

CHUBB®

WEEK-9 ASSIGNMENT

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

INDEX

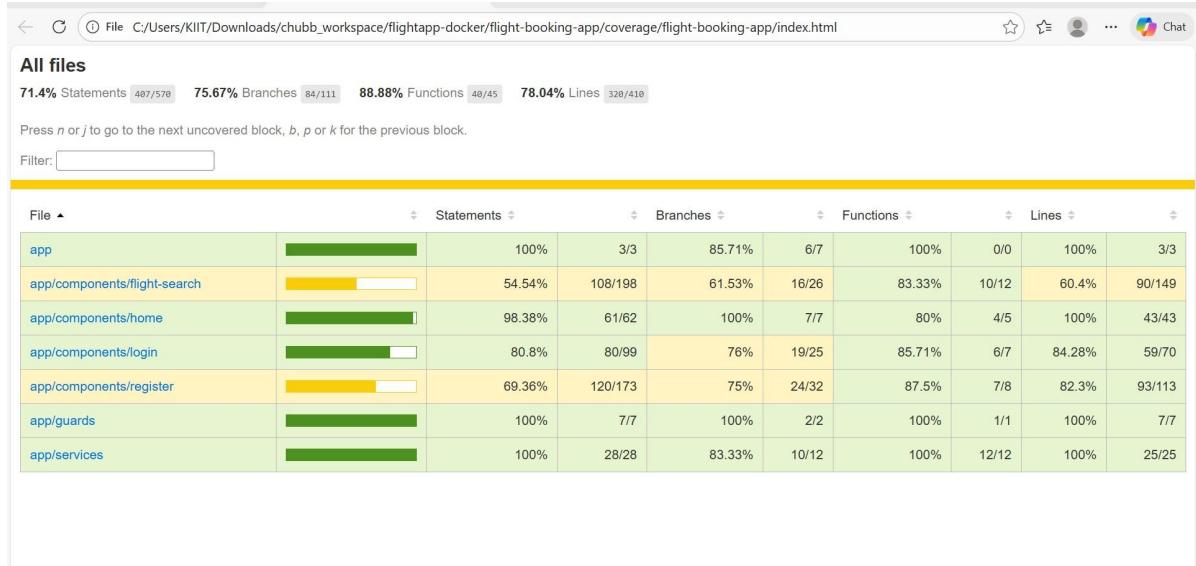
(flight app with angular)

| S. No. | Main Category | Details Included |
|-------------------|--------------------------------|---|
| 1 | Code Coverage Report | <ul style="list-style-type: none">• All Files• Code Coverage Breakdown |
| 2 | Test Run (UI) | <ul style="list-style-type: none">• UI Test Execution Results |
| 3 | Home Page | <ul style="list-style-type: none">• Home Page UI & Functional Testing |
| 4 | Register Page (Create Account) | <ul style="list-style-type: none">• User Registration Flow• Account Creation Validation |
| 5 | Login Page | <ul style="list-style-type: none">• User Login Functionality |
| 6 | Search Flights Page | <ul style="list-style-type: none">• Flight Search UI• Search Functionality |
| 7 | Login for Searching Flights | <ul style="list-style-type: none">• Authentication Required for Flight Search• Search Flights Validation Errors |
| 8 | Validation Errors | <ul style="list-style-type: none">• Login Validation Errors• Email Validation Errors• Registration Page Validation Errors |
| 9 | Build Success | <ul style="list-style-type: none">• Successful Build Confirmation |
| 10 | Clean Flight Search Page | <ul style="list-style-type: none">• No Pre-filled Data• UI Reset Validation |
| 11 | Clean Login Page | <ul style="list-style-type: none">• Empty Fields Validation |
| 12 | Password Mismatch Error | <ul style="list-style-type: none">• Incorrect Password Handling |
| 13 | MongoDB Data | <ul style="list-style-type: none">• Database Records Verification |
| 14 | MySQL Workbench | <ul style="list-style-type: none">• Relational Database Validation |
| 15 | Password Eligibility Cases | <ul style="list-style-type: none">• Length Check• Strength Rules• Format Validation |
| 16 | User Already Exists Case | <ul style="list-style-type: none">• Duplicate User Registration Handling |
| 17 | Postman Reports | <ul style="list-style-type: none">• API Testing Reports• Request–Response Validation |

INDEX

| Content | Page No. |
|---|----------|
| 1. JACOCO Code Coverage Report | 1 |
| 2. SonarQube Report & Issues | 3 |
| 3. System Architecture, ER Diagram | 5 |
| 4. JMeter & RabbitMQ dashboard | 6 |
| — JMeter Result Tree | 6 |
| — JMeter Summary Report | 6 |
| — Apache JMETER Dashboard | 7 |
| 5. Logs | 9 |
| — RabbitMQ Dashboard | 9 |
| — Eureka server, Booking Service | 10 |
| — Flight Service, API Gateway, Notification service | 10 |
| 6. MongoDB Screenshots | 11 |
| 7. Postman Screenshots | 11 |
| — Circuit Breaker, Message broker | 11 |
| — — JWT security through API Gateway | 12 |
| — — Status,Events | 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, get flight by PNR, get flight by ID | 15 |
| — Cancel Booking, FALBACK CASE | 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, MySQL Workbench | 17 |

