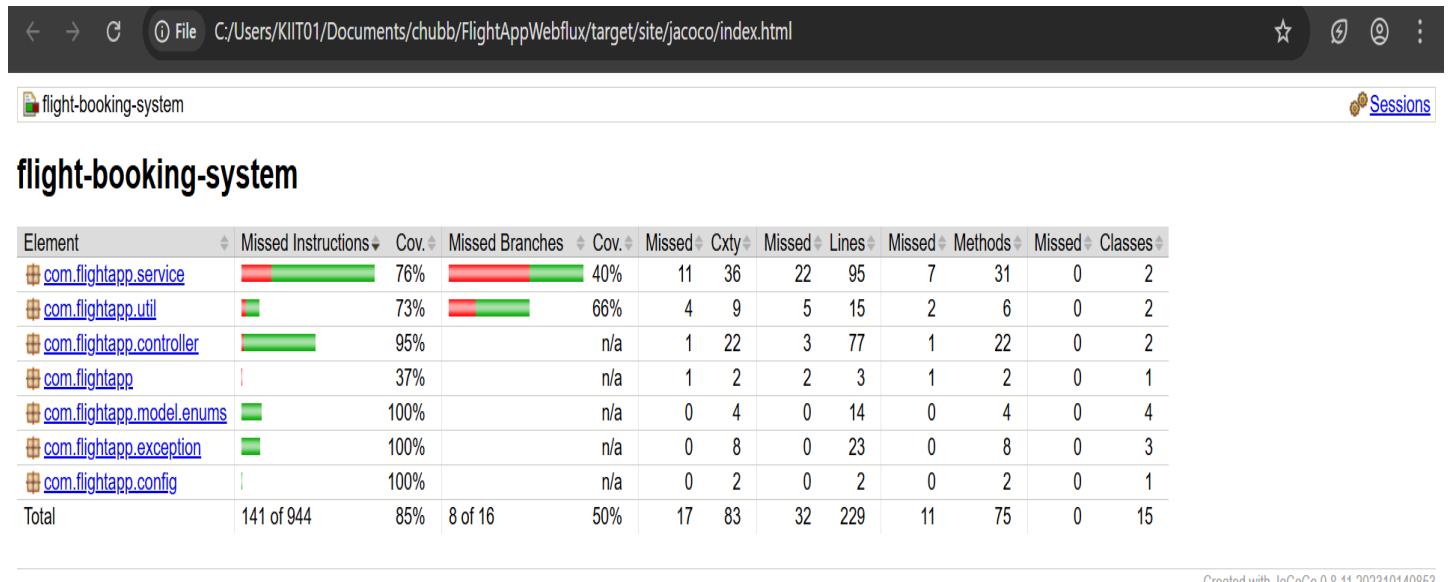


Flight Management System using webflux and mongodb:

Jacoco Report:

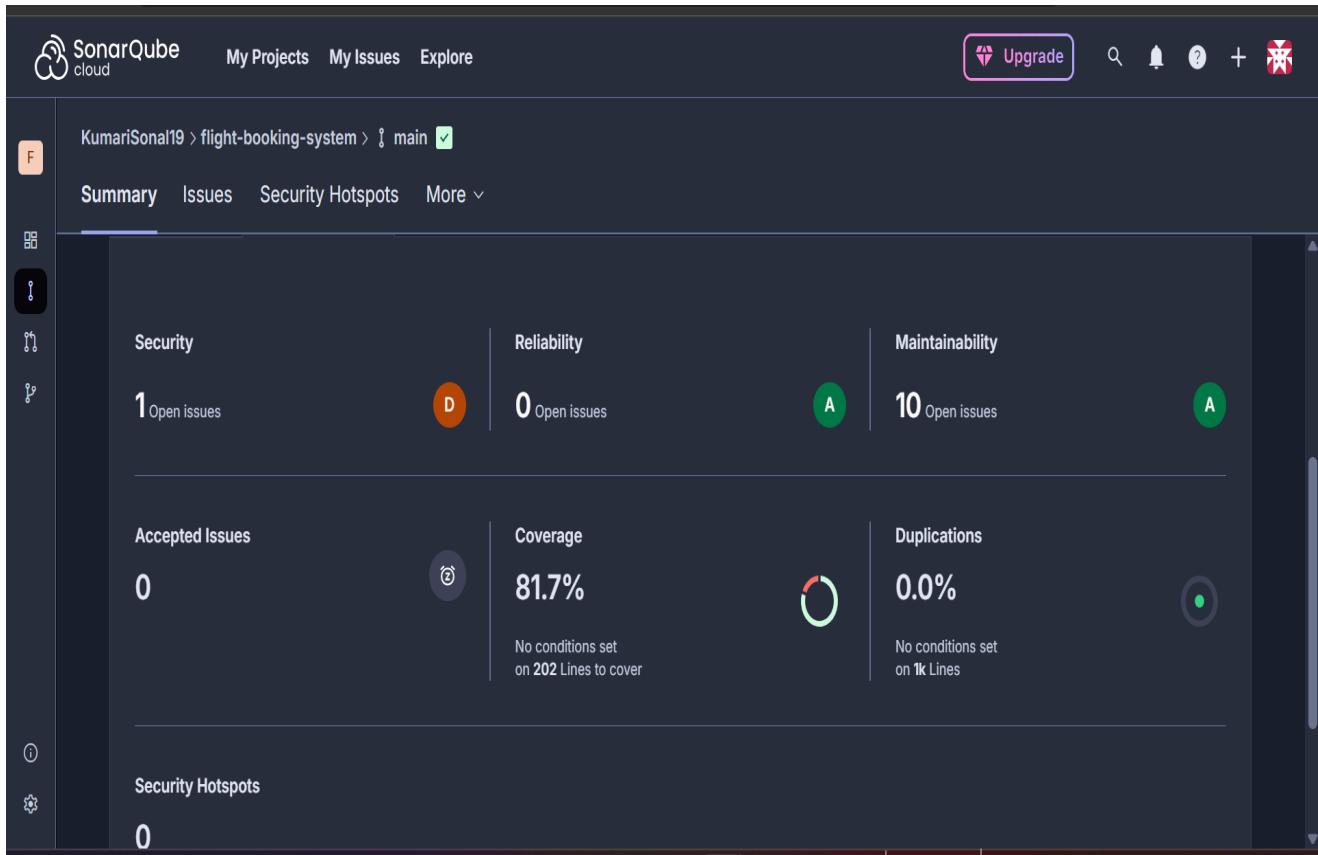
Code Coverage: 85%



SONARQUBE REPORT (Before):

