

Flightbooking using Micorservices

Eureka server :

<http://localhost:8761>

The screenshot shows a web browser window with the URL localhost:8761 in the address bar. The page content is as follows:

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY-FLIGHTBOOKING	n/a (1)	(1)	UP (1) - shoyomshome.net/api-gateway-flightbooking:8084
BOOKING-SERVICE-FLIGHTBOOKING	n/a (1)	(1)	UP (1) - shoyomshome.net/booking-service-flightbooking:8082
CONFIG-SERVER-FLIGHTBOOKING	n/a (1)	(1)	UP (1) - shoyomshome.net/config-server-flightbooking:8888
FLIGHT-SERVICE-FLIGHTBOOKING	n/a (1)	(1)	UP (1) - shoyomshome.net/flight-service-flightbooking:8081

General Info

Name	Value
total-avail-memory	94mb
num-of-cpus	16
current-memory-usage	51mb (54%)
server-upptime	00:02
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/ ,
available-replicas	

Instance Info

Name	Value
ipAddr	172.25.192.1
status	UP

Postman :

Using api gateway- port-8084

Using mongoDb

Add inventory:

The screenshot shows the Postman application interface. At the top, there's a header bar with the title "flightApp-microservices / adding flight" and various buttons for "Save", "Share", and "Send". Below the header, the main area has a "POST" method selected and the URL "http://localhost:8084/api/flight/airline/inventory/add". The "Body" tab is active, showing a JSON payload with line numbers 1 through 12. The payload is as follows:

```
1 {  
2   "airlineName": "Indigo",  
3   "fromPlace": "Delhi",  
4   "toPlace": "Hyderabad",  
5   "departureTime": "2026-12-11T10:00:00",  
6   "arrivalTime": "2026-12-11T12:00:00",  
7   "tripType": "ONE-WAY",  
8   "priceOneWay": 3500,  
9   "priceRoundTrip": 0,  
10  "totalSeats": 120  
11 }  
12
```

Below the body, the response section shows a green "201 Created" status with a timestamp of "52 ms" and a size of "145 B". There are also buttons for "Save Response" and other options like "Raw", "Preview", and "Visualize". The response body itself contains the ID "6935c4e807a4660504087bdf".

Search flights:

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** <http://localhost:8084/api/flight/search>
- Body:** JSON (raw) content:

```
1 {  
2   "fromPlace": "Hyderabad",  
3   "toPlace": "Bangalore",  
4   "journeyDate": "2025-12-10",  
5   "tripType": "ONE-WAY"  
6 }  
7
```
- Response:** 200 OK, 39 ms, 384 B. The response body is:

```
1 [  
2   {  
3     "flightId": "6930cbd877157c2d5b251bf5",  
4     "departureTime": "2025-12-10T10:00:00",  
5     "arrivalTime": "2025-12-10T12:00:00",  
6     "airlineName": "Indigo",  
7     "priceOneWay": 3500.0,  
8     "priceRoundTrip": 0.0,  
9     "tripType": "ONE-WAY",  
10    "availableSeats": 110,  
11    "fromPlace": "Hyderabad",  
12    "toPlace": "Bangalore"  
13  }  
14]
```

Get flight using id:

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** <http://localhost:8084/api/flight/693131a9b290ee4e47e0447a>
- Params:** (empty)
- Response:** 200 OK, 22 ms, 382 B. The response body is:

```
1 {  
2   "flightId": "693131a9b290ee4e47e0447a",  
3   "departureTime": "2025-12-11T10:00:00",  
4   "arrivalTime": "2025-12-11T12:00:00",  
5   "airlineName": "Indigo",  
6   "priceOneWay": 3500.0,  
7   "priceRoundTrip": 0.0,  
8   "tripType": "ONE-WAY",  
9   "availableSeats": 118,  
10  "fromPlace": "Hyderabad",  
11  "toPlace": "Bangalore"  
12 }
```

Booking using id:

The screenshot shows the Postman application interface. At the top, the URL is `http://localhost:8084/api/booking/6930cbd877157c2d5b251bf5`. The method is set to `POST`, and the response status is `201 Created`. The response body contains a detailed JSON object representing the booking information, including PNR, customer details, travel dates, times, and seat numbers.

```
1 {  
2   "customerName": "Shivani",  
3   "email": "shivani@gmail.com",  
4   "numberOfSeats": 2,  
5   "journeyDate": "2025-12-10",  
6   "seatNumbers": "12A,12B",  
7 }  
  
1 {  
2   "pnr": "DPGRH3MA",  
3   "customerName": "Shivani",  
4   "email": "shivani@gmail.com",  
5   "fromPlace": "Hyderabad",  
6   "toPlace": "Bangalore",  
7   "journeyDate": "2025-12-10",  
8   "departureTime": "2025-12-10T10:00:00",  
9   "arrivalTime": "2025-12-10T12:00:00",  
10  "airlineName": "Indigo",  
11  "numberOfSeats": 2,  
12  "seatNumbers": "12A,12B",  
13  "mealPreference": "VEG",  
14  "cancelled": false,  
15  "bookingTime": "2025-12-07T23:51:17.0422246".  
}
```

Get ticket by pnr :