CODE COVERAGE REPORTS



CODE COVERAGE BREAKDOWN

```
PS C:\Users\KIIT\Downloads\chubb_workspace\flightapp-docker\flight-booking-app> npm run test:coverage
> flight-booking-app@0.0.0 test:coverage
> ng test --coverage

Initial chunk files | Names | Raw size
spec-flight-search.js | spec-flight-search | 66.94 kB
spec-register.js | spec-register | 48.87 kB
spec-login.js | spec-login | 32.60 kB
spec-home.js | spec-home | 23.71 kB
spec-auth.js | spec-auth | 5.17 kB
chunk-FB5WBAN.js | - | 4.66 kB
spec-flight.js | spec-flight | 3.76 kB
spec-auth-guard.js | spec-auth-guard | 2.60 kB
chunk-62UV37PV.js | - | 2.50 kB
spec-app.js | spec-app | 1.78 kB
init-testbed.js | init-testbed | 1.20 kB
chunk-LGBCSKED.js | - | 1.11 kB
styles.css | styles | 367 bytes

| Initial total | 195.26 kB

Application bundle generation complete. [4.382 seconds] - 2025-12-14T19:10:40.944Z
Watch mode enabled. Watching for file changes...
DEV v4.0.15 C:/Users/KIIT/Downloads/chubb_workspace/flightapp-docker/flight-booking-app
```

```

DEV v4.0.15 C:/Users/KIIT/Downloads/chubb_workspace/flightapp-docker/flight-booking-app
Coverage enabled with v8

✓ flight-booking-app src/app/components/home/home.spec.ts (7 tests) 1227ms
  ✓ should create 404ms
✓ flight-booking-app src/app/services/auth.spec.ts (14 tests) 154ms
✓ flight-booking-app src/app/app.spec.ts (1 test) 15ms
✓ flight-booking-app src/app/guards/auth-guard.spec.ts (4 tests) 22ms
✓ flight-booking-app src/app/services/flight.spec.ts (8 tests) 46ms
✓ flight-booking-app src/app/components/login/login.spec.ts (17 tests) 3111ms
  ✓ should create 538ms
✓ flight-booking-app src/app/components/register/register.spec.ts (23 tests) 4625ms
  ✓ should create 570ms
✓ flight-booking-app src/app/components/flight-search/flight-search.spec.ts (25 tests) 5938ms
  ✓ should create 657ms
  ✓ should initialize search form with default values 304ms
  ✓ Should set travelDate to today by default 323ms
  ✓ should subscribe to currentUser on init 330ms
  ✓ should not search if form is invalid 320ms

Test Files 8 passed (8)
Tests 99 passed (99)
Start at 00:40:47
Duration 9.11s (transform 873ms, setup 2.55s, import 2.03s, tests 15.14s, environment 9.48s)

% Coverage report from v8

```

| File | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #s |
|------------------------------|--------|---------|--------|--------|---|
| All files | 71.4 | 75.67 | 88.88 | 78.04 | |
| app | 100 | 85.71 | 100 | 100 | |
| app.html | 100 | 100 | 100 | 100 | |
| app.ts | 100 | 85.71 | 100 | 100 | 10 |
| app/components/flight-search | 54.54 | 61.53 | 83.33 | 60.4 | |
| flight-search.html | 45.45 | 41.17 | 0 | 49.13 | 45-46,65-66,83-84,94-98,107-185 |
| flight-search.ts | 100 | 100 | 100 | 100 | |
| app/components/home | 98.38 | 100 | 80 | 100 | |
| home.html | 98.11 | 100 | 0 | 100 | |
| home.ts | 100 | 100 | 100 | 100 | |
| app/components/Login | 80.8 | 76 | 85.71 | 84.28 | |
| login.html | 75 | 50 | 0 | 76.59 | 29-30,48-49,53-54,59-63 |
| login.ts | 100 | 100 | 100 | 100 | |
| app/components/register | 69.36 | 75 | 87.5 | 82.3 | |
| register.html | 63.19 | 50 | 0 | 76.74 | 51-54,72-74,92-94,112-114,118-119,124-128 |
| register.ts | 100 | 100 | 100 | 100 | |
| app/guards | 100 | 100 | 100 | 100 | |
| auth-guard.ts | 100 | 100 | 100 | 100 | |
| app/services | 100 | 83.33 | 100 | 100 | |
| auth.ts | 100 | 80 | 100 | 100 | 30-47 |
| flight.ts | 100 | 100 | 100 | 100 | |

PASS Waiting for file changes...

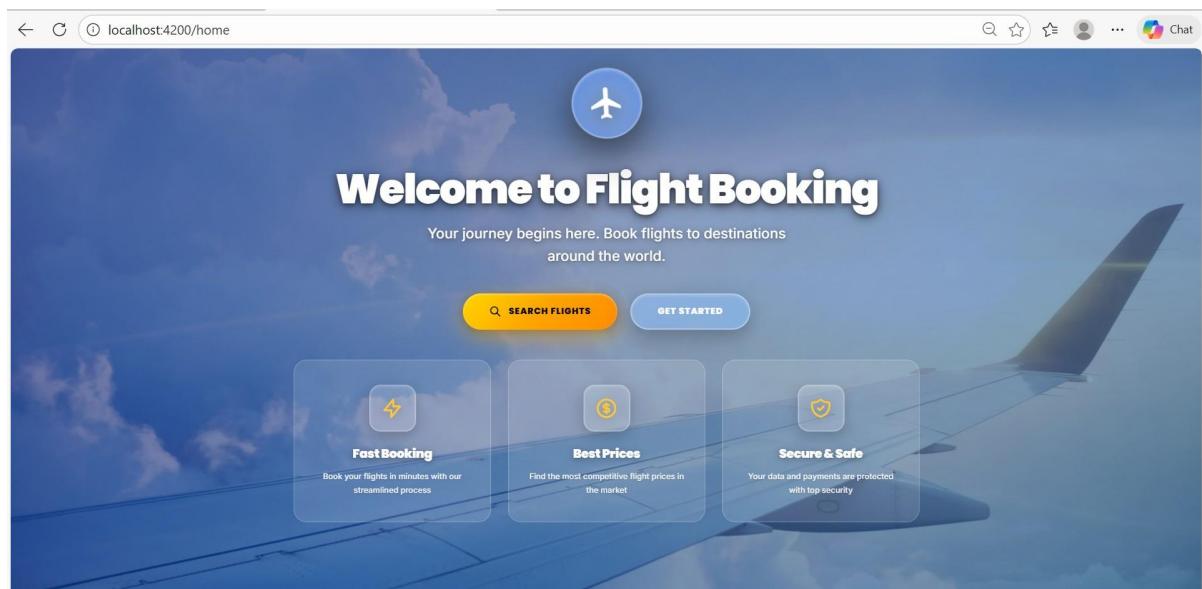
TEST RUN WITH UI

The screenshot shows the Vitest UI interface. On the left, there's a tree view of test files under 'FlightBookingApp'. The 'FlightSearch' component has 25 tests, all of which have passed. The 'Home' component has 10 tests, all of which have passed. The 'Auth' component has 14 tests, all of which have passed. The 'Flight' component has 8 tests, all of which have passed. The 'AuthGuard' component has 4 tests, all of which have passed. The 'AuthService' component has 8 tests, all of which have passed. The 'Auth' service has 14 tests, all of which have passed. The 'Flight' service has 8 tests, all of which have passed.