Code Coverage: 81.7

Total Issues: 11



ISSUES:

KumariSonal19 > flight-booking-system > `main`

Summary Issues Security Hotspots More ▾

Filters

✓ Software quality

Security	1
Reliability	0
Maintainability	10

✓ Severity ⓘ

Blocker	0
High	1
Medium	1
Low	2

src/.../java/com/flightapp/controller/FlightController.java

Replace this persistent entity with a simple POJO or DTO object. Consistency

Security High

cwe spring +

Open KumariSonal19 L27 • 10min effort • 8 hours ago • 🔒 Vulnerability • Critical

src/.../java/com/flightapp/exception/GlobalExceptionHandler.java

Remove the parentheses around the "error" parameter Intentionality

Maintainability Low

Open KumariSonal19 L35 • 2min effort • 6 minutes ago • 💀 Code Smell • Minor

src/main/java/com/flightapp/util/ValidationUtil.java

[REDACTED] [REDACTED]

KumariSonal19 > flight-booking-system >  main 

Summary Issues Security Hotspots More ▾

Filters

✓ Software quality

Security	1
Reliability	0
Maintainability	10

✓ Severity ⓘ

Blocker	0
High	1
Medium	1
Low	2

src/.../java/com/flightapp/model/dto/BookingRequestTest.java

Remove this 'public' modifier. Intentionality

Maintainability  Info junit tests +

Open  KumariSonal19 L12 • 2min effort • 1 hour ago • ⚡ Code Smell • ⓘ Info

src/.../java/com/flightapp/model/dto/BookingResponseTest.java

Remove this 'public' modifier. Intentionality

Maintainability  Info junit tests +

Open  KumariSonal19 L14 • 2min effort • 1 hour ago • ⚡ Code Smell • ⓘ Info

src/.../java/com/flightapp/model/dto/FlightSearchRequestTest.java

Remove this 'public' modifier. Intentionality

KumariSonal19 > flight-booking-system >  main 

Summary Issues Security Hotspots More ▾

Filters

✓ Software quality

Security	1
Reliability	0
Maintainability	10

✓ Severity ⓘ

Blocker	0
High	1
Medium	1
Low	2

src/.../java/com/flightapp/model/dto/FlightSearchRequestTest.java

Remove this 'public' modifier. Intentionality

Maintainability  Info junit tests +

Open  KumariSonal19 L10 • 2min effort • 1 hour ago • ⚡ Code Smell • ⓘ Info

src/.../com/flightapp/model/entity/BookingEntityTest.java

Remove this 'public' modifier. Intentionality

Maintainability  Info junit tests +

Open  KumariSonal19 L13 • 2min effort • 1 hour ago • ⚡ Code Smell • ⓘ Info

src/.../com/flightapp/model/entity/FlightEntityTest.java

Remove this 'public' modifier. Intentionality

KumariSonal19 > flight-booking-system > main ✓

Summary Issues Security Hotspots More ▾

Filters

Software quality

- Security 1
- Reliability 0
- Maintainability 10

Severity ?

- Blocker 0
- High 1
- Medium 1
- Low 2

src/.../java/com/flightapp/service/BookingServiceImplTest.java

Open KumariSonal19 L9 • 2min effort • 1 hour ago • Code Smell • Info

Remove this unnecessary import: same package classes are always implicitly imported. Intentionality

Maintainability Low unused +

Open KumariSonal19 L13 • 1min effort • 6 minutes ago • Code Smell • Minor

11 of 11 shown

© 2018-2025 SonarSource Terms Pricing Privacy Cookie Security Community Documentation Contact Status About us
Sàrl. All rights reserved.

SONARQUBE REPORT (After):

All issues resolved

The screenshot shows the SonarQube Issues report interface. At the top, there are tabs: Summary, Issues (which is selected), Security Hotspots, and More. Below the tabs is a sidebar titled "Filters". The "Software quality" section is expanded, showing counts for Security (0), Reliability (0), and Maintainability (0). The "Severity" section is also expanded, showing counts for Blocker (0), High (0), Medium (0), and Low (0). To the right of the filters, there is a "Select issues for bulk actions" panel with a "Change" button, a "Select issues" dropdown, and navigation buttons for issues. It also displays a counter for 0 issues and 0 effort. A message "No Issues. Hooray!" is centered in the main content area. At the bottom, there is a footer with copyright information: "© 2018-2025 SonarSource Sàrl. All rights reserved." followed by links to Terms, Pricing, Privacy, Cookie, Security, Community, Documentation, Contact, Status, and About us.

Summary Issues Security Hotspots More ▾

Filters

Software quality

- Security 0
- Reliability 0
- Maintainability 0

Severity ?

