

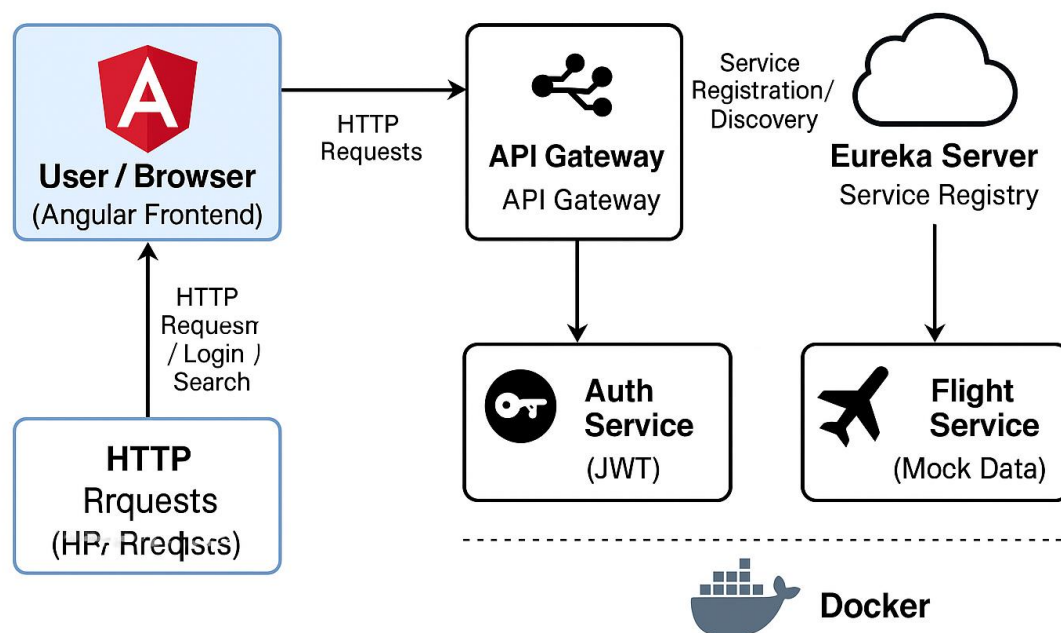
Important Screenshots

Flight Booking Application – Frontend using Angular:

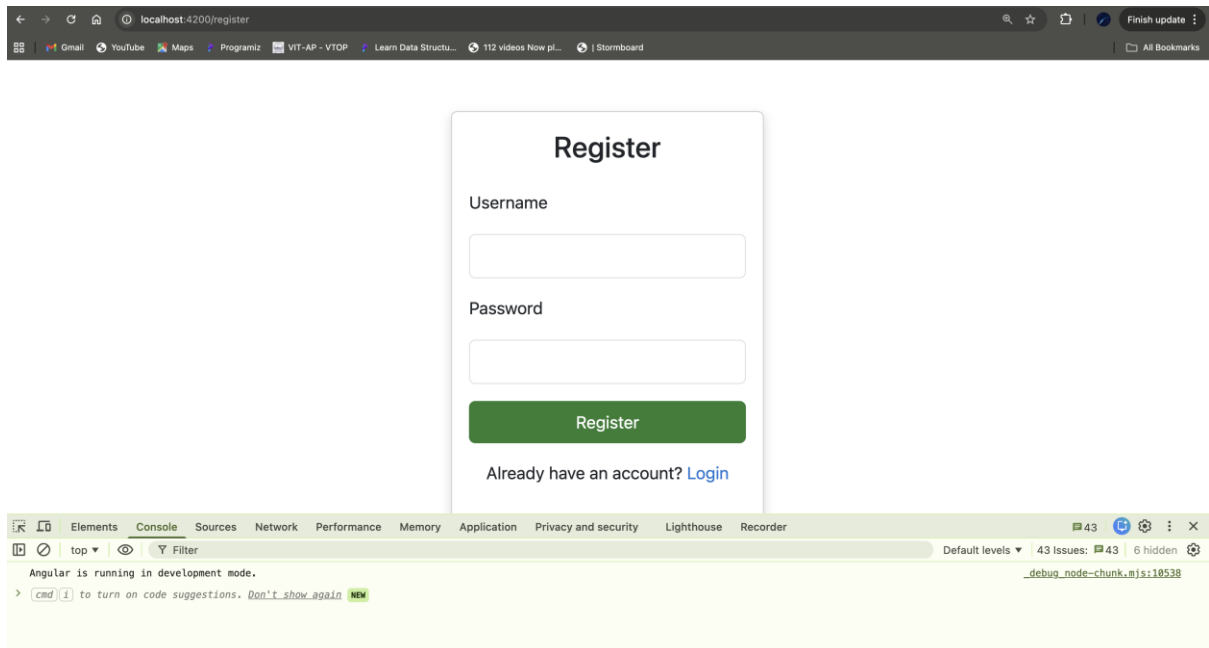
The application provides three main functionalities: User Registration, User Login, and Flight Search. Angular routing is used for navigation between components, and user-friendly forms with validations are implemented.

