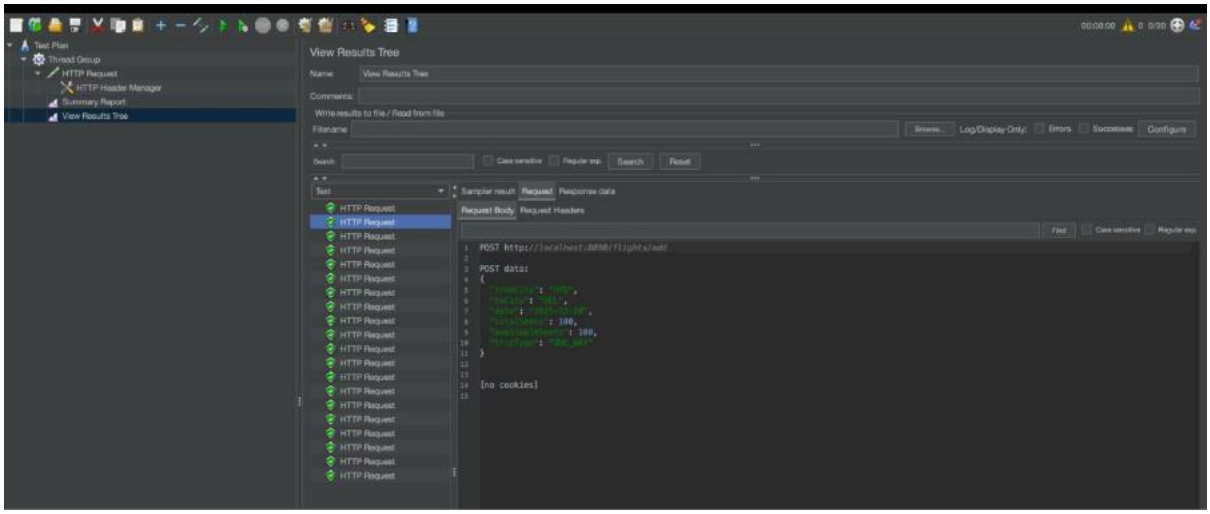


BookingDB:

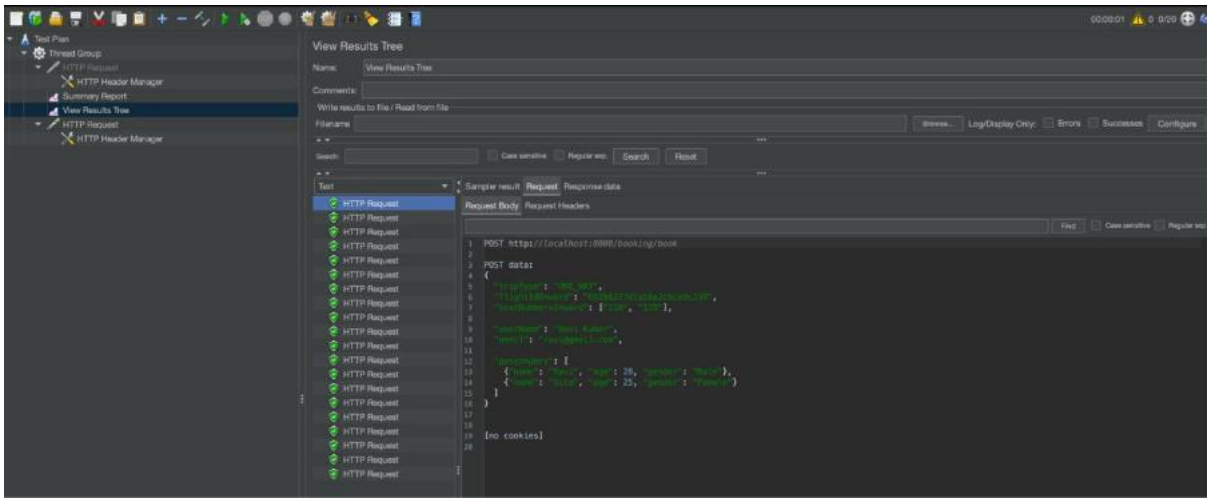


JMeter:

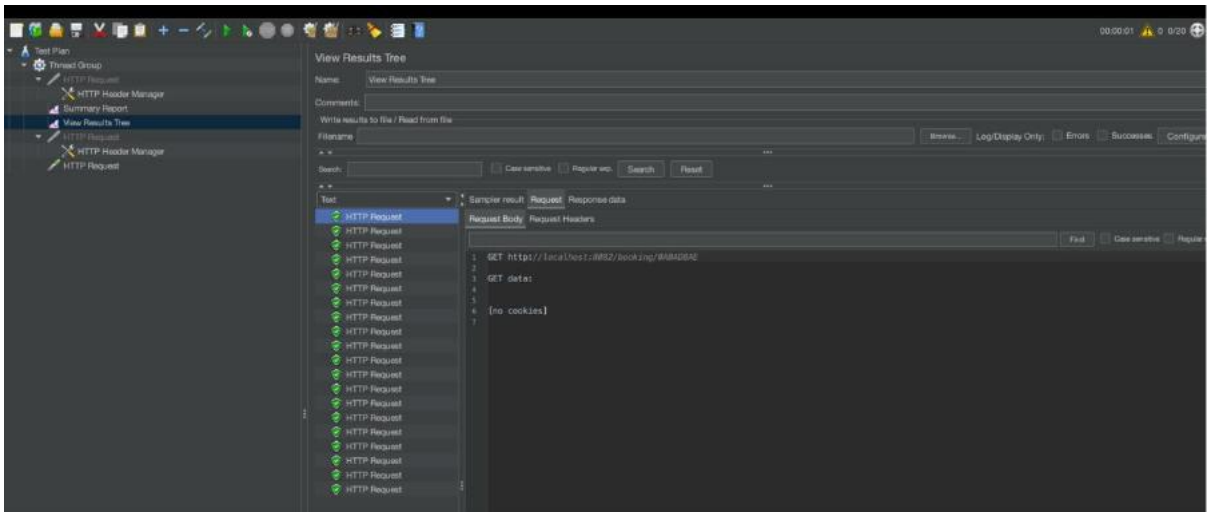
Post request (add inventory):

