

CHUBB®

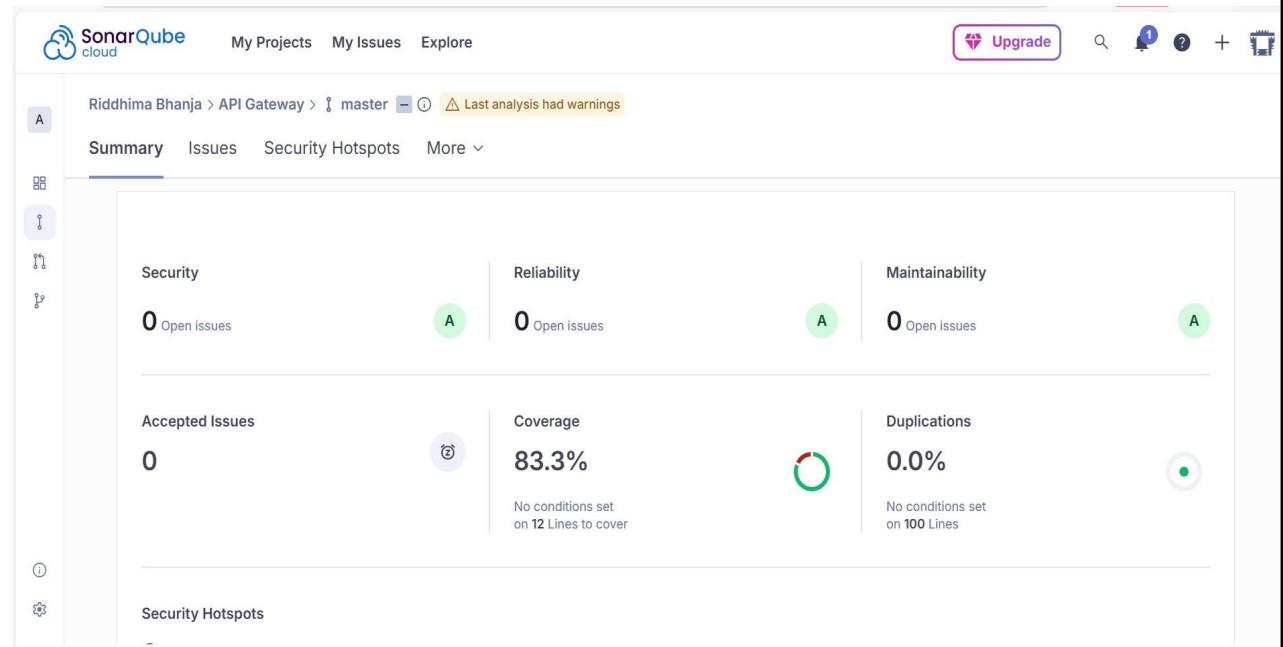
WEEK-6 ASSIGNMENT

-Riddhima Bhanja
Kalinga Institute of Industrial Technology
Bhubaneswar(Java Track)