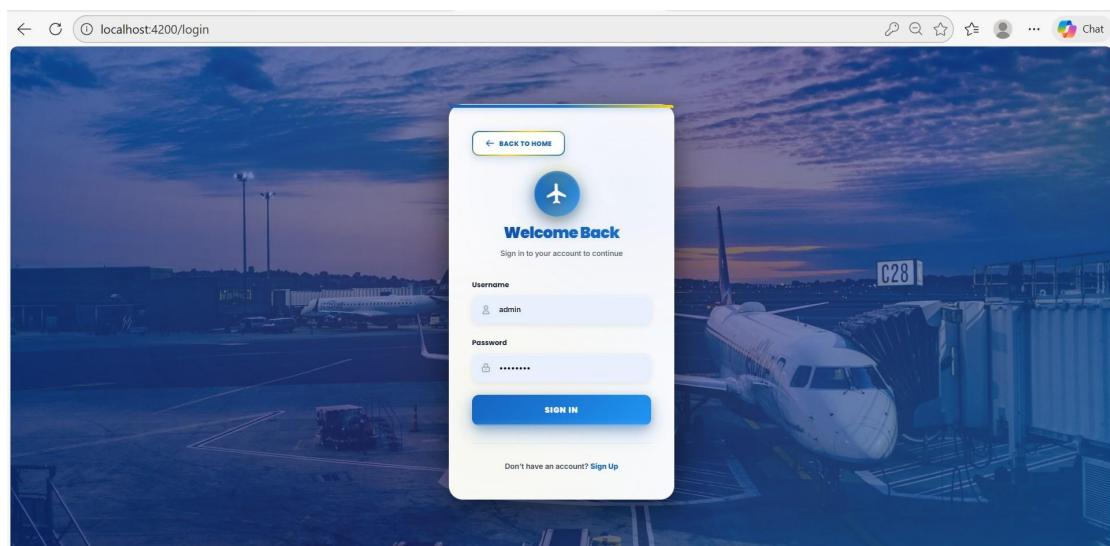
On the right, there's a summary of the test results:

- 99 Pass
- 0 Fail
- 99 Total
- 8 Files
- 8 Pass
- 0 Time: 51.21s

HOME PAGE



ADMIN LOGIN



ADMIN DASHBOARD

The screenshot shows the Admin Dashboard interface. On the left is a sidebar titled "Admin Panel" with options like Overview, Add Flight, View All Flights, Search Flights, Booking History, and Logout. The main area has a header "Welcome, Admin admin" and an "ADMIN" button. It displays four cards: "TOTAL FLIGHTS 45", "BOOKINGS TODAY 28", "ACTIVE USERS 156", and "REVENUE \$24,500". Below these are "Quick Actions" buttons for "Add New Flight", "View All Flights", "Search Flights", and "View Bookings". The background features a photograph of an airplane wing in flight.

ADD FLIGHT ADMIN FEATURE

The screenshot shows the "Add New Flight" form overlaid on a photograph of an airport runway at dusk. The form includes fields for Flight Number (e.g., AI101), Airline (e.g., Air India), Origin (e.g., DEL, BOM, BLR), Destination (e.g., DEL, BOM, BLR), Departure Time (dd-mm-yyyy --::--), Arrival Time (dd-mm-yyyy --::--), Available Seats (e.g., 180), and Price (e.g., 5000). There are "Cancel" and "Add Flight" buttons at the bottom. The background image shows an airplane at gate C28.

ADD FLIGHT VALIDATION ERROR

A screenshot of a web browser displaying the 'Add New Flight' form. The background shows a photograph of an airport tarmac at dusk. The form fields are outlined in red, indicating validation errors:

- Flight Number ***: e.g., AI01 (Error: Flight number is required)
- Airline ***: e.g., Air India (Error: Airline name is required)
- Origin ***: e.g., DEL, BOM, BLR (Error: Origin is required)
- Destination ***: e.g., DEL, BOM, BLR (Error: Destination is required)
- Departure Time ***: dd-mm-yyyy hh:mm (Error: Departure time is required)
- Arrival Time ***: dd-mm-yyyy hh:mm (Error: Arrival time is required)
- Available Seats ***: e.g., 180 (Error: Number of seats is required)
- Price (?) ***: e.g., 5000 (Error: Price is required)

At the bottom right of the form is a purple 'Add Flight' button.

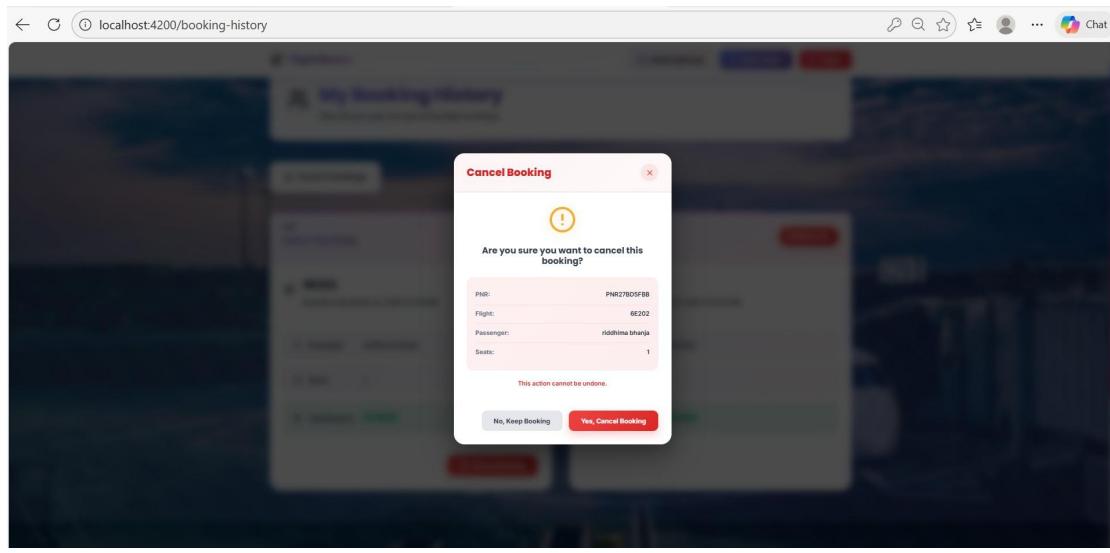
ALL ADDED FLIGHTS

A screenshot of a web browser displaying the 'All Registered Flights' dashboard. The background shows a photograph of an airport terminal at night. The dashboard features a grid of flight cards:

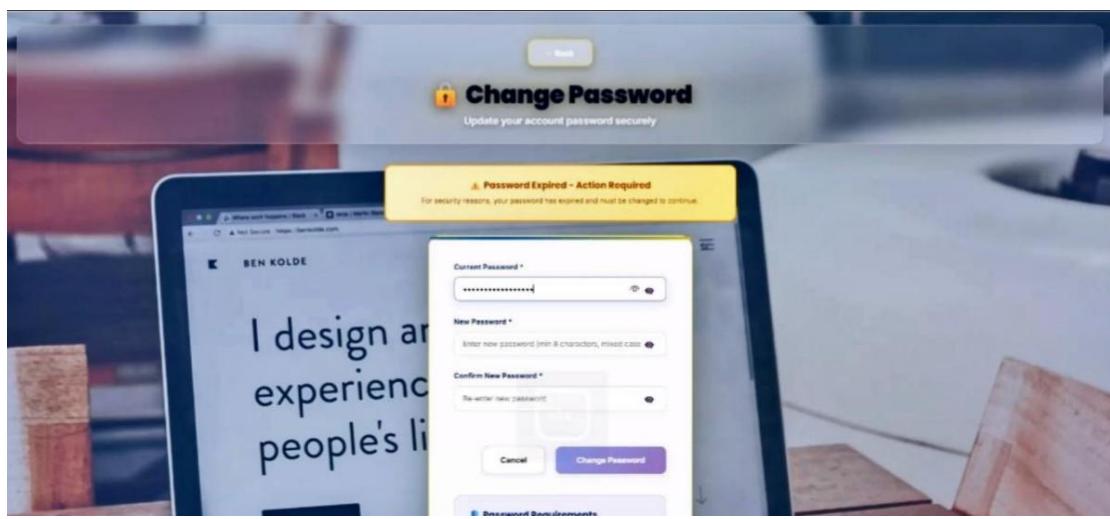
| ID | Airline | Flight ID | Origin | Destination | Time | Price |
|--------|-----------|-----------|--------|-------------|--------------------------|---------|
| ID: 1 | Air India | AI101 | DEL | BOM | 08:00 AM Dec 16, 2025 | ₹ 5,000 |
| ID: 2 | IndiGo | IE202 | DEL | BOM | 02:00 PM Dec 16, 2025 | ₹ 4,500 |
| ID: 3 | SpiceJet | SG303 | BOM | DEL | 09:00 AM Dec 16, 2025 | ₹ 4,200 |
| ID: 4 | Air India | AI404 | DEL | BLR | 07:00 AM Dec 16, 2025 | ₹ 6,000 |
| ID: 5 | IndiGo | IE505 | BLR | DEL | 03:00 PM Dec 16, 2025 | ₹ 5,800 |
| ID: 11 | Air India | AI909 | DEL | BLR | 08:00 AM Dec 20, 2025 | ₹ 6,500 |

The top right of the dashboard shows two buttons: 'TOTAL FLIGHTS 18' and 'ACTIVE 18'. The top left has a 'Back to Dashboard' link.

