

ASSIGNMENT 4 –IMPORTANT SCREENSHOTS:

SonarQube Cloud Analysis (before):

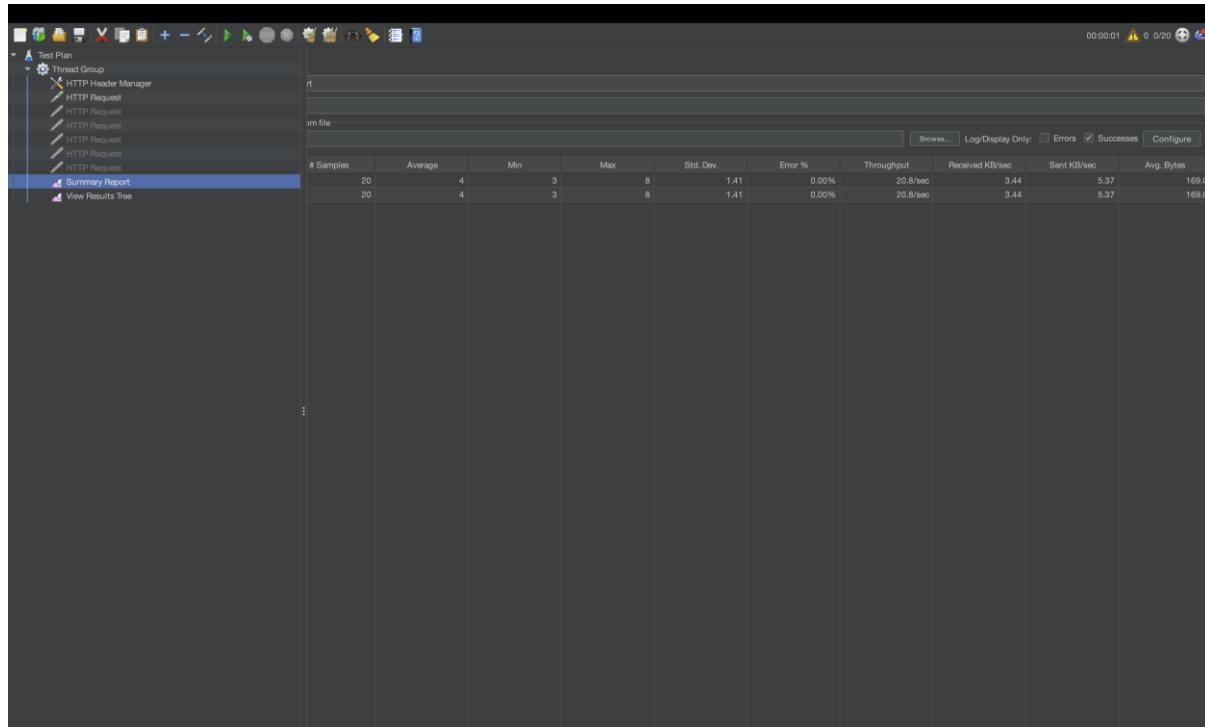
The screenshot shows the SonarQube Cloud interface for the FlightBookingApplication project. The analysis status is marked as 'Failing' with a red exclamation icon. The summary panel displays various metrics: Security (0 Open issues), Reliability (0 Open issues), Maintainability (22 Open issues), Accepted Issues (0), Coverage (20.1%, 351 Lines to cover), Duplications (0.0%, 1.4k Lines), and 1 Security Hotspot. The overall status is labeled 'Failing'.

SonarQube Cloud Analysis (After):

