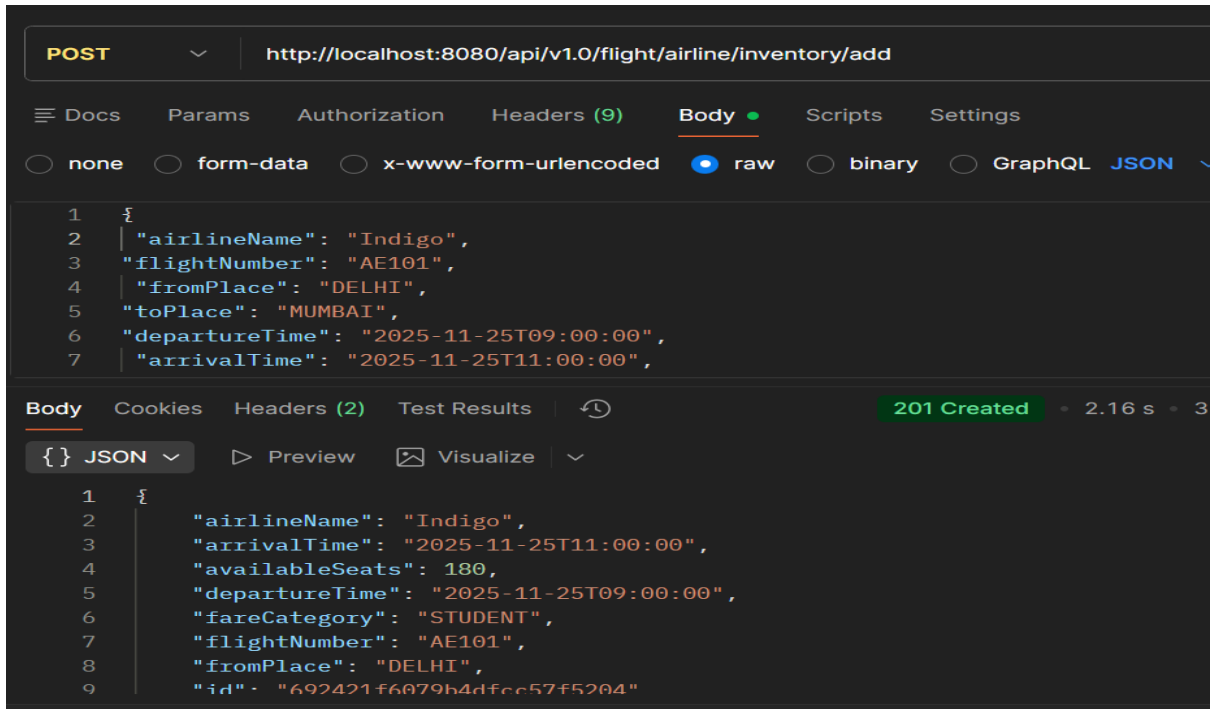


Flight Management using webflux and MongoDB(Screenshots)