- Blocker 0
- High 0
- Medium 0
- Low 0

Select issues for bulk actions

Change Select issues ▾ ▲ Navigate to issue ⏪ ⏩

0 issues 0 effort

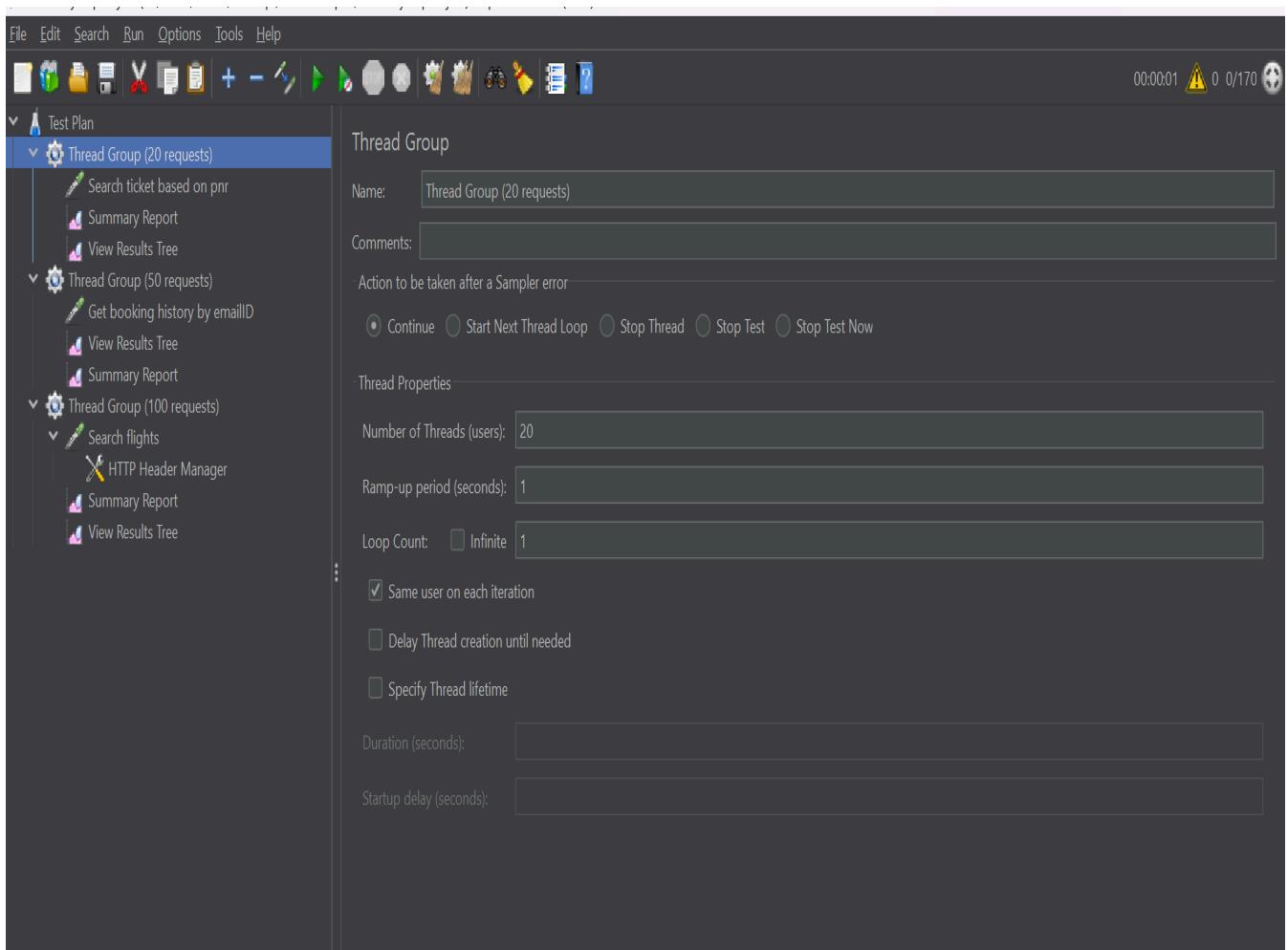
No Issues. Hooray!

© 2018-2025
SonarSource Sàrl.
All rights reserved.

Terms Pricing Privacy Cookie Security Community Documentation Contact Status About us

JMETER TESTING

Case 1: 20 requests



Test Plan

Thread Group (20 requests)

- Search ticket based on pnr
- Summary Report
- View Results Tree

Thread Group (50 requests)

- Get booking history by emailID
- View Results Tree
- Summary Report

Thread Group (100 requests)

- Search flights
- HTTP Header Manager
- Summary Report
- View Results Tree

HTTP Request

Name: Search ticket based on pnr

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: localhost Port Number: 8081

HTTP Request

Method: GET Path: api/flight/ticket/FLT25112588EB65 Content encoding:

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail Add Add from Clipboard Delete Up Down

Summary Report.jmx (C:\Users\KIIIT01\Desktop\Jmeter Scripts\Summary Report.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 ⚠ 0 0/170

Test Plan

- Thread Group (20 requests)
 - Search ticket based on pnr
 - Summary Report**
 - View Results Tree
- Thread Group (50 requests)
 - Get booking history by emaillID
 - View Results Tree
 - Summary Report
- Thread Group (100 requests)
- Search flights
 - HTTP Header Manager
 - Summary Report
 - View Results Tree