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

Flight Booking Application Architecture:

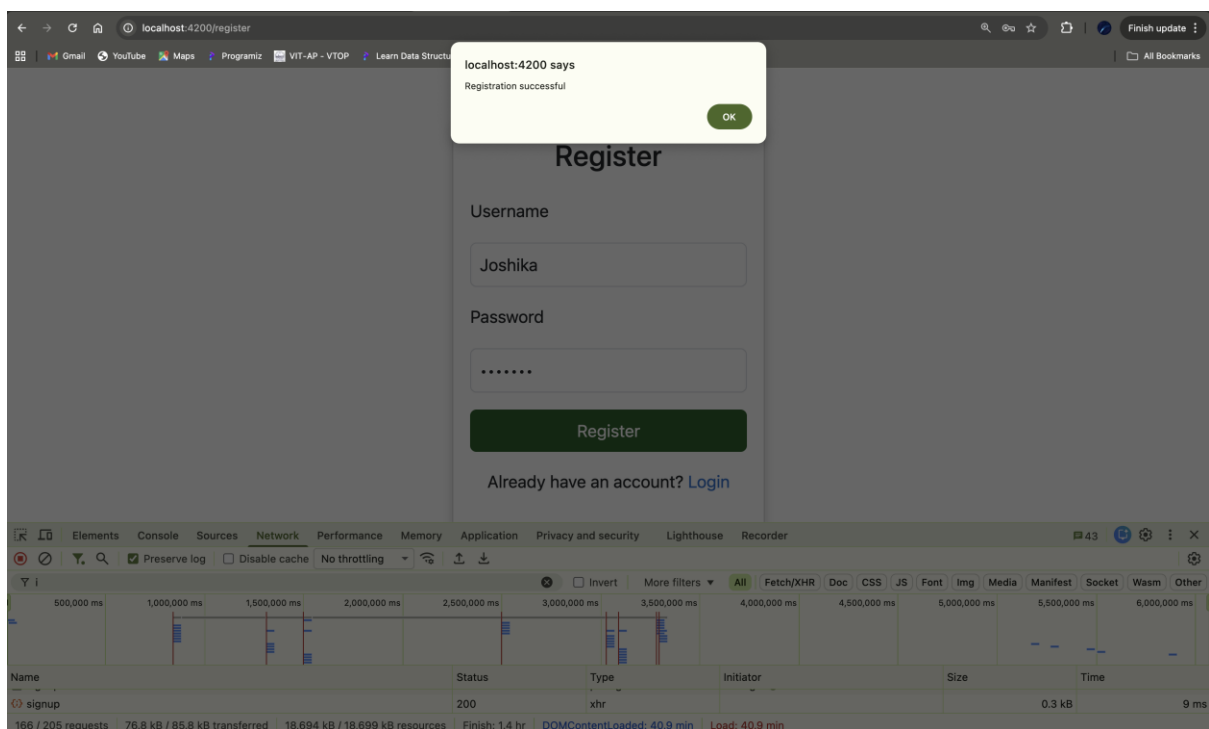


Register Page:



The screenshot shows a web browser at localhost:4200/register. The page has a white background with a light gray border. It features a title "Register" in bold black text. Below the title are two input fields: "Username" and "Password". The "Username" field contains the text "Joshika". Below the "Password" field is a green button labeled "Register". At the bottom of the form, there is a link that says "Already have an account? [Login](#)". The browser's developer tools are open at the bottom, showing the Console tab with a message: "Angular is running in development mode. > [cmd] to turn on code suggestions. Don't show again NEW".