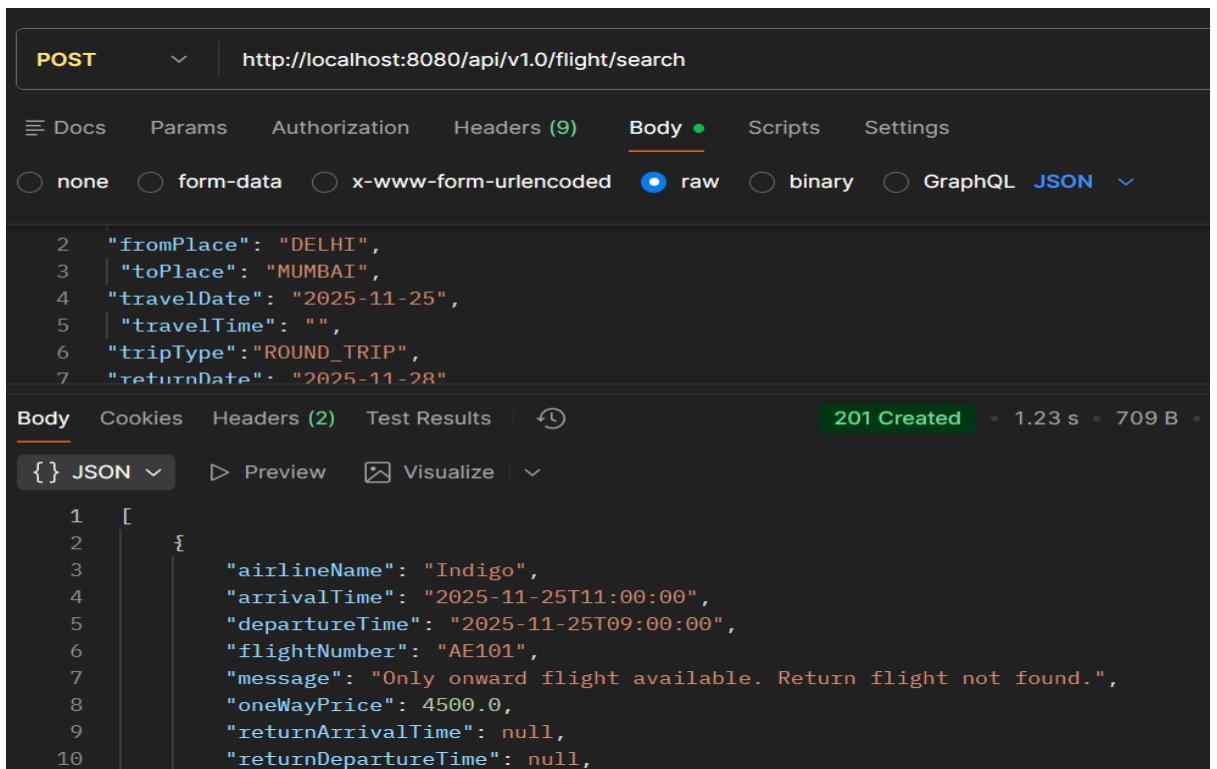
1) Post Request (add flight through inventory)

<http://localhost:8080/api/v1.0/flight/airline/inventory/add>



2) Post request (to search for flights)

<http://localhost:8080/api/v1.0/flight/search>



3) POST request (make bookings)

<http://localhost:8080/api/v1.0/flight/booking/692333797aa12acb5c314279>

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/api/v1.0/flight/booking/692333797aa12acb5c314279`. The request body is a JSON object with the following structure:

```
1 {
2   "email": "amankumar@gmail",
3   "numberOfSeats": 2,
4   "passengers": [
5     {
6       "name": "aman",
```

The response is a 201 Created status with a response time of 2.65 s. The response body is a JSON object:

```
1 {
2   "bookingTime": "2025-11-24T09:23:33.904923700Z",
3   "email": "amankumar@gmail",
4   "message": "Booking successful",
5   "numberOfSeats": 2,
6   "passengers": [
7     {
8       "age": 31,
9       "fareApplied": 3500.0,
10      "fareCategory": "STUDENT",
```

4) GET request(ticket by PNR)