Summary Report

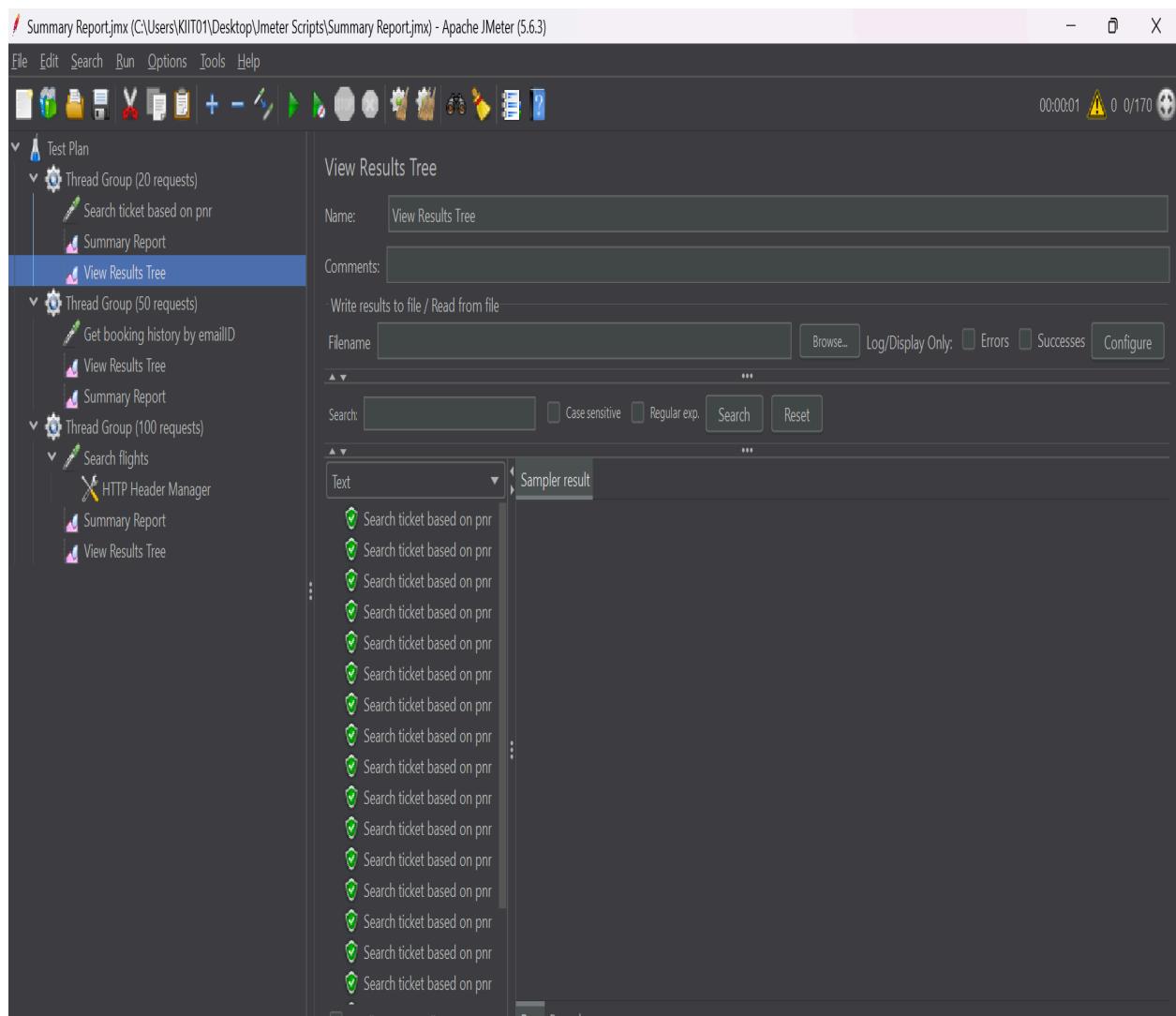
Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
Search ticket...	20	7	5	12	1.95	0.00%	20.9/sec	14.24	3.04	697.0
TOTAL	20	7	5	12	1.95	0.00%	20.9/sec	14.24	3.04	697.0



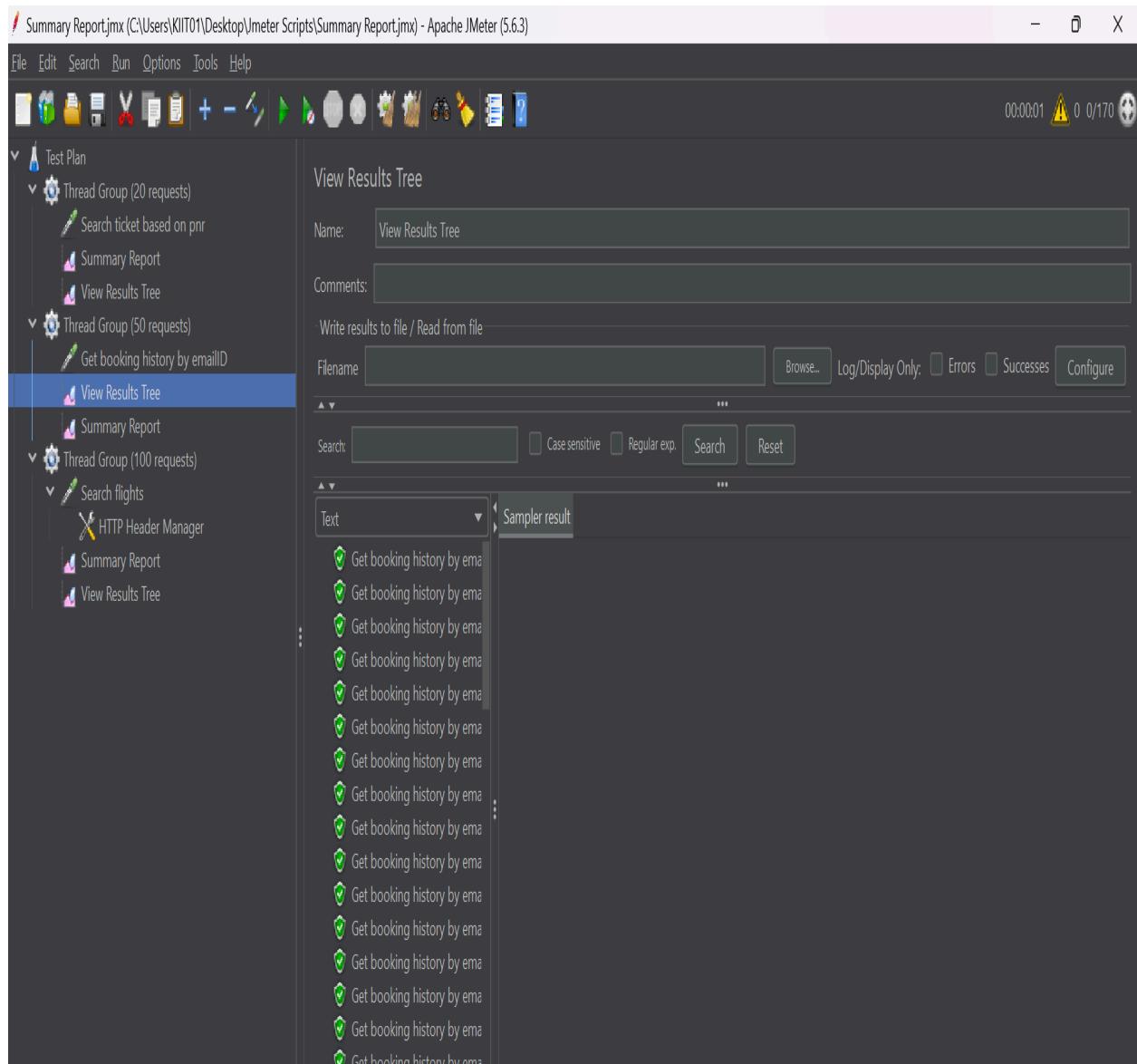
Case 2: 50 requests

The screenshot shows the Apache JMeter 5.6.3 interface with the following configuration:

- Test Plan Structure:**
 - Thread Group (20 requests):
 - Search ticket based on pnr
 - Summary Report
 - View Results Tree
 - Thread Group (50 requests):
 - Get booking history by emailID
 - View Results Tree
 - Summary Report
 - Thread Group (100 requests):
 - Search flights
 - HTTP Header Manager
 - Summary Report
 - View Results Tree

- HTTP Request Configuration:**
- Basic:** Protocol [http]: [] Server Name or IP: localhost Port Number: 8081
- Advanced:** Method: GET Path: /api/flight/booking/history/simran.sharma@example.com Content encoding: []
 - Redirect Automatically
 - Follow Redirects
 - Use KeepAlive
 - Use multipart/form-data
 - Browser-compatible headers
- Parameters:** Send Parameters With the Request:

Name:	Value:	URL Encode?	Content-Type:	Include Equals?
-------	--------	-------------	---------------	-----------------



Summary Report.jmx (C:\Users\KITT01\Desktop\Jmeter Scripts\Summary Report.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0/0/170

Test Plan

- Thread Group (20 requests)
 - Search ticket based on pnr
 - Summary Report
 - View Results Tree
- Thread Group (50 requests)
 - Get booking history by emailID
 - View Results Tree
- Summary Report**
- Thread Group (100 requests)
 - Search flights
 - HTTP Header Manager
 - Summary Report
 - View Results Tree

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Get booking...	50	6	4	13	1.71	0.00%	50.1/sec	30.06	8.21	615.0
TOTAL	50	6	4	13	1.71	0.00%	50.1/sec	30.06	8.21	615.0

Include group name in label? Save Table Header

Case 3: 100 requests

Summary Report.jmx (C:\Users\KIT01\Desktop\Jmeter Scripts\Summary Report.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0 0/170

Test Plan

Thread Group (20 requests)

- Search ticket based on pnr
- Summary Report
- View Results Tree

Thread Group (50 requests)

- Get booking history by emailID
- View Results Tree
- Summary Report

Thread Group (100 requests) **(Selected)**

- Search flights
- HTTP Header Manager
- Summary Report
- View Results Tree

Thread Group

Name: Thread Group (100 requests)

Comments:

Action to be taken after a Sampler error

Continue Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 100

Ramp-up period (seconds): 1

Loop Count: Infinite 1

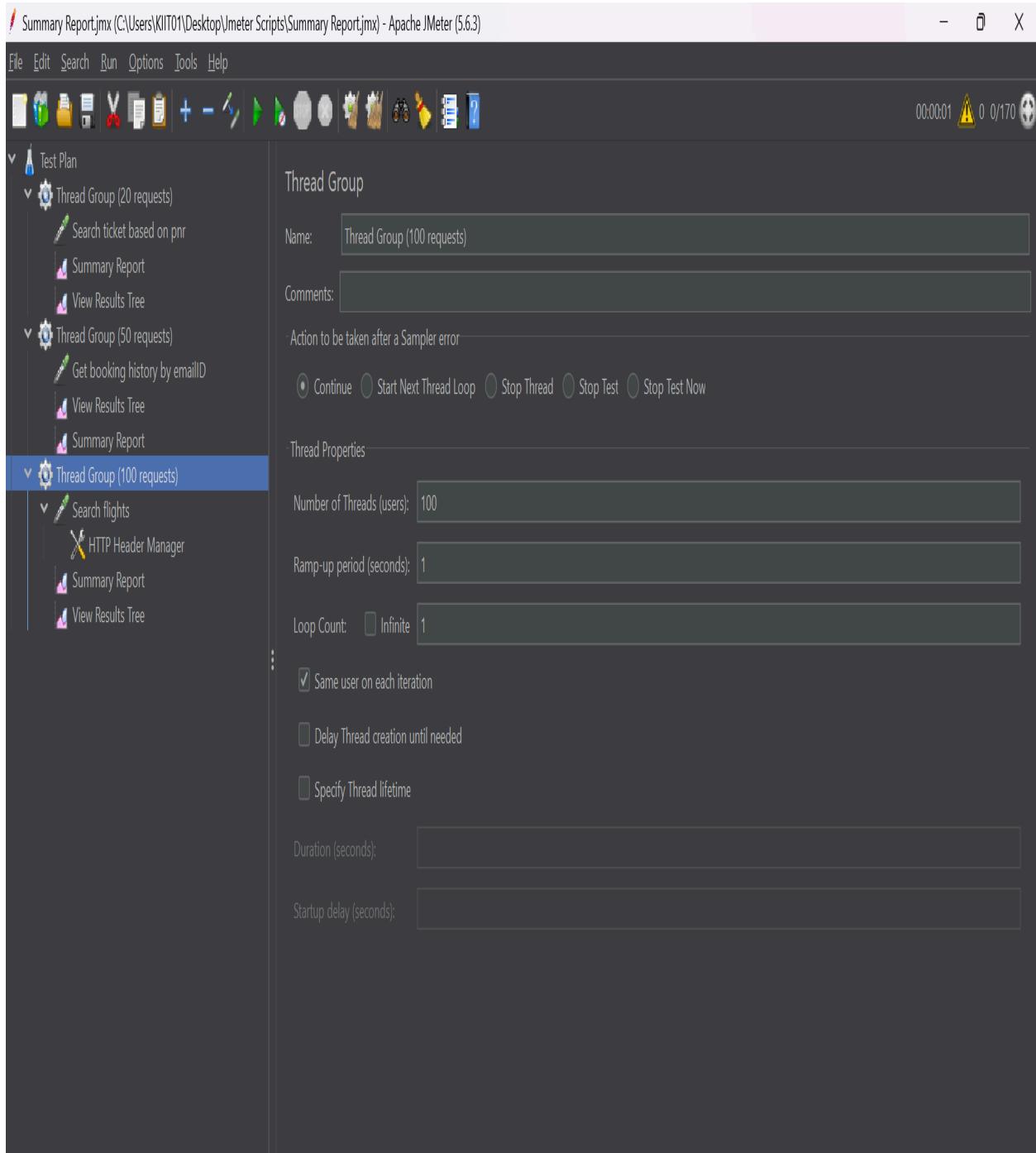
Same user on each iteration

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):