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

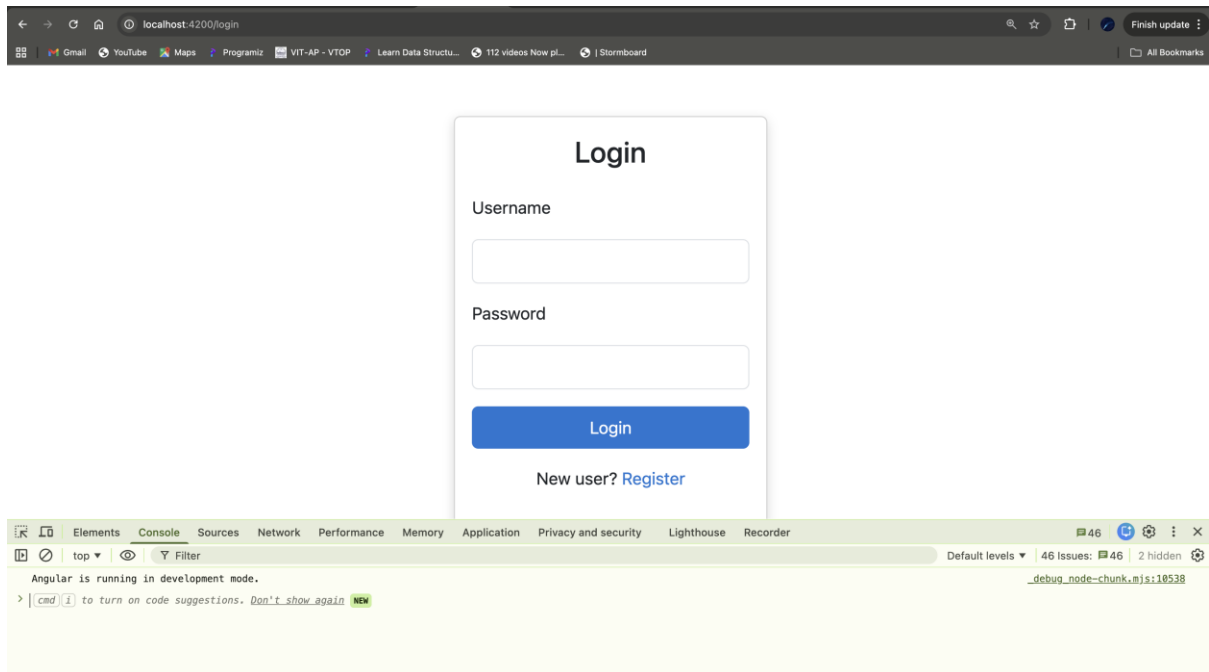


The screenshot shows the same Register page as before, but now with a yellow toast notification that says "localhost:4200 says Registration successful" with an "OK" button. The browser's developer tools are open, and the Network tab is selected. It shows a list of requests, with the first one being a "signup" request. The status is "200", the type is "xhr", and the size is "0.3 kB". The time taken for the request is "9 ms". The console also shows a message: "166 / 205 requests 76.8 kB / 85.8 kB transferred 16,694 kB / 16,699 kB resources Finish: 1.4 hr DOMContentLoaded: 40.9 min Load: 40.9 min".

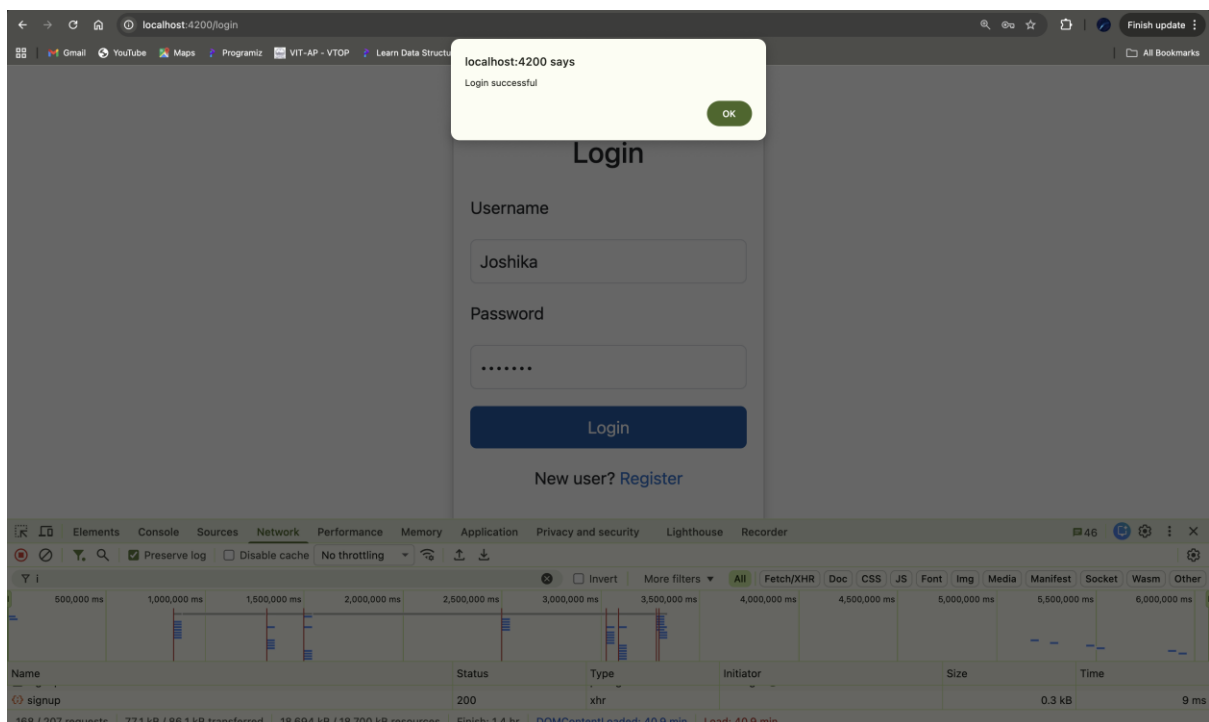
We can see the post status here. It shows that the backend is connected successfully.