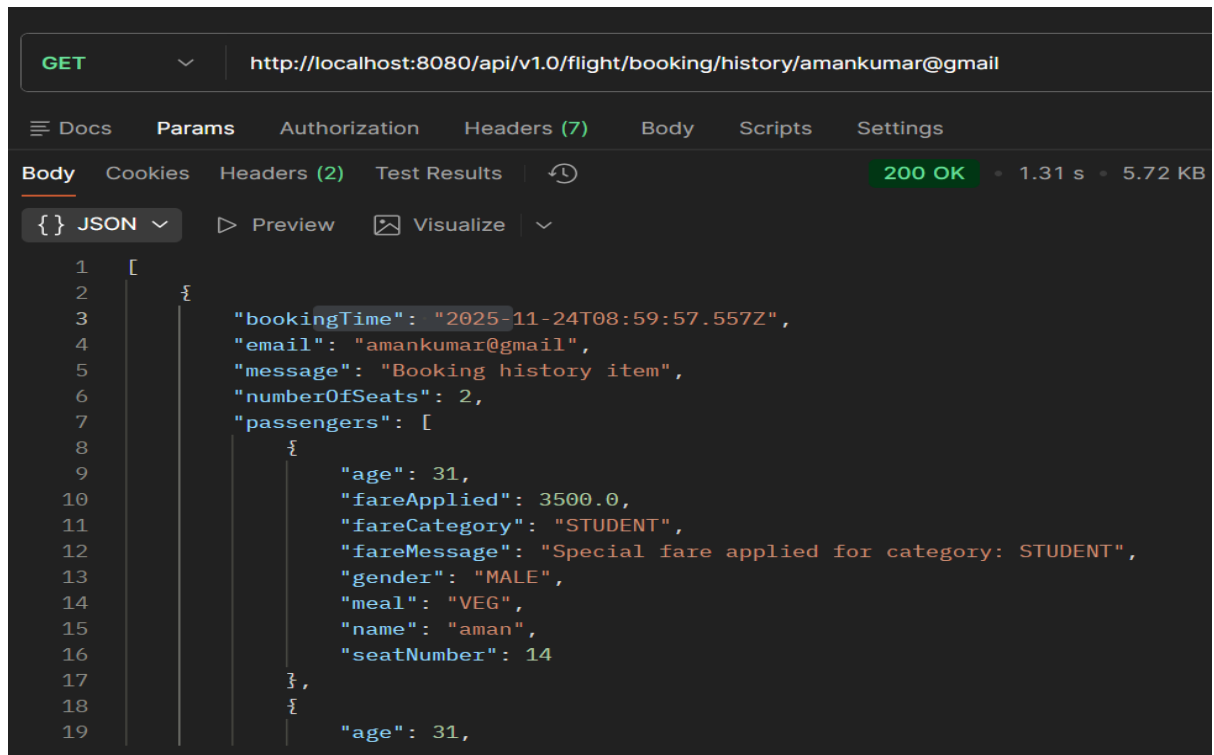
<http://localhost:8080/api/v1.0/flight/ticket/L7V0HA>

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/v1.0/flight/ticket/L7V0HA`. The response is a 200 OK status with a response time of 573 ms. The response body is a JSON object:

```
1 {
2   "bookingTime": "2025-11-24T08:59:57.524Z",
3   "email": "amankumar@gmail",
4   "message": "Booking retrieved",
5   "numberOfSeats": 2,
6   "passengers": [
7     {
8       "age": 31,
9       "fareApplied": 3500.0,
10      "fareCategory": "STUDENT",
11      "fareMessage": "Special fare applied for category: STUDENT",
12      "gender": "MALE",
13      "meal": "VEG",
14      "name": "aman",
15      "seatNumber": 14
```

5) *GET request(see history by email)*

<http://localhost:8080/api/v1.0/flight/booking/history/amankumar@gmail>



```
GET http://localhost:8080/api/v1.0/flight/booking/history/amankumar@gmail

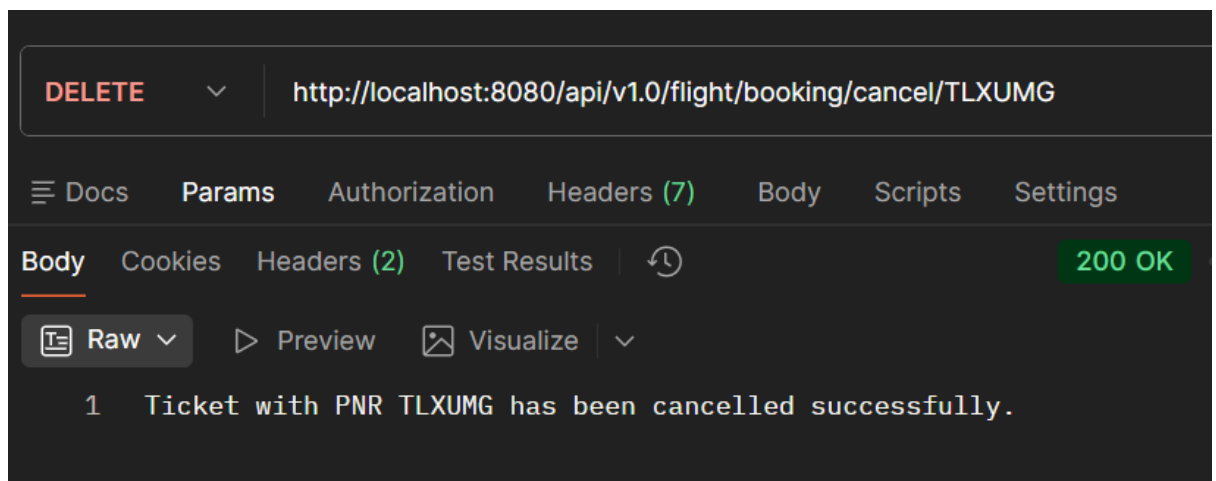
Body Cookies Headers (2) Test Results 200 OK 1.31 s 5.72 KB

{ } JSON Preview Visualize

1  [
2    {
3      "bookingTime": "2025-11-24T08:59:57.557Z",
4      "email": "amankumar@gmail",
5      "message": "Booking history item",
6      "numberOfSeats": 2,
7      "passengers": [
8        {
9          "age": 31,
10         "fareApplied": 3500.0,
11         "fareCategory": "STUDENT",
12         "fareMessage": "Special fare applied for category: STUDENT",
13         "gender": "MALE",
14         "meal": "VEG",
15         "name": "aman",
16         "seatNumber": 14
17       },
18       {
19         "age": 31,
```