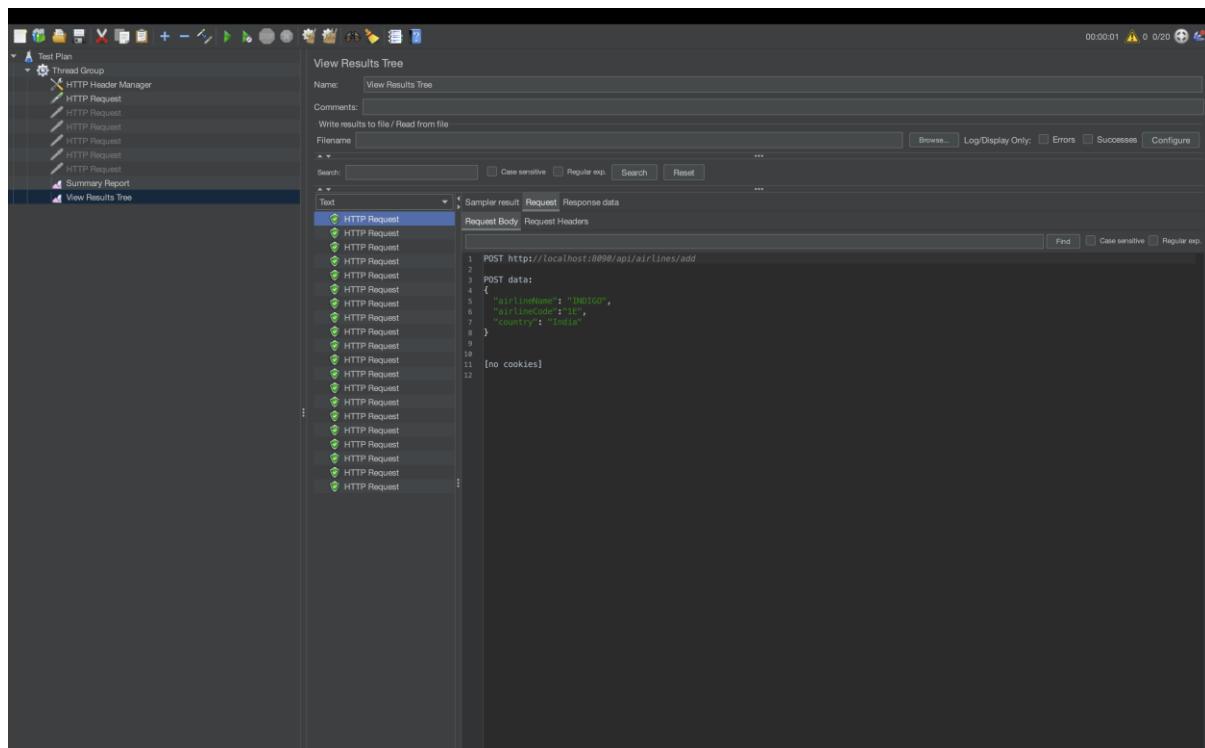
The screenshot shows the SonarQube Cloud interface for the FlightBookingApplication project after a code review. The analysis status is now marked as 'Passed' with a green checkmark icon. The summary panel displays updated metrics: New Code (Since about 8 hours ago), Accepted Issues (0), Coverage (97.42%, 151 Lines to cover), Duplications (0.0%), and 0 Security Hotspots. The overall status is labeled 'Passed'.

JMETER Output for 20 requests:

Summary-report:

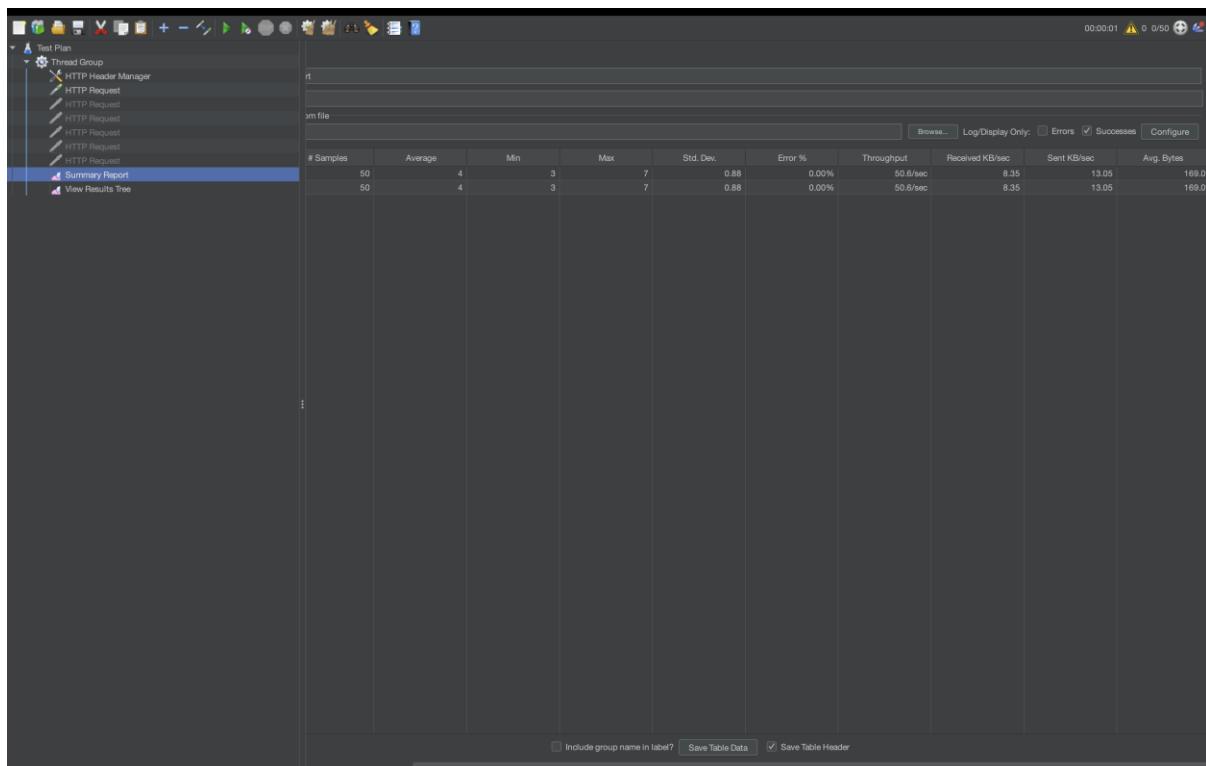


View results tree:

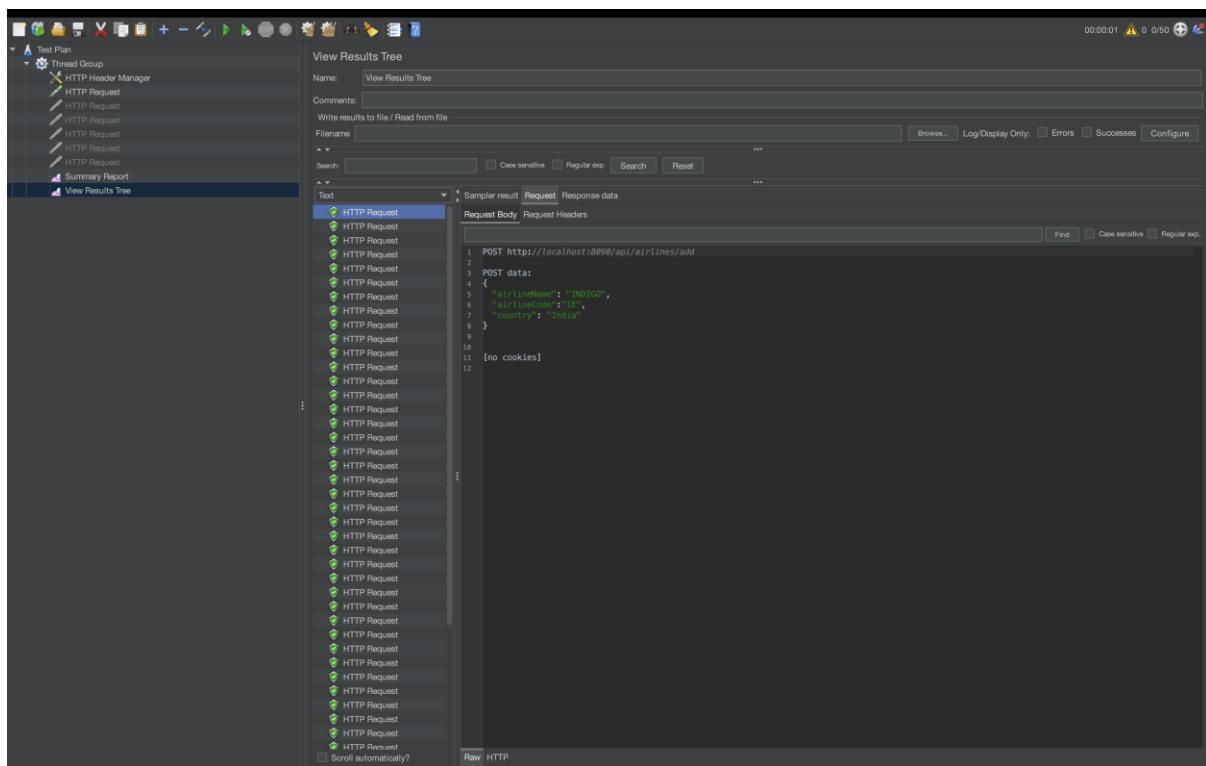


JMETER Output for 50 requests:

Summary-report:

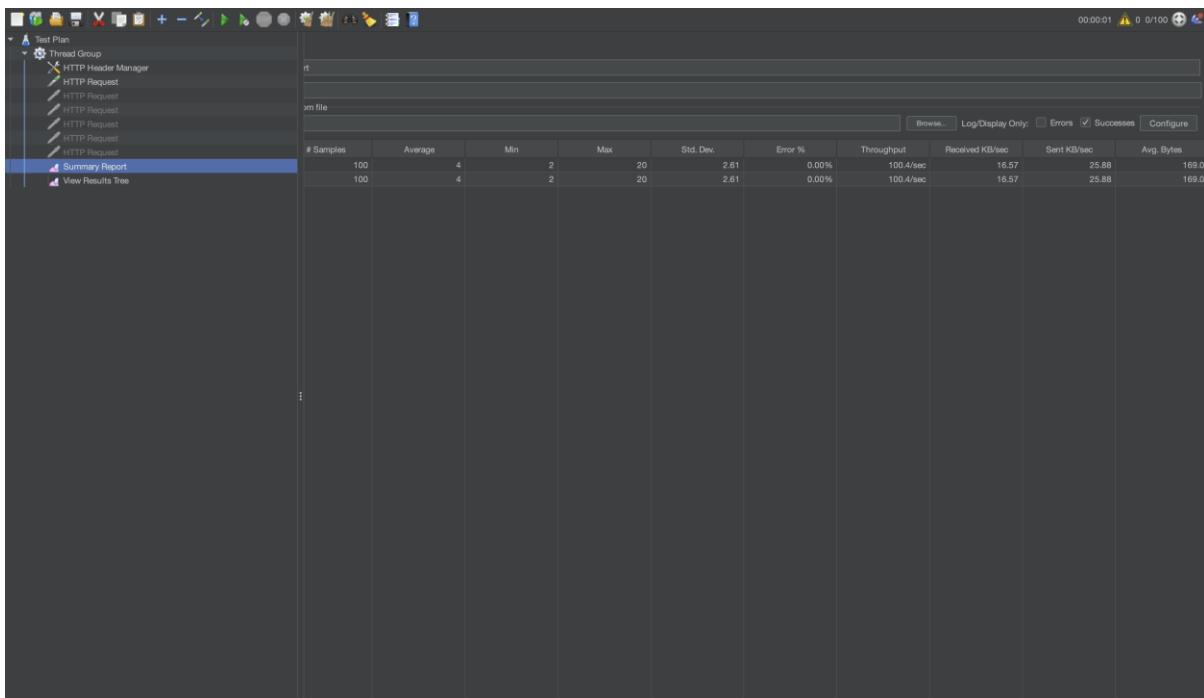


View results tree:

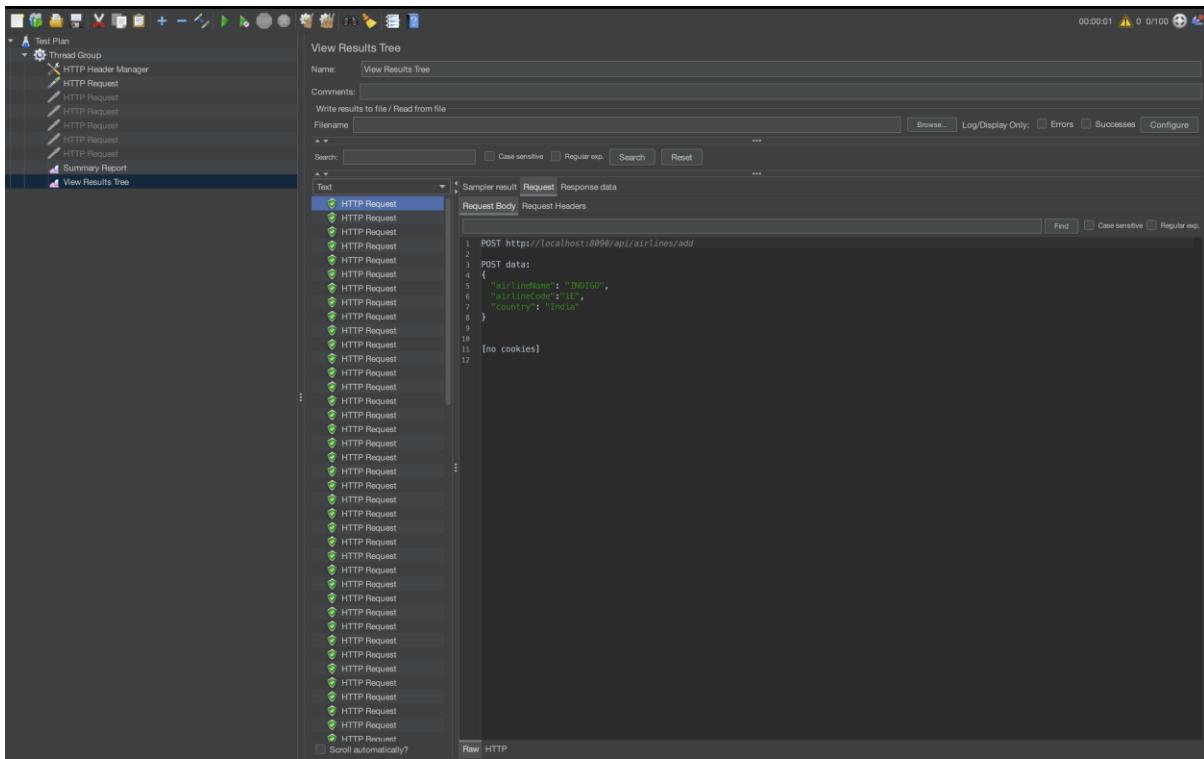


JMETER Output for 100 requests:

Summary-report:



View results tree:



MongoDB Connected :

The screenshot shows the Compass MongoDB interface. At the top, it displays the connection details: localhost:27017, flight_booking, and the current collection: airlines. There are tabs for 'My Queries' and 'Data Modeling'. On the left, the 'CONNECTIONS' sidebar shows a tree structure of databases and collections, including 'flight_booking' which contains 'airlines', 'bookings', 'flights', 'passengers', 'payments', and sub-collections 'local', 'reactive_db', 'test', and 'student'. The main area lists the collections with their properties: Storage size, Documents, Avg. document size, Indexes, and Total index size. The 'airlines' collection has 36.86 kB storage, 2 documents, and 125.00 B average document size.

Working PostMan:

The screenshot shows the Postman application interface. The left sidebar shows 'Joshika sree Polaku's Workspace' with various collections and environments. The main workspace shows a POST request to 'http://localhost:890/api/airlines/add'. The request body is set to 'raw' JSON format with the following content:

```

1 {
2   "airlineName": "INDIGO",
3   "airlineCode": "6E",
4   "country": "India"
5 }
6

```

The response status is '201 Created' with a timestamp of '15 ms' and a size of '169 B'. The response body is also shown in JSON format:

```

1 {
2   "id": "6922f0a9f4239e0557a9301a",
3   "airlineName": "INDIGO",
4   "airlineCode": "6E",
5   "country": "India"
6 }

```

Working SwaggerUI:

The screenshot shows a browser window displaying the SwaggerUI interface at `localhost:8090/webjars/swagger-ui/index.html#/`. The title bar includes the URL and a 'New Chrome available' notification. The main content area is titled 'OpenAPI definition v0 OAS 3.0' and shows an 'Explore' button. Below this, there's a 'Servers' dropdown set to '`http://localhost:8090 - Generated server url`'. The interface is organized into sections for different controllers:

- payment-controller**: Contains three items:
 - `POST /api/payments/pay/{pnrr}` (green)
 - `GET /api/payments/{id}` (blue)
 - `DELETE /api/payments/{id}` (red)
- flight-controller**: Contains four items:
 - `POST /api/flights/search` (green)
 - `POST /api/flights/add` (green)
 - `GET /api/flights/{id}` (blue)
 - `GET /api/flights/stats/seats` (blue)
- booking-controller**: Contains one item:
 - `POST /api/bookings/book/{flightId}` (green)