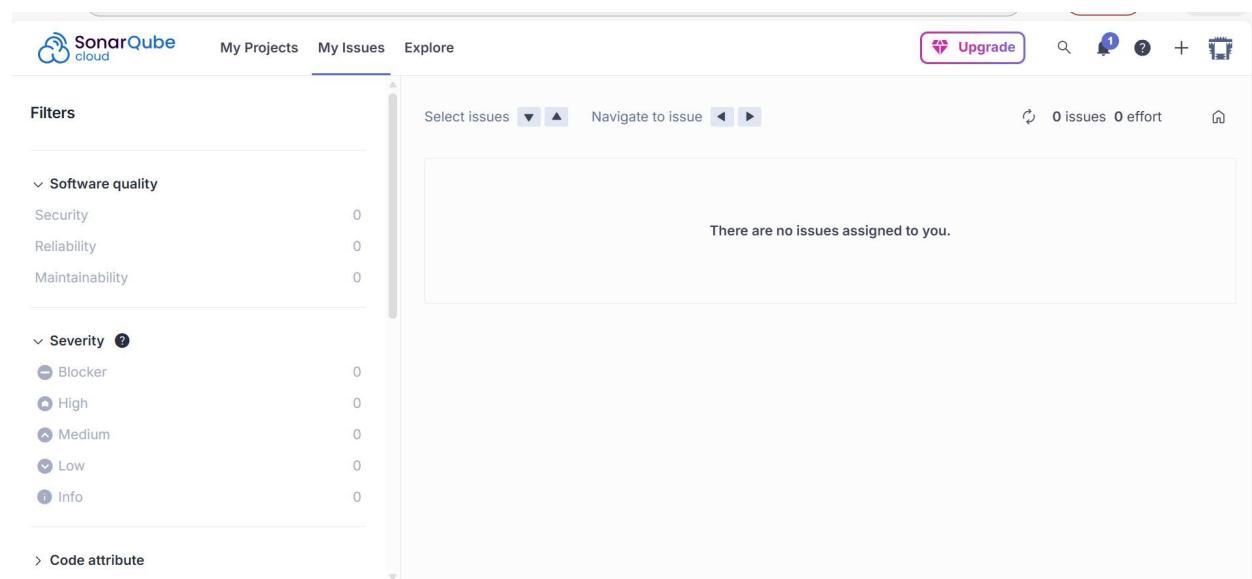
INDEX

Content	Page No.
1. SonarQube Code Coverage	1
2. SonarQube Issues	3
3. System Architecture	5
4. JMeter	6
— JMeter Result Tree	6
— JMeter Summary Report	6
— Apache JMeter Dashboard	7
5. Jacoco Code Coverage	9
— API Gateway	9
— Booking Service	10
— Flight Service	10
6. MongoDB Screenshots	11
7. Postman Screenshots	11
— Circuit Breaker	11
— Events	12
— Status	12
— Health Checks	13
— API Gateway Health	13
— Booking Service Health	13
— Eureka Server Health	14
— Flight Service Health	14
— Add Flight	15
— Book Flight	15
— Cancel Booking	15
— Get Booking by PNR	16
— Get Booking History	16
8. Email	17
— With PDF	17
— Without PDF	17
— Fallback Case	17
9. Eureka Dashboard	17

1. SonarQUBE Code Coverage



2. SonarQube Issues



3. System Architecture



4. Jmeter

- Apache Jmeter Dashboard

The screenshot shows the Apache JMeter Dashboard interface. On the left, there's a sidebar with links for Dashboard, Charts, and Custom Graphs. The main area has several sections:

- Test and Report information:** Shows details like Source file ("results_20251201_210227.jtl"), Start Time ("12/1/25, 9:02 PM"), End Time ("12/1/25, 9:02 PM"), and Filter for display ("").
- APDEX (Application Performance Index):** A table showing APdex values for different labels. For "Total", APdex is 0.989, T is 500 ms, and F is 1 sec 500 ms. For "Search Flights", APdex is 0.989, T is 500 ms, and F is 1 sec 500 ms.
- Requests Summary:** A donut chart indicating 0.8% FAIL and 99.2% PASS.
- Statistics:** A table with tabs for Requests, Executions, Response Times (ms), Throughput, and Network (KB/sec). The Response Times (ms) tab shows detailed metrics for Average, Min, Max, Median, 90th pct, 95th pct, 99th pct, Transactions/s, Received, and Sent.

This part of the dashboard continues from the previous section:

- Statistics:** A detailed table for Response Times (ms) showing data for Total and Search Flights samples.
- Errors:** A table showing errors categorized by type. One entry is 405/Method Not Allowed with 4 errors, 100.00% in errors, and 0.80% in all samples.
- Top 5 Errors by sampler:** A table showing errors grouped by sampler. It lists two samplers: Total and Search Flights, each with 4 errors of type 405/Method Not Allowed.

5. Mongodb screenshots

- Booking_db

The screenshot shows the MongoDB Compass interface for the `booking_db` database. The `booking` collection is selected. Two documents are displayed:

```
_id: ObjectId('692d8254fd1536848c1d1b37')
pnr : "PNR252ABD9"
flightId : "692d822651cef85aec2564cb"
flightNumber : "A1101"
airline : "Air India"
fromPlace : "Delhi"
toPlace : "Mumbai"
departureDateTime : 2024-12-15T04:30:00.000+00:00
arrivalDateTime : 2024-12-15T07:00:00.000+00:00
userName : "John Doe"
userEmail : "john.doe@example.com"
journeyDate : 2024-12-14T18:30:00.000+00:00
noOfSeats : 2
mealType : "VEG"
totalAmount : 10000
bookingStatus : "CANCELLED"
bookingDateTime : 2025-12-01T11:56:04.147+00:00
* passengers: Array (2)
  _class : "com.flighthapp.booking.entity.Booking"

_id: ObjectId('692d8280fd1536848c1d1b38')
pnr : "PNRBD545"
flightId : "692d822651cef85aec2564cb"
flightNumber : "A1101"
airline : "Air India"
fromPlace : "Delhi"
```