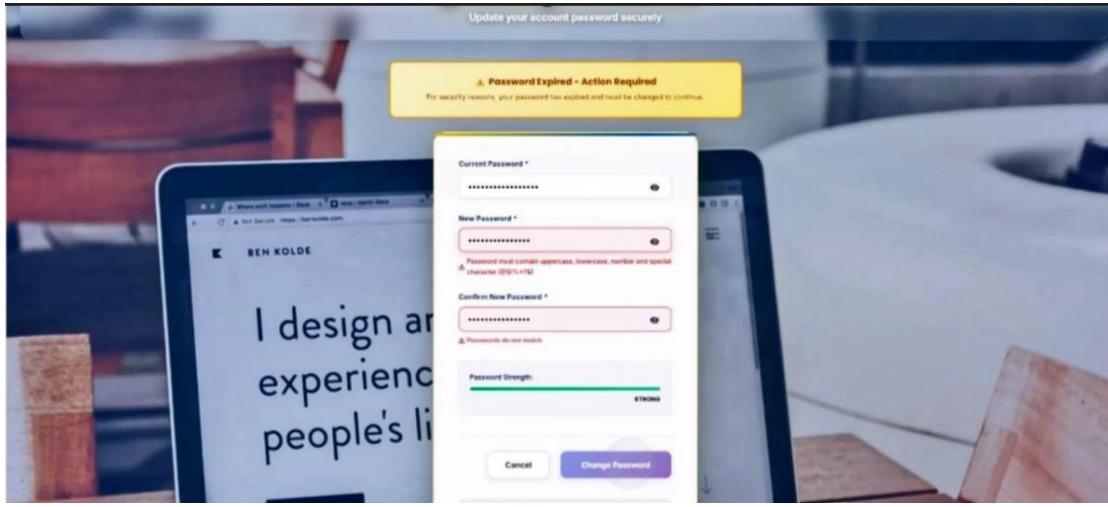
CANCEL BOOKING



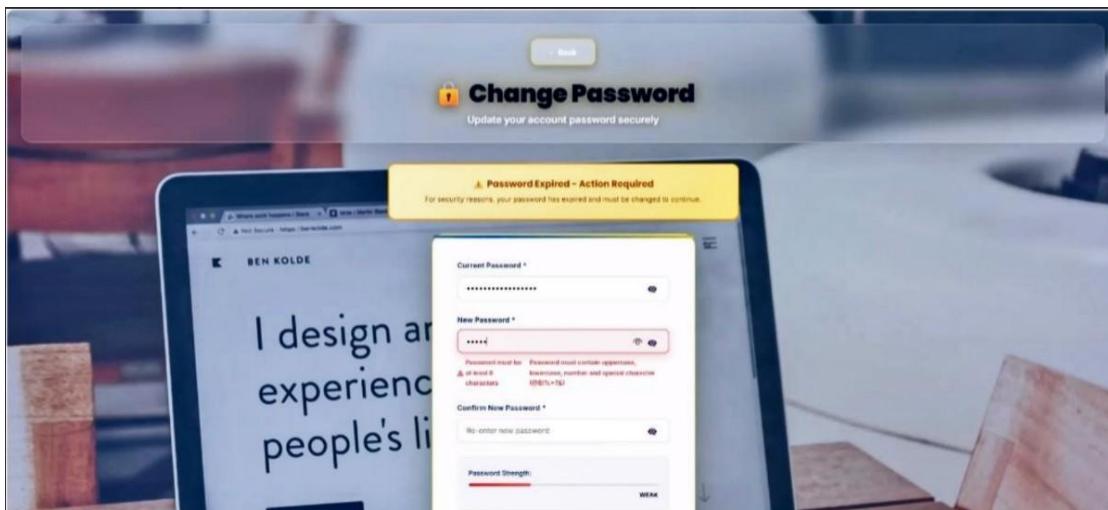
PASSWORD EXPIRATION ALERT



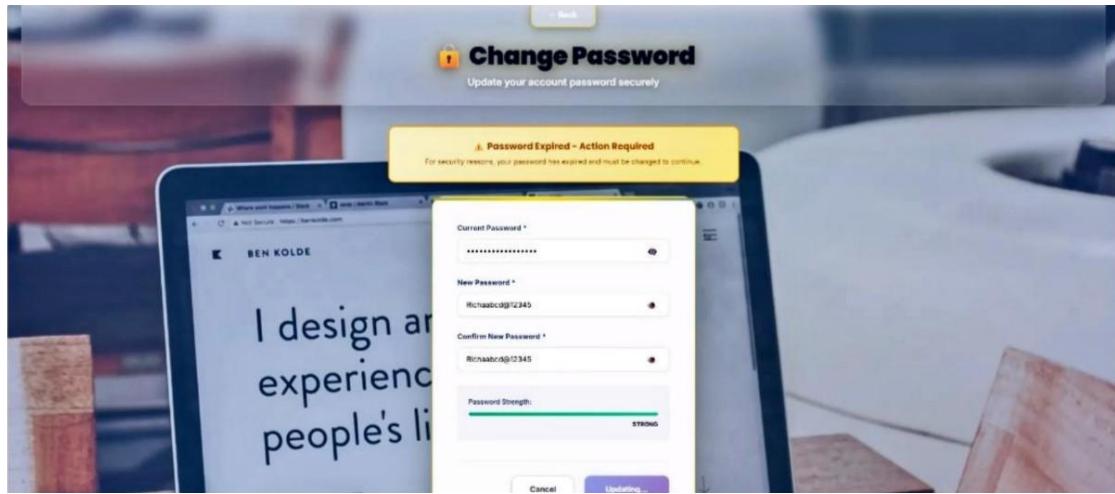
PASSWORD EXPIRY VALIDATION



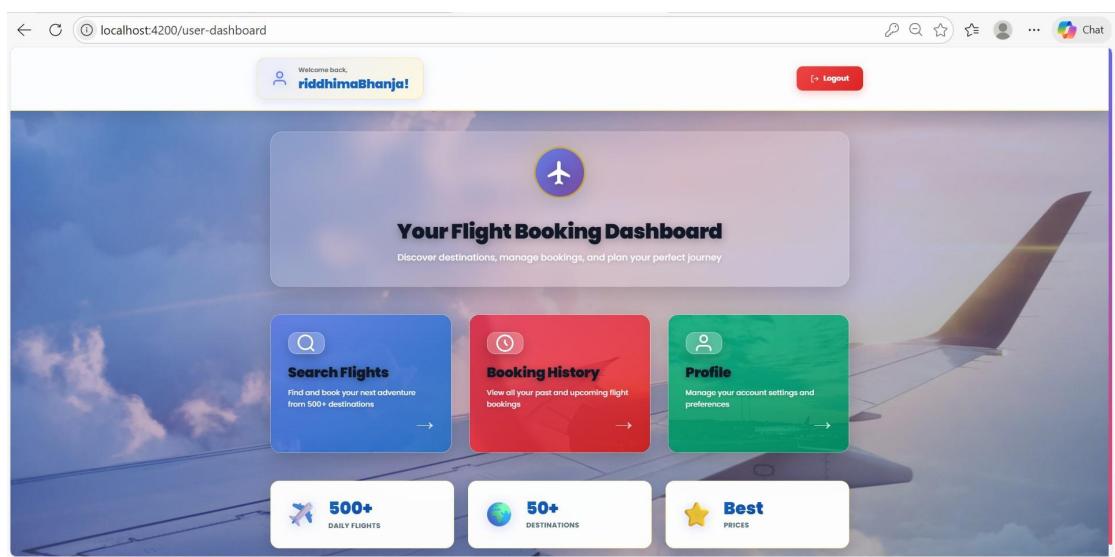