The post request shows 200OK!

Login Page:



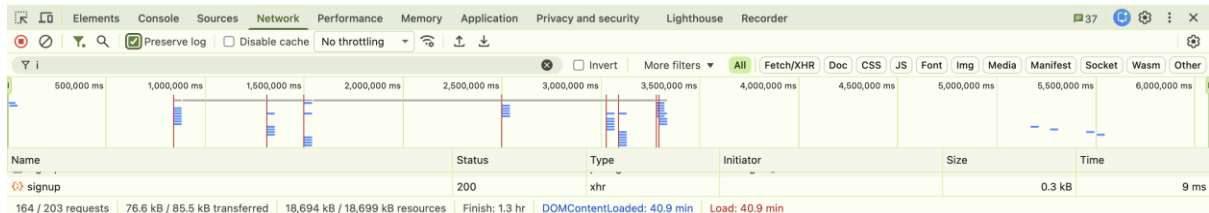
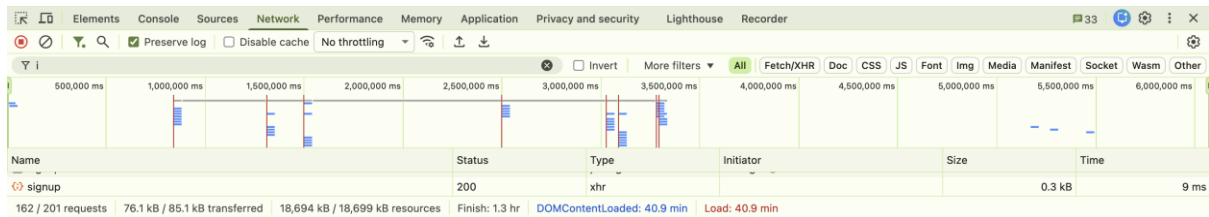
This is the Login page. Now we'll see response after adding username and password.



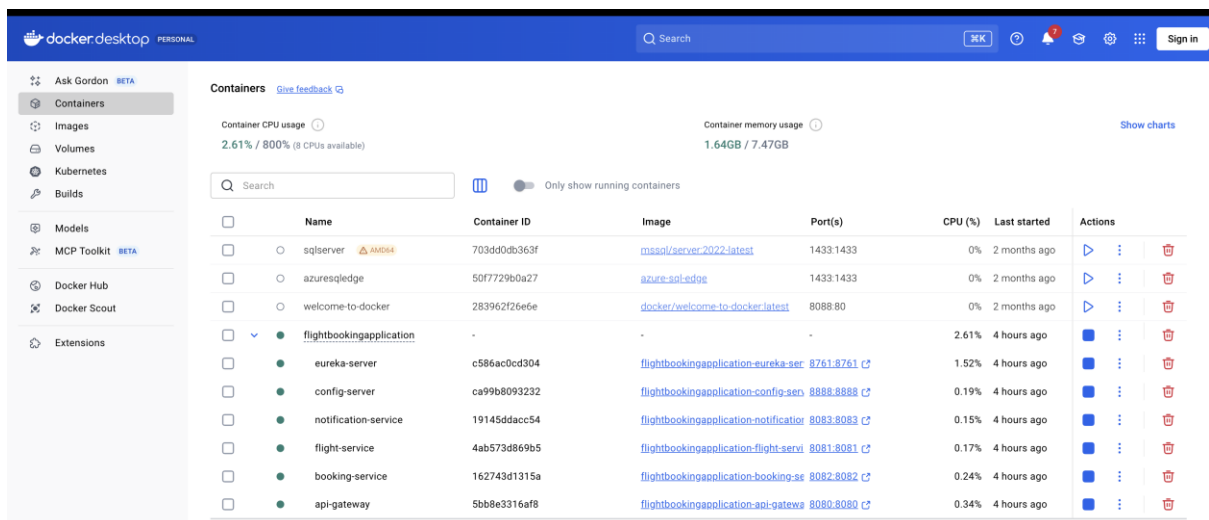
We can see the post status here. It shows the backend is connected successfully.

The post request shows 200OK!

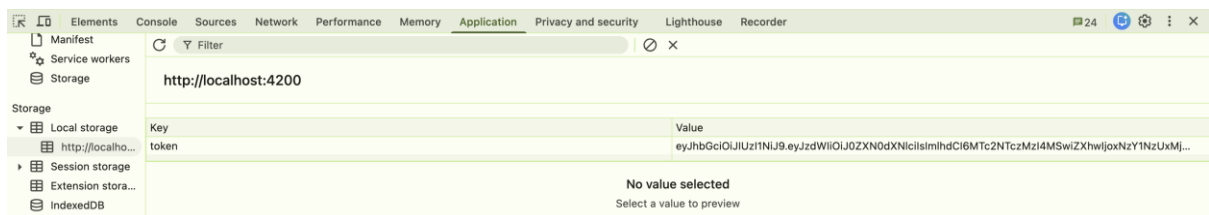
Backend Connection Proof:



The requests used in backend.