- Flight db

The screenshot shows the MongoDB Compass interface for the `flight_db` database. The `flight_inventory` collection is selected. One document is displayed:

```
_id: ObjectId('692d822651cef85aec2564cb')
airline : "Air India"
flightNumber : "A1101"
fromPlace : "Delhi"
toPlace : "Mumbai"
departureDateTime : 2024-12-15T04:30:00.000+00:00
arrivalDateTime : 2024-12-15T07:00:00.000+00:00
totalSeats : 180
availableSeats : 174
ticketPrice : 5000
flightStatus : "ACTIVE"
oneWayPrice : 5000
roundTripPrice : 9000
mealAvailable : true
_class : "com.flighthapp.flight.entity.FlightInventory"
```

6. Email

- Email with pdf

Your e-ticket and booking details [Inbox](#)

96 riddhimabhanja2003@gmail.com to me 5:38 PM (4 hours ago)

Hi Riddhima,

Please find your ticket attached (PNR PNR123456).

Flight: XY123

Seat: 12A

Regards,
Flight Booking Team

One attachment · Scanned by Gmail Add to Drive

ticket_PNR12345...

Reply Forward

- Email without pdf

Your Ticket Update [Inbox](#)

4 of 335 riddhimabhanja2003@gmail.com to me 5:38 PM (4 hours ago)

Hello, Riddhima

Your flight has been confirmed. Thank you for choosing us!

Regards,
Flight Booking Team

Reply Forward

- Flight cancel email

The screenshot shows a Gmail inbox with 96 messages. The message subject is "Flight Booking Cancelled" from "riddhimabhanja2003@gmail.com" to "john.doe". The message body states: "Dear John Doe, Your flight booking has been cancelled. Booking Details: PNR: PNR45010D76 Flight: AI101 Route: Delhi to Mumbai Departure: 2024-12-15T10:00 Total Amount: \$10000.0 Thank you for choosing our service! Best regards, Flight Booking Team". The timestamp is 9:57 PM (32 minutes ago).