6) *DELETE request (delete the bookings through pnr)*

<http://localhost:8080/api/v1.0/flight/booking/cancel/TLXUMG>



```
DELETE http://localhost:8080/api/v1.0/flight/booking/cancel/TLXUMG

Body Cookies Headers (2) Test Results 200 OK

Raw Preview Visualize

1  Ticket with PNR TLXUMG has been cancelled successfully.
```

Jacoco report (code coverage)

flight_webflux_assignment

Sessions

flight_webflux_assignment

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.flightapp.webflux.ServiceImplementation	<div><div></div></div>	90%	<div><div></div></div>	82%	8	52	20	196	1	32	0	2
com.flightapp.webflux.exception	<div><div></div></div>	92%	<div><div></div></div>	50%	10	30	11	93	7	27	0	3
com.flightapp.webflux.model	<div><div></div></div>	92%	<div><div></div></div>	n/a	6	68	7	81	6	68	0	4
com.flightapp.webflux.DTO	<div><div></div></div>	100%	<div><div></div></div>	n/a	0	90	0	132	0	90	0	7
com.flightapp.webflux.Controller	<div><div></div></div>	100%	<div><div></div></div>	n/a	0	10	0	11	0	10	0	4
com.flightapp.webflux	<div><div></div></div>	100%	<div><div></div></div>	n/a	0	2	0	3	0	2	0	1
Total	127 of 1,787	92%	10 of 46	78%	24	252	38	516	14	229	0	21