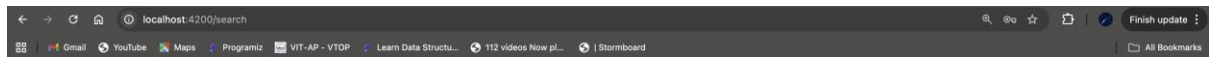


Docker running all the services



Token generated after Login.

Flight Search Page:



Flight Search

Source

Destination

Enter source

Enter destination

Search

No flights found



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



Flight Search

Source

Destination

VJA

BLR

Search

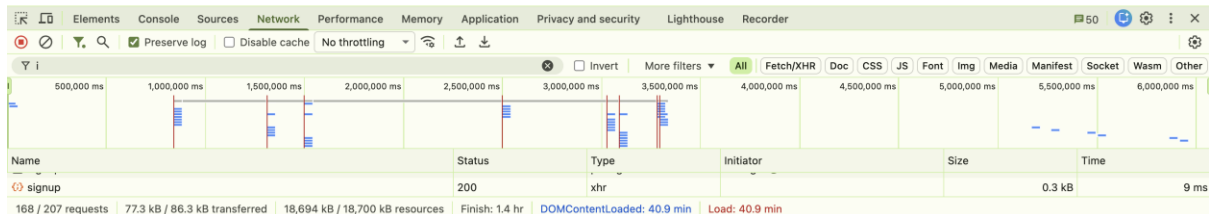
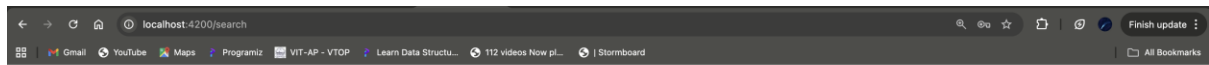
Flight No	Source	Destination	Price (₹)
AI101	VJA	BLR	4500
GO404	VJA	BLR	4800



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:

localhost:8761

GmailYouTubeMapsProgramizVIT-AP - VTOPLearn Data Structu...112 videos Now pl...Stormboard

Finish update

All Bookmarks

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2025-12-14T16:57:33 +0000
Data center	default	Uptime	01:39
		Lease expiration enabled	true
		Renews threshold	8
		Renews (last min)	14

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 5bb8e3316af8:api-gateway:8080
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - 162743d1315a:booking-service:8082
CONFIG-SERVER	n/a (1)	(1)	UP (1) - ca99b8093232:config-server:8888
FLIGHT-SERVICE	n/a (1)	(1)	UP (1) - 4ab573d869b5:FLIGHT-SERVICE:8081

General Info

Name	Value
------	-------

KAFKA:

```
polakujoshikasree@POLAKUs-Laptop FlightBookingApplication % kafka-topics --bootstrap-server localhost:9092 --list

booking-cancelled
booking-created
bookings-topic
cancellations-topic
polakujoshikasree@POLAKUs-Laptop FlightBookingApplication %
```

Notification:

```
polakujoshikasree@POLAKUs-Laptop FlightBookingApplication % brew services list

Name      Status User      File
kafka     started polakujoshikasree ~/Library/LaunchAgents/homebrew.mxcl.kafka.plist
mongodb-community started polakujoshikasree ~/Library/LaunchAgents/homebrew.mxcl.mongodb-community.plist
zookeeper started polakujoshikasree ~/Library/LaunchAgents/homebrew.mxcl.zookeeper.plist
polakujoshikasree@POLAKUs-Laptop FlightBookingApplication % kafka-console-consumer --bootstrap-server localhost:9092 --topic booking-events --from-beginning

Booking Successful - PNR: 38A6510D
```

POSTMAN:

SignUp:

The screenshot shows the Postman interface for a workspace named "Joshika sree Polaku's Workspace". The left sidebar displays a collection of requests, with "SIGNUP" selected under the "new fbp" folder. The main panel shows a POST request to "http://localhost:8080/auth/signup". The request body is a raw JSON object:

```
{  "username": "testuser",  "password": "test123"}
```

. The response status is "200 OK" with a response time of 41 ms and a size of 182 B. The response body is "1 Signup success".