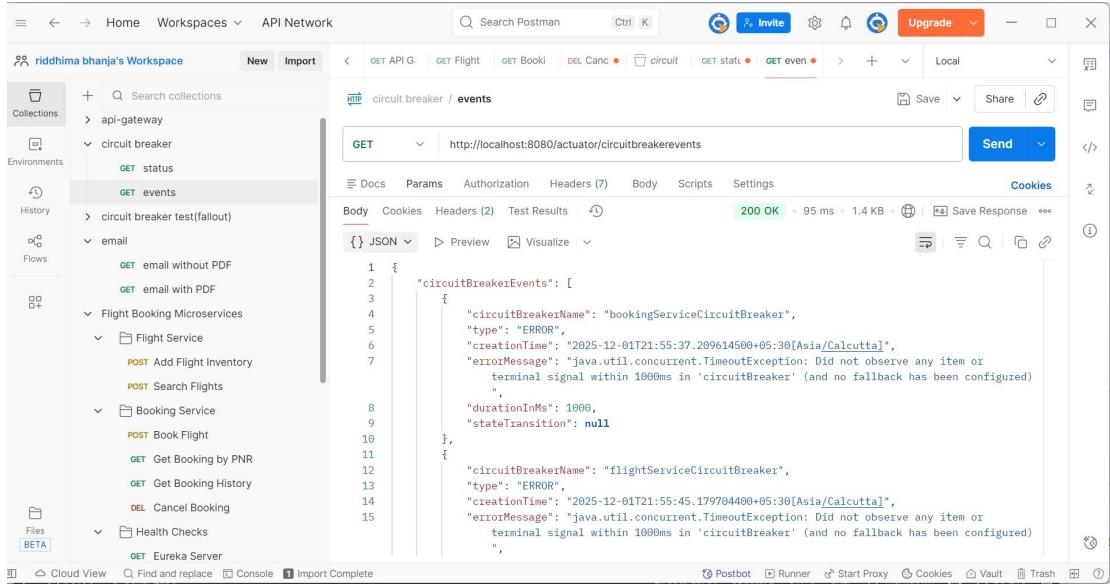
- Flight Confirm email

The screenshot shows a Gmail inbox with 96 messages. The message subject is "Flight Booking Confirmed" from "riddhimabhanja2003@gmail.com" to "john.doe". The message body states: "Dear John Doe, Your flight booking has been confirmed. Booking Details: PNR: PNR45010D76 Flight: AI101 Route: Delhi to Mumbai Departure: 2024-12-15T10:00 Total Amount: \$10000.0 Thank you for choosing our service! Best regards, Flight Booking Team". The timestamp is 5:55 PM (4 hours ago).

7. Postman screenshots

- Circuit Breaker**

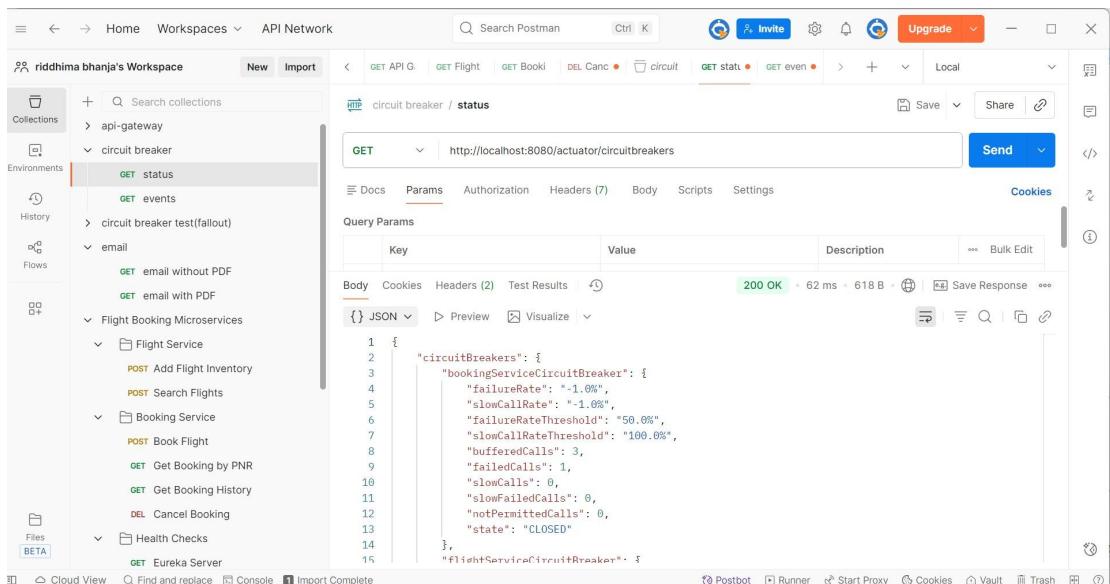
Events:



The screenshot shows the Postman interface with the URL `http://localhost:8080/actuator/circuitbreakerevents`. The response body is a JSON object containing two entries for circuit breaker events. The first entry is for the `bookingServiceCircuitBreaker`, which failed at 2025-12-01T21:55:37.209614500+05:30[Asia/Calcutta]. The second entry is for the `flightServiceCircuitBreaker`, which failed at 2025-12-01T21:55:45.179704400+05:30[Asia/Calcutta]. Both entries indicate they did not observe any item or terminal signal within 1000ms.

```
1  {
2      "circuitBreakerEvents": [
3          {
4              "circuitBreakerName": "bookingServiceCircuitBreaker",
5              "type": "ERROR",
6              "creationTime": "2025-12-01T21:55:37.209614500+05:30[Asia/Calcutta]",
7              "errorMessage": "java.util.concurrent.TimeoutException: Did not observe any item or terminal signal within 1000ms in 'circuitBreaker' (and no fallback has been configured)",
8          },
9          {
10             "durationInMs": 1000,
11             "stateTransition": null
12         },
13         {
14             "circuitBreakerName": "flightServiceCircuitBreaker",
15             "type": "ERROR",
16             "creationTime": "2025-12-01T21:55:45.179704400+05:30[Asia/Calcutta]",
17             "errorMessage": "java.util.concurrent.TimeoutException: Did not observe any item or terminal signal within 1000ms in 'circuitBreaker' (and no fallback has been configured)"
18         }
19     ]
20 }
```