PASSWORD EXPIRY VALIDATION-2



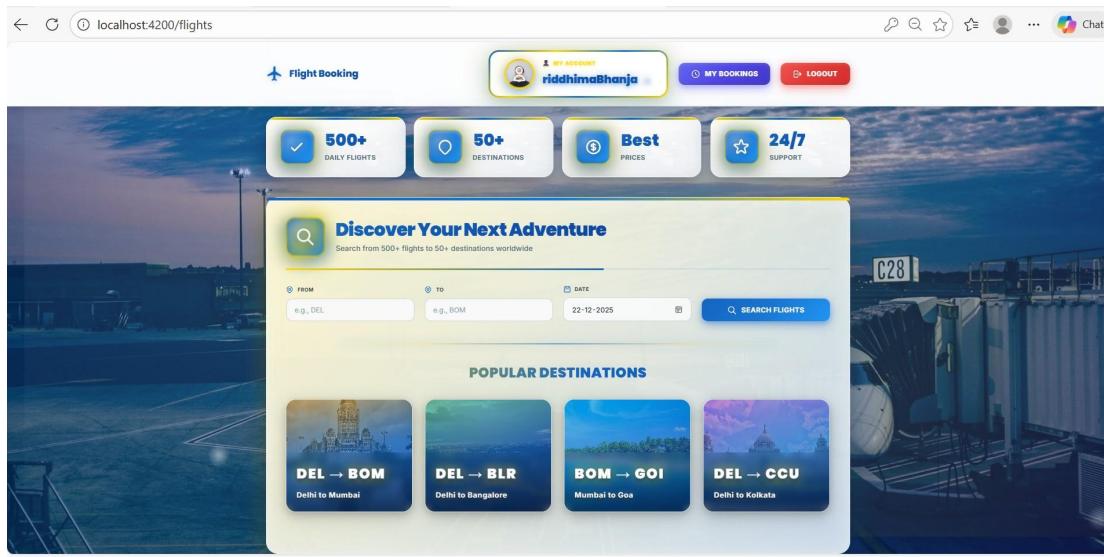
CHANGE PASSWORD SUCCESS



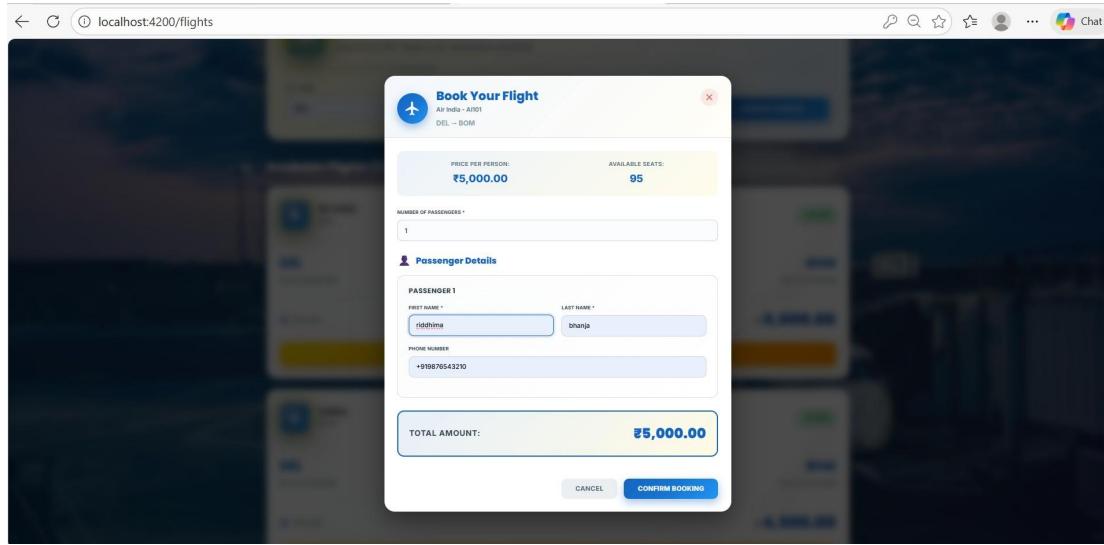
USER DASHBOARD



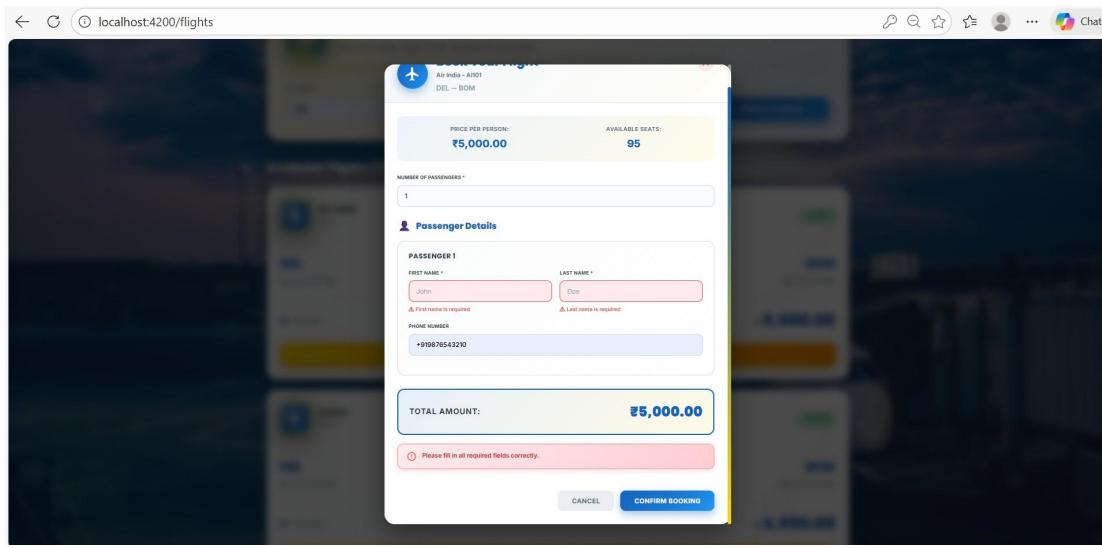
SEARCH FLIGHT