Summary Report.jmx (C:\Users\KIT01\Desktop\Jmeter Scripts\Summary Report.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 ⚠ 0 0/170 ⏪

Test Plan

- Thread Group (20 requests)
 - Search ticket based on pnr
 - Summary Report
 - View Results Tree
- Thread Group (50 requests)
 - Get booking history by emailID
 - View Results Tree
 - Summary Report
- Thread Group (100 requests)
 - Search flights
 - HTTP Header Manager
 - Summary Report
 - View Results Tree

HTTP Request

Name: Search flights

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: localhost Port Number: 8081

HTTP Request

Method: POST Path: /api/flight/search Content encoding:

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters Body Data Files Upload

```
1 {
2   "fromLocation": "Delhi",
3   "toLocation": "Mumbai",
4   "travelDate": "2025-11-26",
5   "tripType": "ONE WAY"
6 }
```

Summary Report.jmx (C:\Users\KIT01\Desktop\Jmeter Scripts\Summary Report.jmx) - Apache JMeter (5.6.3)

- X

File Edit Search Run Options Tools Help

00:00:01 ⚠ 0/170

Test Plan

Thread Group (20 requests)

- Search ticket based on prn
- Summary Report
- View Results Tree

Thread Group (50 requests)

- Get booking history by emailID
- View Results Tree
- Summary Report

Thread Group (100 requests)

- Search flights
- HTTP Header Manager

Summary Report

View Results Tree

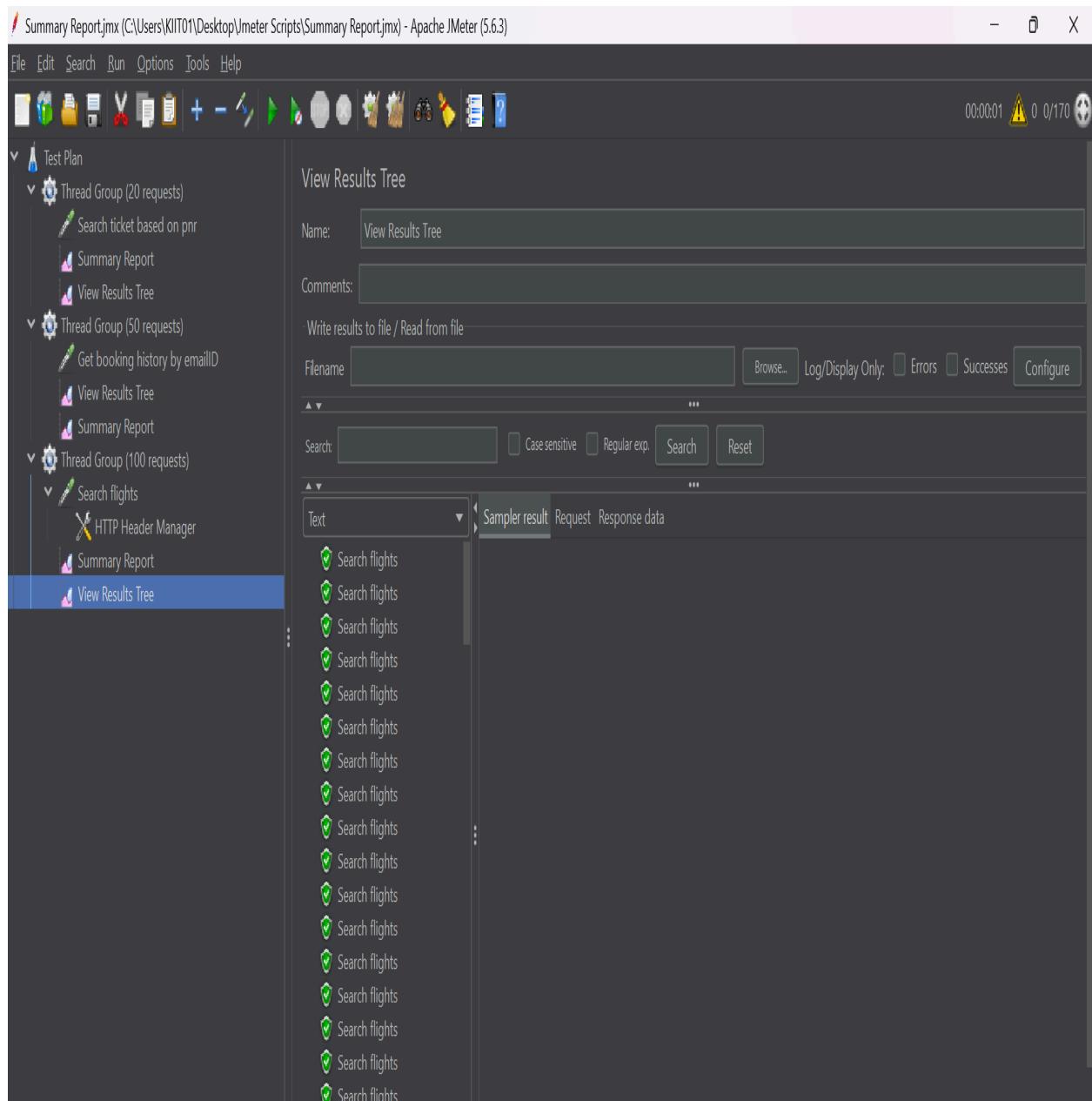
Summary Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
Search flights	100	9	7	18	1.69	0.00%	100.9/sec	69.67	29.76	707.0
TOTAL	100	9	7	18	1.69	0.00%	100.9/sec	69.67	29.76	707.0



POSTMAN API ENDPOINT TESTING

1. POST:

<http://localhost:8081/api/flight/airline/inventory>

The screenshot shows the Postman application interface. At the top, it displays the collection name "My Collection / FlightAppWebflux / Add flights to inventory". Below this, the request details are shown: a POST method, the URL "http://localhost:8081/api/flight/airline/inventory", and a "Send" button. The "Body" tab is selected, showing a raw JSON payload with 12 numbered lines. The payload is as follows:

```
1 {  
2   "airlineName": "SpiceJet",  
3   "airlineLogo": "https://example.com/spicejet.png",  
4   "fromLocation": "Bangalore",  
5   "toLocation": "Chennai",  
6   "departureTime": "2025-11-26T08:00:00",  
7   "arrivalTime": "2025-11-26T09:00:00",  
8   "price": 3000.0,  
9   "availableSeats": 150,  
10  "totalSeats": 150,  
11  "flightNumber": "SG302"  
12 }
```