Created with JaCoCo 0.8.10.202304240956

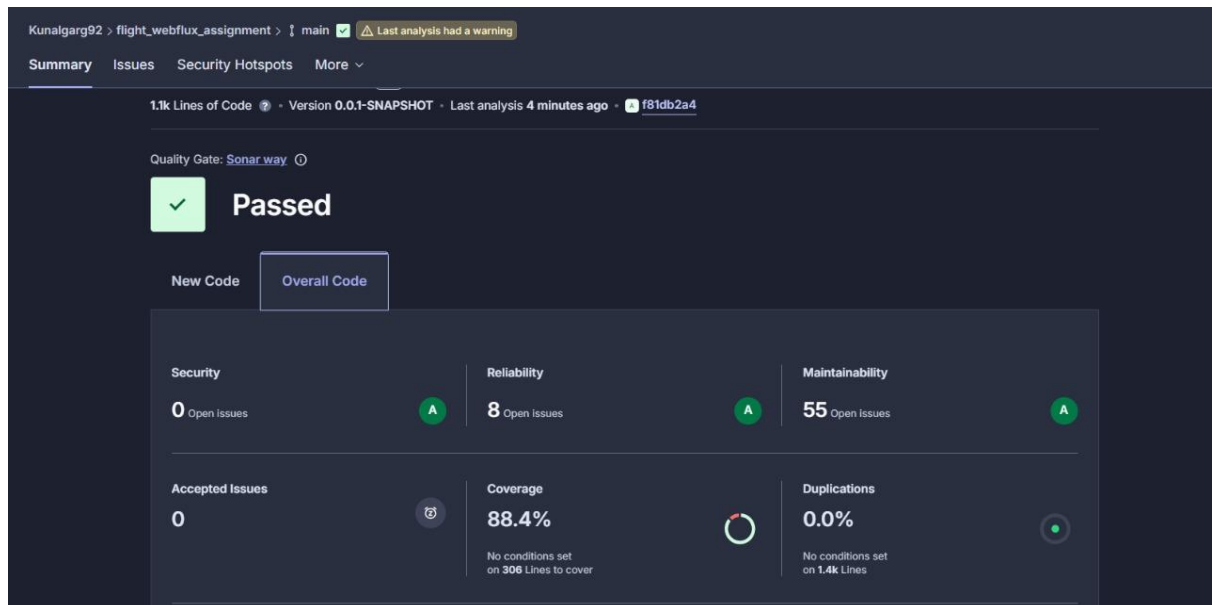
SonarQube attachment (before)

The screenshot shows the SonarQube dashboard for project 'Kunalgarg92 > flightapp_webflux_assignment'. The main status is 'Not computed'. Below this, there are sections for Security (0 Open Issues), Reliability (8 Open Issues), Maintainability (55 Open Issues), Accepted Issues (0), Coverage (8.0%), and Duplications (8.0%). A message indicates that a few extra steps are needed for SonarQube Cloud to analyze code coverage, with a link to 'Set up coverage analysis'. Security Hotspots are also shown as 3.

After running in cmd using {sonartoken}

The screenshot shows the SonarQube dashboard after running the analysis. The main status is 'New Code' and 'Overall Code'. The dashboard now shows updated metrics: Security (0 Open Issues), Reliability (8 Open Issues), Maintainability (57 Open Issues), Accepted Issues (0), Coverage (91.5%), and Duplications (8.0%). A message indicates that no conditions are set on 516 Lines to cover. Security Hotspots are also shown as 3.

After (removed all duplications using Lombok and code coverage)



Jmeter Testing (for 20,50,100)

1)for 20:

The JMeter Summary Report shows the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
GET Requ...	20	22	10	89	19.39	0.00%	2.1/sec	0.20	0.34	96.0
TOTAL	20	22	10	89	19.39	0.00%	2.1/sec	0.20	0.34	96.0

2) for 50 :

The screenshot shows the JMeter Summary Report window. The left sidebar displays a test plan with a 'flight_assignment' test element containing a 'GET Request' and a 'Summary Report' (highlighted in blue). The main panel shows the 'Summary Report' details. The 'Name' field is 'Summary Report'. The 'Comments' field is empty. Below the 'Write results to file / Read from file' section, the 'Filename' field is empty, and the 'Log/Display Only' section has 'Errors' and 'Successes' checkboxes, both of which are unchecked. The 'Configure' button is visible. The table below shows the test results for 50 samples.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
GET Requ...	50	29	10	151	28.04	0.00%	5.1/sec	0.48	0.81	96.0
TOTAL	50	29	10	151	28.04	0.00%	5.1/sec	0.48	0.81	96.0

3) For 100:

The screenshot shows the JMeter Summary Report window for 100 samples. The left sidebar is identical to the previous screenshot, with the 'Summary Report' element highlighted. The main panel shows the 'Summary Report' details. The 'Name' field is 'Summary Report'. The 'Comments' field is empty. Below the 'Write results to file / Read from file' section, the 'Filename' field is empty, and the 'Log/Display Only' section has 'Errors' and 'Successes' checkboxes, both of which are unchecked. The 'Configure' button is visible. The table below shows the test results for 100 samples.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
GET Requ...	100	17	7	109	17.66	0.00%	10.1/sec	0.95	1.62	96.0
TOTAL	100	17	7	109	17.66	0.00%	10.1/sec	0.95	1.62	96.0