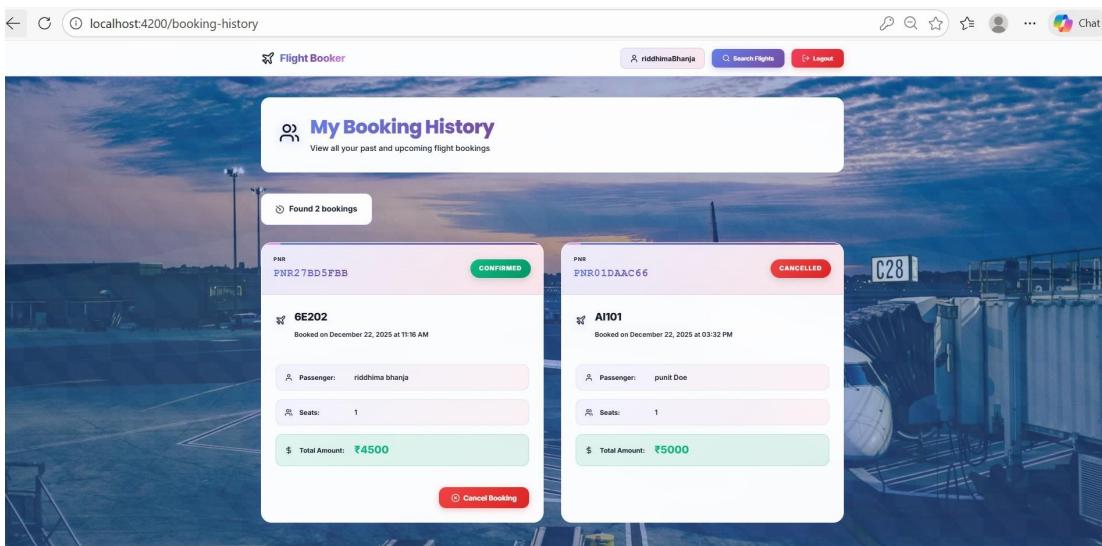
BOOK FLIGHT



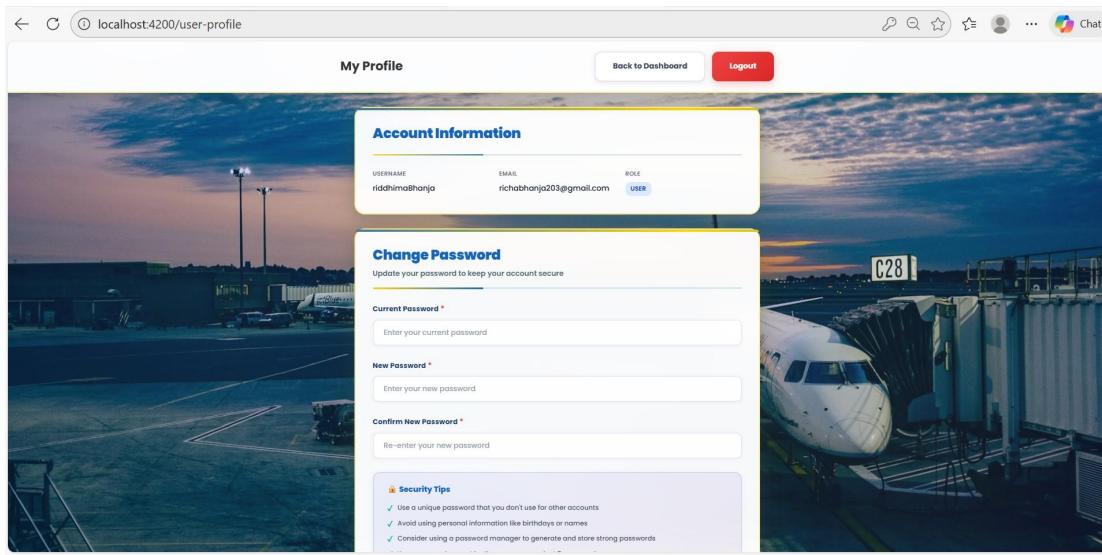
BOOK FLIGHT VALIDATION ERROR



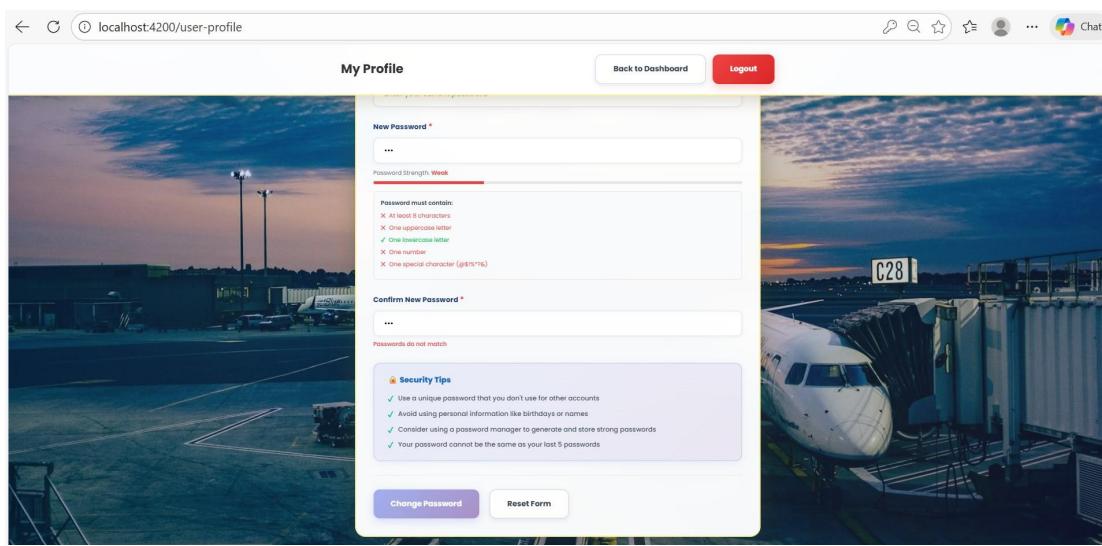
BOOKING HISTORY



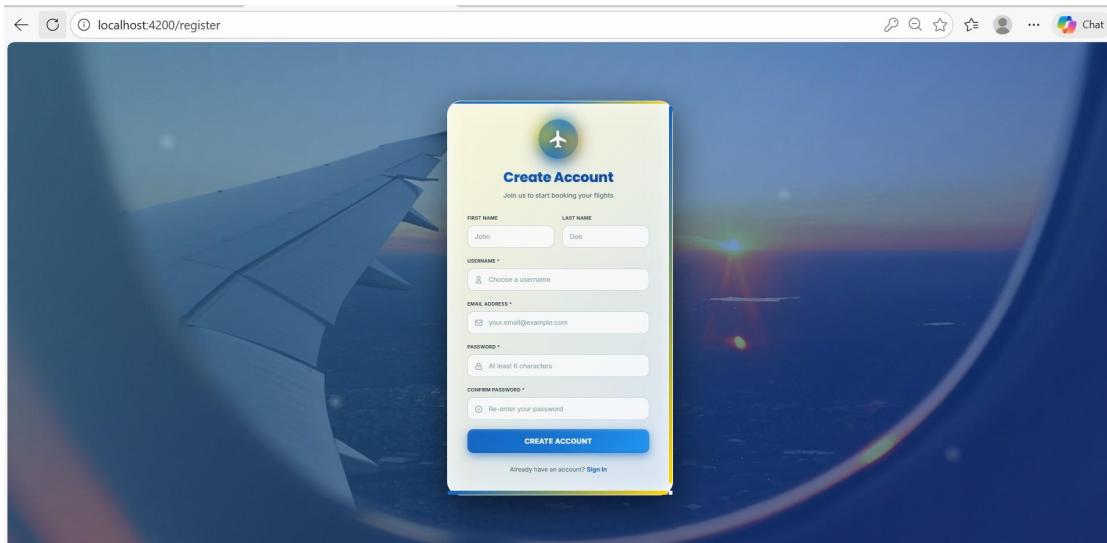