Login:

The screenshot shows the Postman interface for the same workspace. The left sidebar displays a collection of requests, with "New Request" selected under the "new fbp" folder. The main panel shows a POST request to "http://localhost:8080/auth/login". The request body is a raw JSON object:

```
{  "username": "testuser",  "password": "test123"}
```

. The response status is "200 OK" with a response time of 31 ms and a size of 376 B. The response body is a JSON object:

```
{  "username": "testuser",  "role": "CUSTOMER",  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ0IiwiaWF0Ij0iMTc2NTczNDkzNCwiZXhwIjoxNzY1NzUyOTM0LCJyb2x1cyI6WyJ0VWVudT01FUjJ3dGQ5ERAiYjBELLWjAoBx975pYfV9U14TmCvcodsVjN-Dc"}
```


Logout:

The screenshot shows the Postman interface with a workspace named 'Joshika sree Polaku's Workspace'. A collection named 'flightboooking jwt' is selected, containing a 'logout' request. The request is a POST to 'http://localhost:8080/auth/logout'. The headers tab is active, showing 8 headers: Postman-Token, Content-Length (0), Host, User-Agent, Accept, Accept-Encoding (gzip, deflate, br), Connection (keep-alive), and Authorization (Bearer eyJhbGciOiJIUzI1NiU9.eyJzdWIiOiJhZ...).

The response body shows a 200 OK status with a response time of 22 ms and a size of 182 B. The body content is 'Logout success'.

MongoDB:(2 databases)

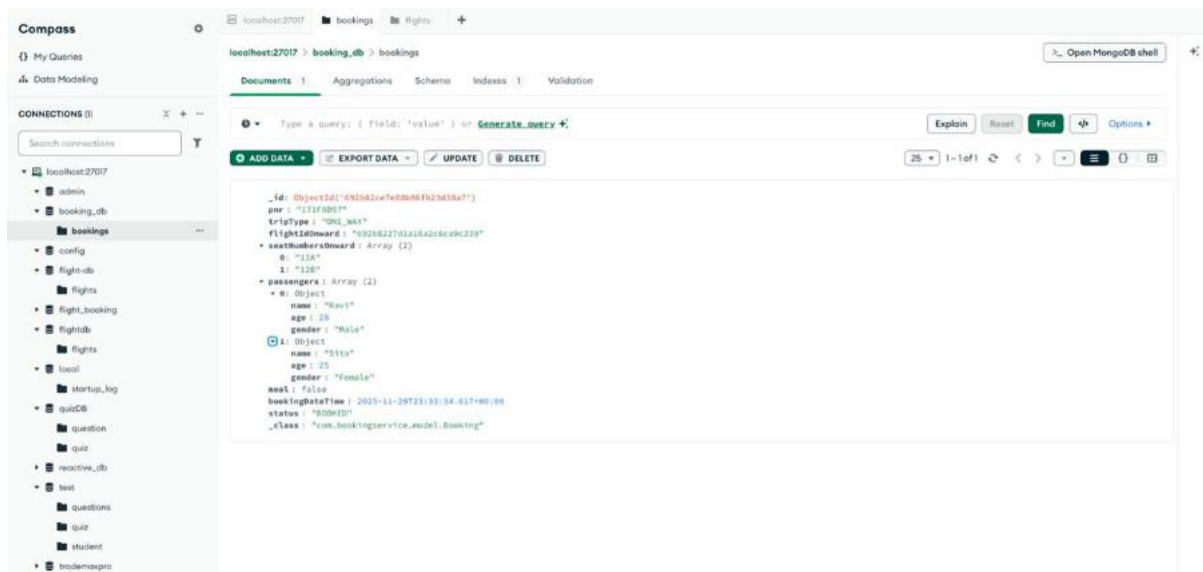
booking_db	20.48 kB	1	1
config	12.29 kB	0	2
flightdb	20.48 kB	1	1

FlightDB:

The screenshot shows the MongoDB Compass interface. The 'flightdb' database is selected, and the 'flights' collection is open. A document is displayed with the following fields:

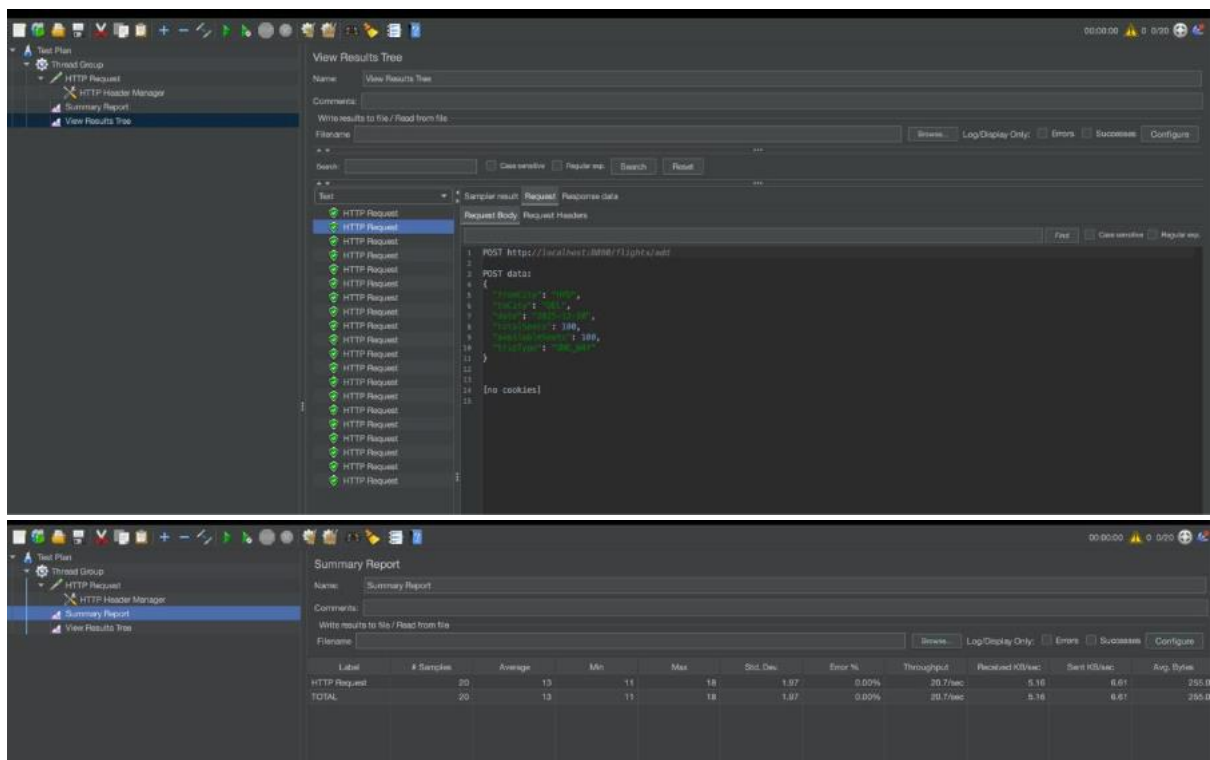
```
{
  "_id": ObjectId("4b0b82701a1a26ca9c238"),
  "fromCity": "HYD",
  "toCity": "DEL",
  "date": "2025-12-18",
  "totalSeats": 100,
  "availableSeats": 100,
  "_class": "com.FlightService.model.Flight"
}
```

BookingDB:

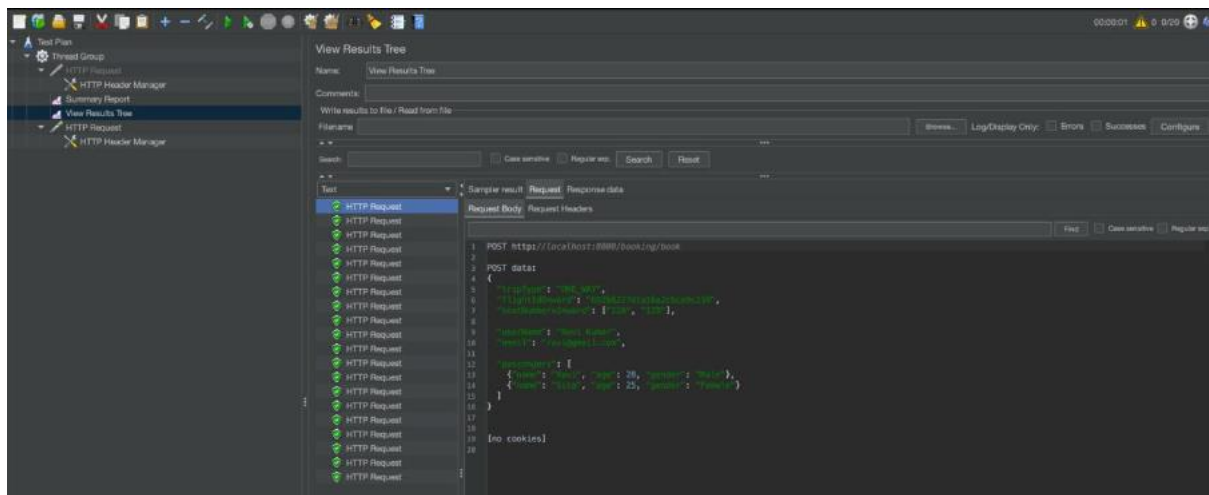


JMeter:

Post request (add inventory):