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8084/api/booking/ticket/7WBAU7KZ`
- Method:** GET
- Headers:** (7 items listed)
- Body:** (Empty JSON object)
- Response Status:** 200 OK
- Response Time:** 32 ms
- Response Size:** 572 B
- Response Content:**

```
1 {  
2   "pnr": "7WBAU7KZ",  
3   "customerName": "Shivani",  
4   "email": "shivani@gmail.com",  
5   "fromPlace": "Hyderabad",  
6   "toPlace": "Bangalore",  
7   "journeyDate": "2025-12-11",  
8   "departureTime": "2025-12-11T10:00:00",  
9   "arrivalTime": "2025-12-11T12:00:00",  
10  "airlineName": "Indigo",  
11  "numberOfSeats": 2,  
12  "seatNumbers": "12A,12B",  
13  "mealPreference": "VEG",  
14  "cancelled": false,  
15  "bookingTime": "2025-12-04T12:32:11.674",  
16  "passengers": [  
17    {"name": "Shivani", "age": 25, "gender": "Female"},  
18    {"name": "Rahul", "age": 28, "gender": "Male"}  
19  ]  
20}
```

Get history by email:

The screenshot shows the Postman interface with a GET request to `http://localhost:8084/api/booking/history/sravya@gmail.com`. The response is a 200 OK status with a JSON payload containing flight booking details:

```
1 [  
2 {  
3   "pnr": "1XQ5F3HE",  
4   "customerName": "Sravya",  
5   "email": "sravya@gmail.com",  
6   "fromPlace": "Chennai",  
7   "toPlace": "Bangalore",  
8   "journeyDate": "2025-12-10",  
9   "departureTime": "2025-12-10T10:00:00",  
10  "arrivalTime": "2025-12-10T12:00:00",  
11  "airlineName": "Indigo",  
12  "numberOfSeats": 2,  
13  "status": "Booked"
```

Cancel by pnr:

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8084/api/booking/cancel/DPGRH3MA`. The response is a 200 OK status with the message:

```
1 Ticket cancelled successfully
```

Jmeter:

Search flights:

The screenshot shows the Apache JMeter interface. On the left, the Test Plan tree view has a Thread Group named "search". The main panel displays the "HTTP Request" configuration. The "Basic" tab is selected, showing the URL as "http://localhost:8084/api/v1/flight/search". The "Advanced" tab is also visible. Below the URL, there is a "Parameters" section containing a JSON payload:

```
1  "fromPlace": "Hyderabad",
2  "toPlace": "Bangalore",
3  "journeyDate": "2025-12-10",
4  "tripType": "ONE WAY"
5
6
```

20 samples:

The screenshot shows the Apache JMeter interface. The Test Plan tree view includes two Thread Groups: one with 20 samples and another with 5 samples. The summary report table shows the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
search	20	5	4	9	1.37	0.00%	20.7/sec	8.03	6.78	396.5
TOTAL	20	5	4	9	1.37	0.00%	20.7/sec	8.03	6.78	396.5

50 samples:

The screenshot shows the Apache JMeter interface. The Test Plan tree view includes two Thread Groups: one with 50 samples and another with 5 samples. The summary report table shows the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
search	50	5	4	8	0.87	0.00%	50.7/sec	19.66	16.57	397.4
TOTAL	50	5	4	8	0.87	0.00%	50.7/sec	19.66	16.57	397.4

100 samples:

The screenshot shows the Apache JMeter interface. The Test Plan tree view includes two Thread Groups: one with 100 samples and another with 5 samples. The summary report table shows the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
search	100	5	3	9	0.99	0.00%	100.0/sec	38.75	32.71	396.9
TOTAL	100	5	3	9	0.99	0.00%	100.0/sec	38.75	32.71	396.9

Get flight details by id:

The screenshot shows the Apache JMeter interface with a single test plan containing a Thread Group. The Thread Group has one HTTP Request sampler named "get details by id". The sampler is configured with a GET method, protocol http, server name localhost, port 8084, and path /api/v1.0/flight/6930db8025b45b0fe4b8015d. The "Follow Redirects" checkbox is checked.

20 samples:

The screenshot shows the Apache JMeter interface after running a 20-sample load test. The results are displayed in a Summary Report table. The table includes columns for Label, # Samples, Average, Min, Max, Std. Dev., Error %, Throughput, Received KB/sec, Sent KB/sec, and Avg. Bytes. The data shows 20 samples with an average response time of 6 ms, a minimum of 5 ms, and a maximum of 20 ms. The throughput is 20.9/sec, and the average bytes transferred are 392.0.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	20	6	5	20	3.22	0.00%	20.9/sec	7.08	4.26	392.0
TOTAL	20	6	5	20	3.22	0.00%	20.9/sec	7.08	4.26	392.0

50 samples:

The screenshot shows the Apache JMeter interface after running a 50-sample load test. The results are displayed in a Summary Report table. The data shows 50 samples with an average response time of 6 ms, a minimum of 5 ms, and a maximum of 13 ms. The throughput is 50.8/sec, and the average bytes transferred are 392.0.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
get details by id	50	6	5	13	1.48	0.00%	50.8/sec	19.43	10.61	392.0
TOTAL	50	6	5	13	1.48	0.00%	50.8/sec	19.43	10.61	392.0

100 samples:

The screenshot shows the Apache JMeter interface after running a 100-sample load test. The results are displayed in a Summary Report table. The data shows 100 samples with an average response time of 5 ms, a minimum of 4 ms, and a maximum of 15 ms. The throughput is 100.2/sec, and the average bytes transferred are 392.1.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
get details by id	100	5	4	15	1.30	0.00%	100.2/sec	38.36	20.94	392.1
TOTAL	100	5	4	15	1.30	0.00%	100.2/sec	38.36	20.94	392.1