Status:



The screenshot shows the Postman interface with the URL `http://localhost:8080/actuator/circuitbreakers`. The response body is a JSON object containing information about the `bookingServiceCircuitBreaker` and `flightServiceCircuitBreaker`. The `bookingServiceCircuitBreaker` has a failure rate of 1.0% and a slow call rate of 1.0%. The `failureRateThreshold` is 50.0%, and the `slowCallRateThreshold` is 100.0%. It has 3 buffered calls, 1 failed call, 0 slow calls, and 0 slow failed calls. The state is `CLOSED`. The `flightServiceCircuitBreaker` is not shown in the current view.

```
1  {
2      "circuitBreakers": [
3          {
4              "bookingServiceCircuitBreaker": {
5                  "failureRate": "+1.0%",
6                  "slowCallRate": "+1.0%",
7                  "failureRateThreshold": "50.0%",
8                  "slowCallRateThreshold": "100.0%",
9                  "bufferedCalls": 3,
10                 "failedCalls": 1,
11                 "slowCalls": 0,
12                 "slowFailedCalls": 0,
13                 "notPermittedCalls": 0,
14                 "state": "CLOSED"
15             }
16         }
17     ]
18 }
```

- Health Check

API gateway health:

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Collection:** Flight Booking Microservices
- Request:** GET /actuator/health
- Response Status:** 200 OK
- Response Body (JSON):**

```
{
  "status": "UP",
  "components": {
    "discoveryComposite": {
      "status": "UP",
      "components": {
        "discoveryClient": {
          "status": "UP",
          "details": {
            "services": [
              "api-gateway",
              "flight-service",
              "booking-service"
            ]
          }
        }
      }
    }
  }
}
```

Booking Service Health:

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Collection:** Flight Booking Microservices
- Request:** GET /actuator/health
- Response Status:** 200 OK
- Response Body (JSON):**

```
{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UP",
      "details": {
        "flightService": {
          "status": "UP",
          "details": {
            "failureRate": "-1.0%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "-1.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 0,
            "slowCalls": 0
          }
        }
      }
    }
  }
}
```

Eureka server:

The screenshot shows the Postman application interface. On the left, the sidebar displays 'riddhima bhanja's Workspace' with various collections, environments, and flows. The main workspace shows a 'Flight Booking Microservices / Health Checks / Eureka Server' collection. A 'GET' request is selected with the URL `http://localhost:8761/`. The response status is 200 OK, with a response time of 24 ms and a size of 5.47 KB. The response body is displayed as HTML, showing the Eureka Server's health check page.

```
<!doctype html>
<html lang="en">
<title>Eureka</title>
<meta name="description" content="">
<meta name="viewport" content="width=device-width">
<base href="/">
<meta charset="utf-8">
<link rel="stylesheet" href="eureka/css/wro.css">
```

Flight Service Health:

The screenshot shows the Postman application interface. The sidebar is identical to the previous one. The main workspace shows a 'Flight Booking Microservices / Health Checks / Flight Service Health' collection. A 'GET' request is selected with the URL `http://localhost:8081/actuator/health`. The response status is 200 OK, with a response time of 35 ms and a size of 1.03 KB. The response body is displayed as JSON, showing the health status of the discovery composite and its components.

```
{
  "status": "UP",
  "components": {
    "discoveryComposite": {
      "status": "UP",
      "components": {
        "discoveryClient": {
          "status": "UP",
          "details": {
            "services": [
              "api-gateway",
              "flight-service",
              "booking-service"
            ]
          }
        }
      }
    }
}
```

ADD FLIGHT

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- API:** POST Add Flight Inventory
- URL:** http://localhost:8080/api/v1/flight/inventory
- Body (raw JSON):**

```
1 {
2   "airline": "Air India",
3   "flightNumber": "AI101",
4   "fromPlace": "Delhi",
5   "toPlace": "Mumbai",
6   "departureDateTime": "2024-12-15T10:00:00",
```