Below the body, the response details are shown: a green "201 Created" status box, a response time of "721 ms", a response size of "157 B", and a "Save Response" button. The response body is displayed in JSON format with 4 numbered lines:

```
1 {  
2   "message": "Flight inventory added successfully",  
3   "id": "6924ed46234fb520a88ccdc4"  
4 }
```

2. POST:

<http://localhost:8081/api/flight/search>

The screenshot shows the Postman application interface. On the left, the sidebar displays 'SONAL KUMARI_3114's Workspace' with various collections like 'FlightApp', 'SpringBootWebfluxExample', 'MongoSpring', 'SpringMongodbReactive', and 'FlightAppWebflux'. Under 'FlightAppWebflux', several API endpoints are listed: 'POST Add flights to inventory', 'POST Search flights' (which is selected), 'POST Book tickets', 'GET Get ticket by PNR', 'GET Get Booking history by email', and 'DEL Cancel Ticket'. The main workspace shows a POST request to 'http://localhost:8081/api/flight/search'. The 'Body' tab is selected, showing the following JSON payload:

```
1 {
2   "fromLocation": "Delhi",
3   "toLocation": "Mumbai",
4   "travelDate": "2025-11-26",
5   "tripType": "ONE WAY"
6 }
```

SONAL KUMARI_3114's Workspace

Search Postman

Invite Upgrade

SONAL KUMARI_3114's Workspace

New Import

DEL Delete Flight POST Search POST Book GET Get ticket GET Get Booking history by email DEL Cancel Ticket

HTTP My Collection / FlightAppWebflux / Search flights

POST http://localhost:8081/api/flight/search

Docs Params Authorization Headers (8) Body Scripts Settings Cookies

Body Cookies Headers (2) Test Results

200 OK 131 ms 683 B Save Response

{} JSON Preview Visualize

```
1 [ { "id": "692411b9702bc266c6879fde", "airlineName": "Air India", "airlineLogo": "https://example.com/airindia.png", "fromLocation": "Delhi", "toLocation": "Mumbai", "departureTime": "2025-11-26T10:00:00", "arrivalTime": "2025-11-26T12:30:00", "price": 5500.0, "totalSeats": 180, "availableSeats": 180, "flightNumber": "AI101" }, { "id": "6924137c702bc266c6879fdf", "airlineName": "IndiGo", "airlineLogo": "https://example.com/indigo.png", "fromLocation": "Delhi", "toLocation": "Mumbai", "departureTime": "2025-11-26T10:00:00", "arrivalTime": "2025-11-26T12:30:00", "price": 5500.0, "totalSeats": 180, "availableSeats": 180, "flightNumber": "IG101" } ]
```

3. POST:

<http://localhost:8081/api/flight/booking/692413c1702bc266c6879fe0>

The screenshot shows the Postman interface with the following details:

- Collection:** My Collection / FlightAppWebflux
- Request Type:** POST
- URL:** http://localhost:8081/api/flight/booking/692413c1702bc266c6879fe0
- Body (raw JSON):**

```
1 {
2   "userName": "Simran Sharma",
3   "email": "simran.sharma@example.com",
4   "passengers": [
5     {
6       "name": "Simran Sharma",
7       "gender": "MALE",
8       "age": 28,
9       "seatNumber": "A1",
10      "mealPreference": "VEG"
11    },
12    {
13      "name": "Priya Sharma",
14      "gender": "FEMALE",
15      "age": 26,
16      "seatNumber": "A2",
17      "mealPreference": "VEG"
18    }
19 ]
```

The screenshot shows the Postman interface with the following details:

- Collection:** My Collection / FlightAppWebflux
- Request Type:** POST
- URL:** http://localhost:8081/api/flight/booking/692413c1702bc266c6879fe0
- Body (raw JSON):**
- Headers:** Headers (8) - Content-Type: application/json
- Response Status:** 201 Created
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Ticket booked successfully",
4   "data": {
5     "pnr": "FLT25112588EB65",
6     "flightId": "692413c1702bc266c6879fe0",
7     "userName": "Simran Sharma",
8     "email": "simran.sharma@example.com",
9     "passengers": [
10       {
11         "name": "Simran Sharma",
12         "gender": "MALE",
13         "age": 28,
14         "seatNumber": "A1",
15         "mealPreference": "VEG"
16       },
17       {
18         "name": "Priya Sharma",
19         "gender": "FEMALE",
20       }
21     ]
22   }
23 }
```

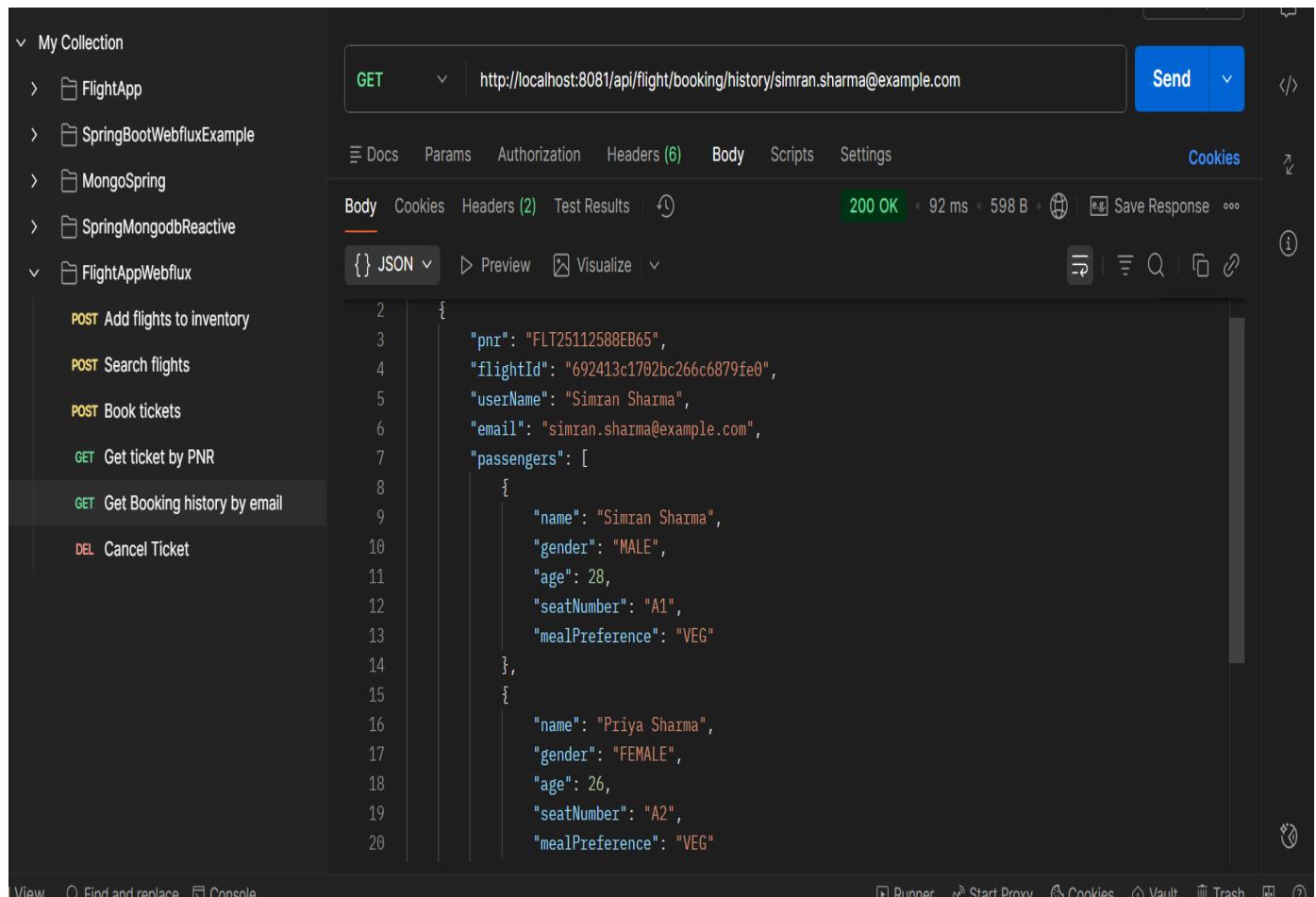
4. GET: <http://localhost:8080/api/flight/booking/ticket/FLT25112588EB65>

The screenshot shows the Postman application interface. On the left, there's a sidebar with a collection named "My Collection" containing several projects: FlightApp, SpringBootWebfluxExample, MongoSpring, SpringMongodbReactive, and FlightAppWebflux. Under FlightAppWebflux, there are four methods listed: POST Add flights to inventory, POST Search flights, POST Book tickets, and GET Get ticket by PNR. The "GET Get ticket by PNR" method is currently selected. The main panel shows an HTTP request configuration: Method GET, URL http://localhost:8080/api/flight/booking/ticket/FLT25112588EB65, Headers (6), Body (JSON), and Settings. Below this, the response details are shown: Status 200 OK, Time 363 ms, Size 697 B, and a green "Save Response" button. The response body is displayed as JSON, showing a ticket record with a passenger list:

```
5   "pnr": "FLT25112588EB65",
6   "flightId": "692413c1702bc266c6879fe0",
7   "userName": "Simran Sharma",
8   "email": "simran.sharma@example.com",
9   "passengers": [
10      {
11         "name": "Simran Sharma",
12         "gender": "MALE",
13         "age": 28,
14         "seatNumber": "A1",
15         "mealPreference": "VEG"
16      },
17      {
18         "name": "Priya Sharma",
19         "gender": "FEMALE",
20         "age": 26,
21         "seatNumber": "A2",
22         "mealPreference": "VEG"
23     }
```

5. GET:

<http://localhost:8080/api/flight/booking/history/>



The screenshot shows the Postman interface with the following details:

- Collection:** My Collection
- Request Method:** GET
- Request URL:** <http://localhost:8081/api/flight/booking/history/simran.sharma@example.com>
- Response Status:** 200 OK
- Response Time:** 92 ms
- Response Size:** 598 B
- Headers:** Authorization, Content-Type, Date, User-Agent
- Body (JSON):**

```
2   {
3     "pnr": "FLT25112588EB65",
4     "flightId": "692413c1702bc266c6879fe0",
5     "userName": "Simran Sharma",
6     "email": "simran.sharma@example.com",
7     "passengers": [
8       {
9         "name": "Simran Sharma",
10        "gender": "MALE",
11        "age": 28,
12        "seatNumber": "A1",
13        "mealPreference": "VEG"
14      },
15      {
16        "name": "Priya Sharma",
17        "gender": "FEMALE",
18        "age": 26,
19        "seatNumber": "A2",
20        "mealPreference": "VEG"
21     ]
22   }
```

6. DELETE: <http://localhost:8080/api/flight/booking/cancel/FLT25112588EB65>

The screenshot shows the Postman application interface. At the top, a header bar displays "DELETE" and the URL "http://localhost:8080/api/flight/booking/cancel/FLT25112588EB65". To the right of the URL is a blue "Send" button. Below the header, a navigation bar includes "Docs", "Params", "Authorization", "Headers (6)", "Body", "Scripts", "Settings", and "Cookies". A "Query Params" section is visible below the navigation bar. The main content area shows a table with columns "Key", "Value", "Description", and "Bulk Edit". Under "Key", there is a single entry "Key". Under "Value", there is a single entry "Value". Under "Description", there is a single entry "Description". Below the table, tabs for "Body", "Cookies", "Headers (2)", and "Test Results" are present, with "Body" being the active tab. The "Body" tab displays a JSON response with three lines of code: "1 {", "2 | \"success\": true,", and "3 | \"message\": \"Booking cancelled successfully\",". Above the body content, status information is shown: "200 OK" in green, followed by "99 ms", "185 B", and a globe icon. To the right of this status info are "Save Response" and "Copy" buttons. At the bottom of the interface are standard browser-style navigation buttons.