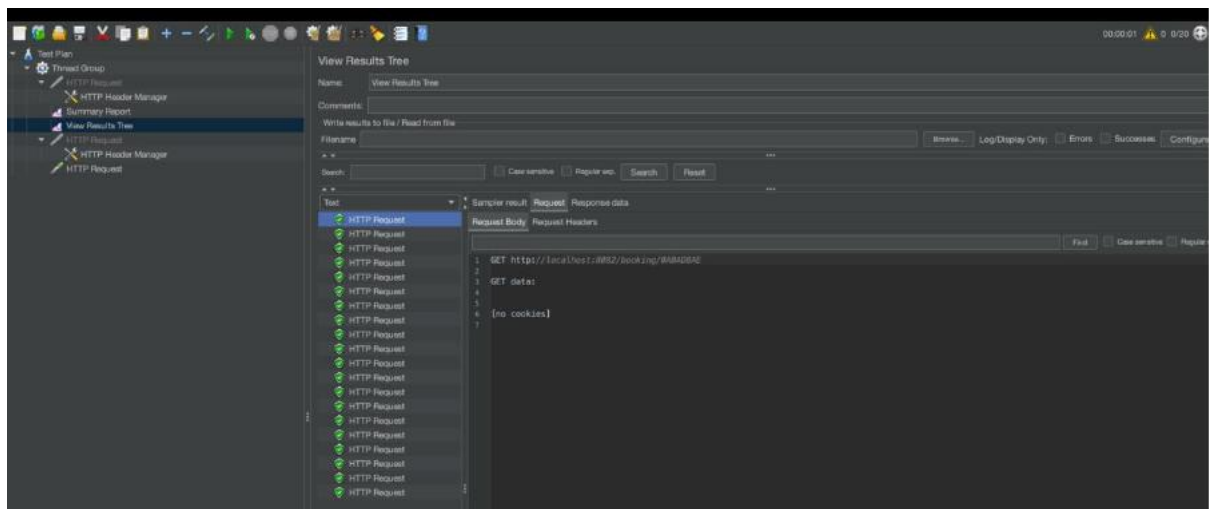
Add Booking :



The screenshot shows the JMeter Summary Report. The table below summarizes the test results.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	20	10	5	16	3.23	100.00%	20.7/sec	6.21	10.89	307.6
TOTAL	20	10	5	16	3.23	100.00%	20.7/sec	6.21	10.89	307.6

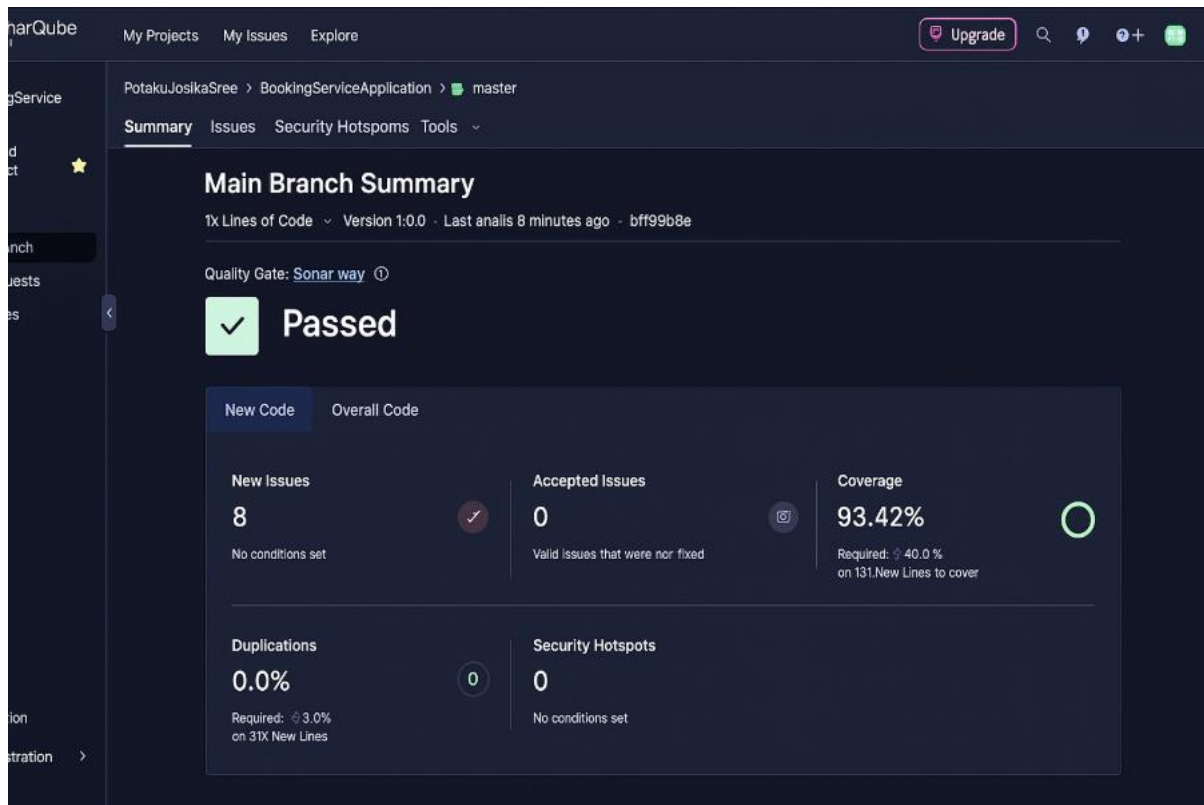
Get PNR:



The screenshot shows the JMeter Summary Report. The table below summarizes the test results.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	50	6	4	11	1.54	0.00%	80.5/sec	31.04	6.50	633.0
TOTAL	50	6	4	11	1.54	0.00%	80.5/sec	31.04	6.50	633.0

Sonar Analysis:



CONCLUSION:

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