- Response:** 201 Created (88 ms, 425 B)

BOOK FLIGHT

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- API:** POST Book Flight
- URL:** http://localhost:8080/api/v1/booking/book/:flightId
- Path Variables:**

Key	Value	Description
flightId	692d822651cef85aec2564cb	Description

- Response:** 200 OK (92 ms, 717 B)

SEARCH FLIGHTS

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request Type:** POST
- URL:** http://localhost:8080/api/v1/flight/search
- Body (JSON):** (Empty)
- Response:** 200 OK (360 ms, 429 B)
 - Preview: A JSON object representing a flight search result.

```
1 [  
2 {  
3   "id": "692d022651cef05aec2564cb",  
4   "airline": "Air India",  
5   "flightNumber": "AI101",  
6   "fromPlace": "Delhi",  
7   "toPlace": "Mumbai",  
8   "departureDateTime": "2024-12-15T10:00:00",  
9   "arrivalDateTime": "2024-12-15T12:30:00",  
10  "totalSeats": 180,  
11  "availableSeats": 180,  
12  "ticketPrice": 5000.0,  
13  "flightStatus": "ACTIVE",  
14  "oneWayPrice": 5000.0,  
15  "roundTripPrice": 9000.0,  
..]
```

CANCEL BOOKING

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request Type:** DELETE
- URL:** http://localhost:8080/api/v1/booking/cancel/:pnr
- Headers (8):** (Hidden)
- Body (JSON):** (Empty)
- Response:** 200 OK (201 ms, 712 B)
 - Preview: A JSON object representing a booking cancellation result.

```
1 {  
2   "pnr": "PNR45010D76",  
3   "flightId": "692d022651cef05aec2564cb",  
4   "flightNumber": "AI101",  
5   "airline": "Air India",  
6   "fromPlace": "Delhi",  
7   "toPlace": "Mumbai",  
8   "departureDateTime": "2024-12-15T10:00:00",  
9   "arrivalDateTime": "2024-12-15T12:30:00",  
10  "userName": "John Doe",  
11  "userEmail": "john.doe@example.com",  
12  "journeyDate": "2024-12-15",  
13  "noOfSeats": 2,  
14  "mealType": "VEG",  
15  "totalAmount": 10000.0,  
..]
```

GET BOOKING BY PNR

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/v1/booking/:pnr`. The response body is:

```
1 {  
2   "pnr": "PNR252ABD96",  
3   "flightId": "692d822651ce1f85aec2564cb",  
4   "flightNumber": "AI101",  
5   "airline": "Air India",  
6   "fromPlace": "Delhi",  
7   "toPlace": "Mumbai",  
8   "departureDateTime": "2024-12-15T10:00:00",  
9   "arrivalDateTime": "2024-12-15T12:30:00",  
10  "userName": "John Doe"
```

GET BOOKING HISTORY

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/v1/booking/history/:email`. The response body is:

```
11 {  
12   "userName": "John Doe",  
13   "userEmail": "john.doe@example.com",  
14   "journeyDate": "2024-12-15",  
15   "noOfSeats": 2,  
16   "mealType": "VEG",  
17   "totalAmount": 10000.0,  
18   "bookingStatus": "CANCELLED",  
19   "bookingDateTime": "2025-12-01T17:26:04.147",  
20   "passenger": [
```

FALLBACK case

The screenshot shows the Postman application interface. On the left, the sidebar displays various collections, environments, and flows. In the center, a specific API endpoint for 'Flight Booking Microservices / Booking Service / Book Flight' is selected. The request method is set to 'POST' and the URL is `http://localhost:8080/api/v1/booking/book/:flightId`. The 'Params' tab shows a parameter named 'flightId' with the value '692d822651cef85aec2564cb'. The 'Headers' tab contains a single header 'Content-Type: application/json'. The 'Body' tab is set to 'JSON' and contains the following JSON response:

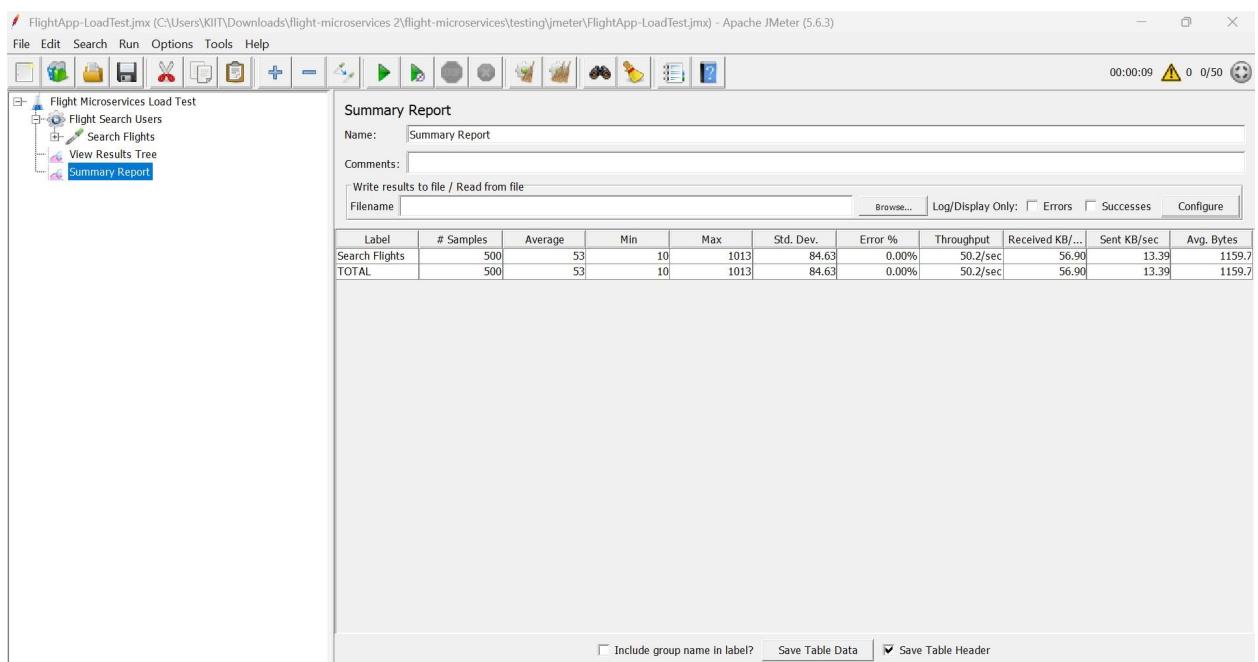
```
1 "timestamp": "2025-12-01T17:31:18.341+00:00",
2 "path": "/api/v1/booking/book/692d822651cef85aec2564cb",
3 "status": 405,
4 "error": "Method Not Allowed",
5 "requestId": "0b378212-10"
```

The status bar at the bottom indicates a '405 Method Not Allowed' error with a response time of 1.04 s and a size of 256 B.

- Jmeter result tree**

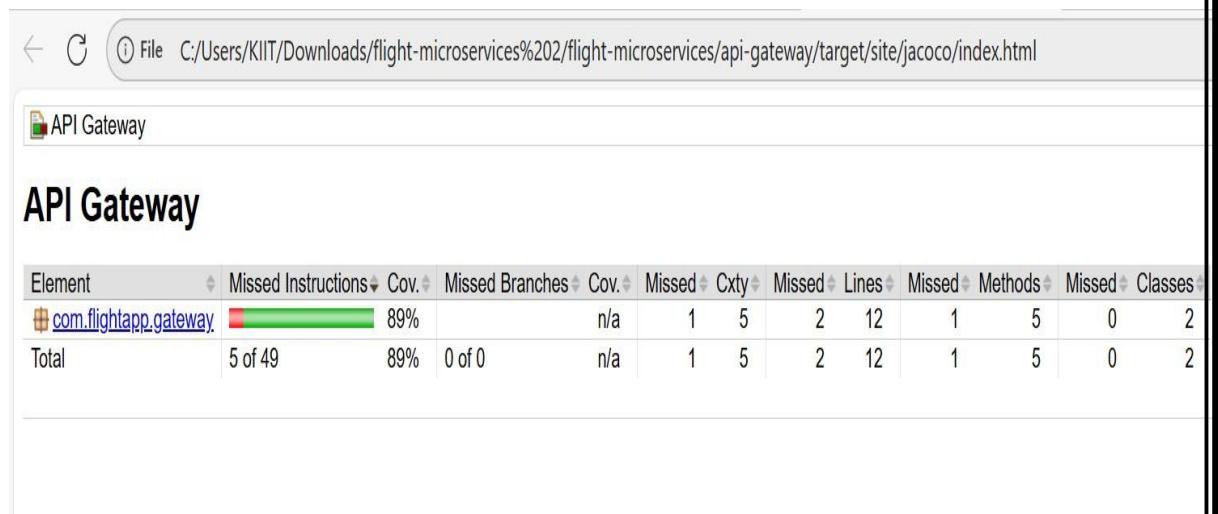
The screenshot shows the Apache JMeter interface. The top menu bar includes File, Edit, Search, Run, Options, Tools, and Help. The main window displays a 'View Results Tree' panel. The tree structure shows multiple 'Search Flights' entries, each with a green checkmark indicating success. The 'Text' dropdown is set to 'Sampler result'. At the bottom of the panel, there are tabs for 'Raw' and 'Parsed' data. The status bar at the bottom right shows '00:00:09' and '0 0/50'.