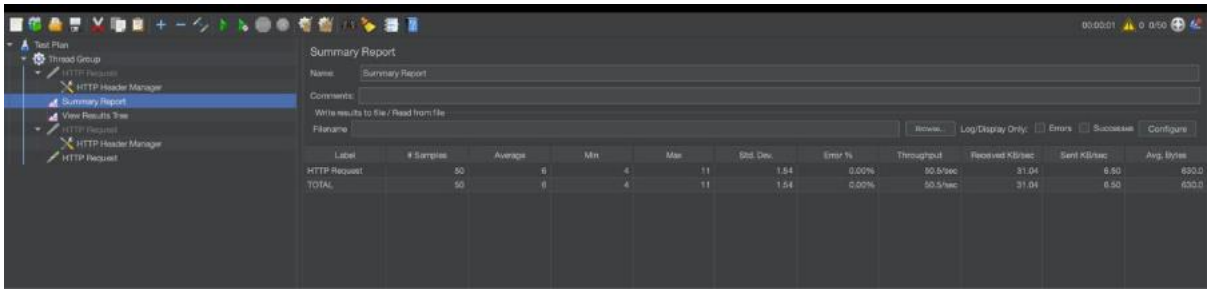


Add Booking :

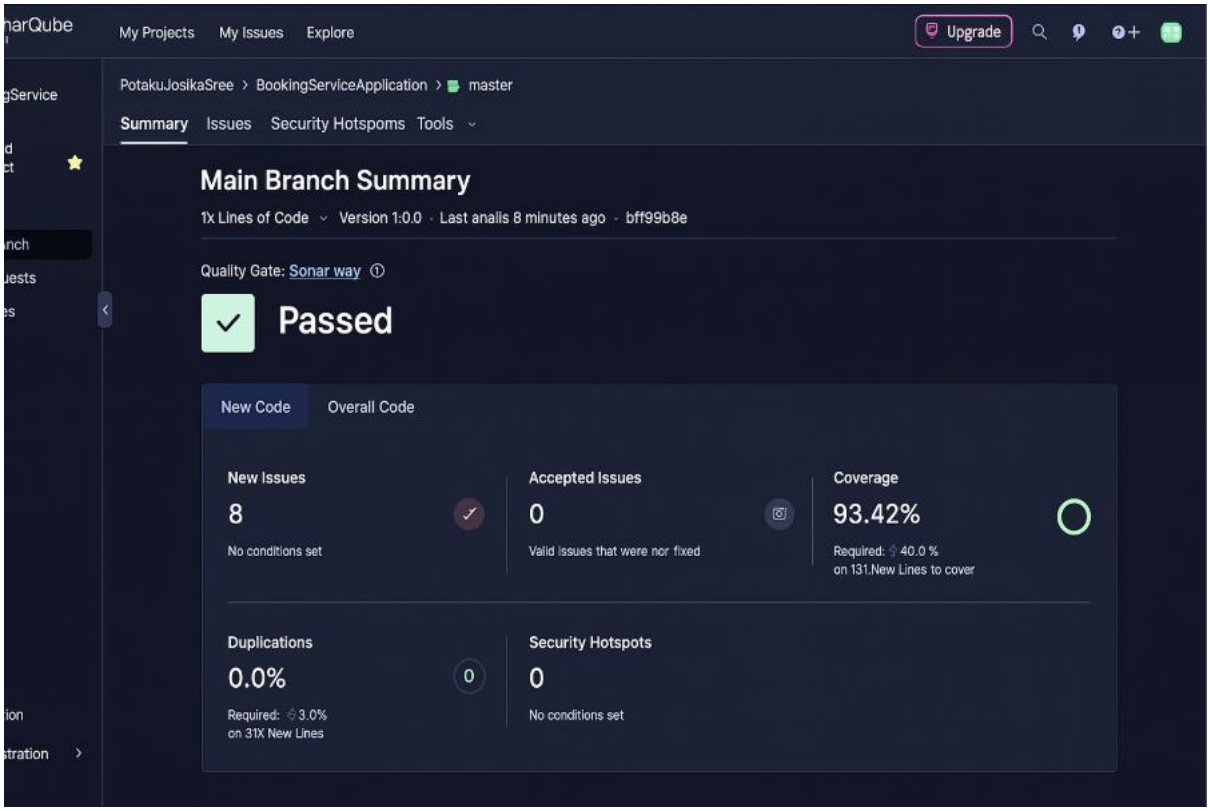


Get PNR:





Sonar Analysis:



CONCLUSION:

- Successfully implemented Angular frontend
- Authentication integrated with backend
- Clean UI and component-based design