- ◆ Jmeter Summary report

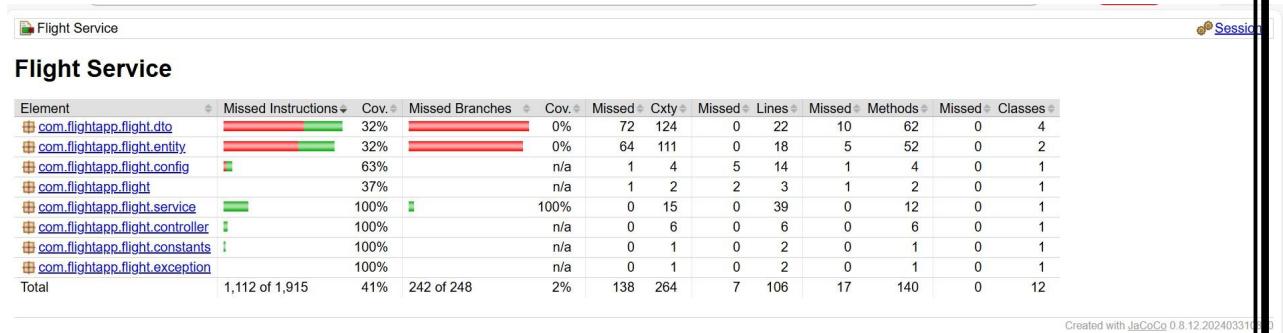


8. Jacoco Code Coverage

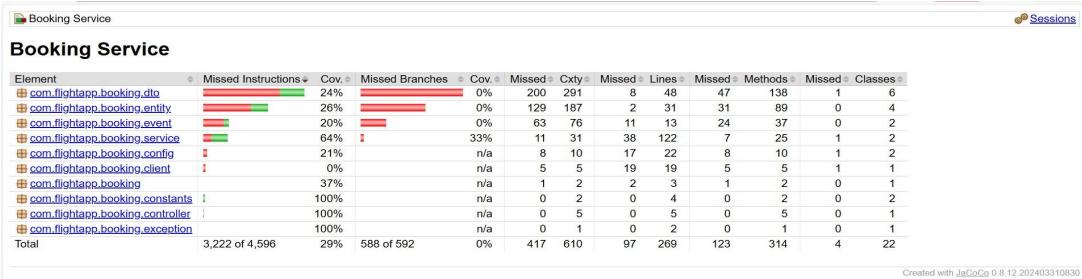
- API Gateway



- Flight Service



- Booking Service



9. EUREKA DASHBOARD

spring Eureka HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2025-12-01T20:47:22+0530
Data center	default	Uptime	00:03
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	3

THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - localhost:api-gateway:8080
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - booking-service:8082
FLIGHT-SERVICE	n/a (1)	(1)	UP (1) - flight-service:8081

General Info

Name	Value
total-available-memory	94mb
num-of-cpus	8
current-memory-usage	64mb (68%)
server-upptime	00:03
registered-replicas	
unavailable-replicas	
available-replicas	

Instance Info

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