CHANGE PASSWORD FOR USER



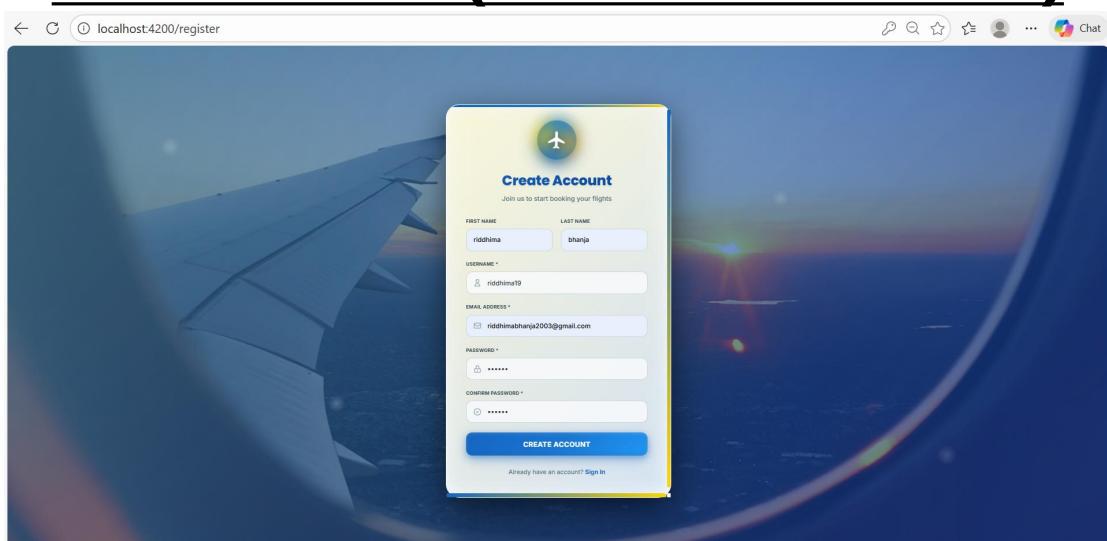
CHANGE PASSWORD VALIDATION ERROR



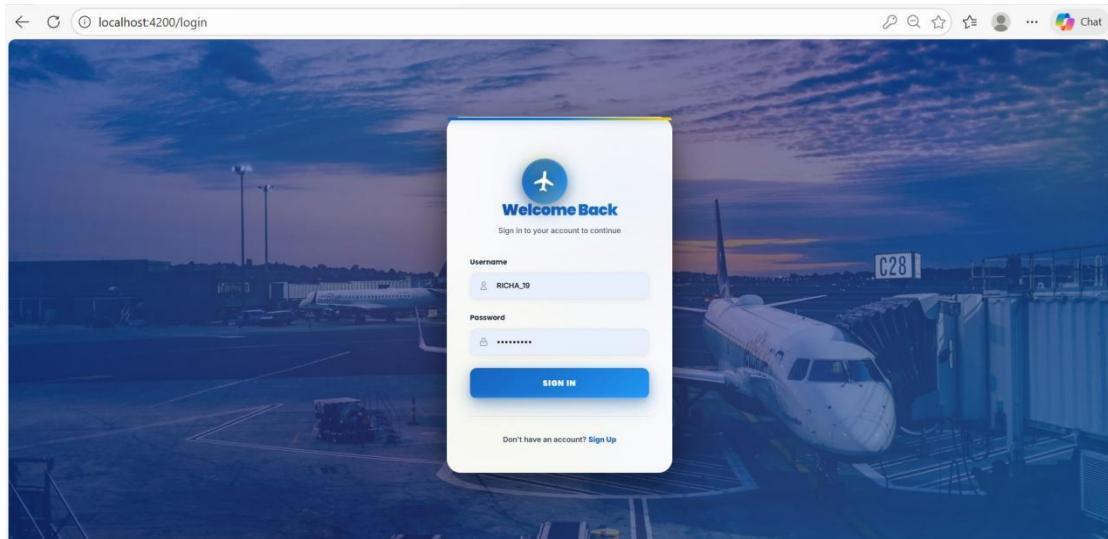
CLEAN REGISTER PAGE



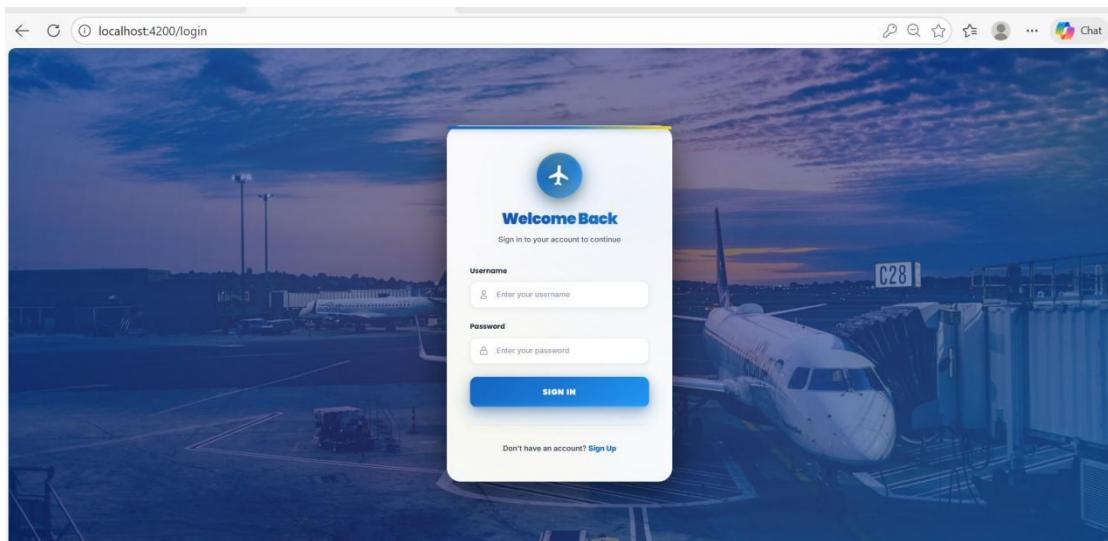
REGISTER PAGE(ACCOUNT CREATION)



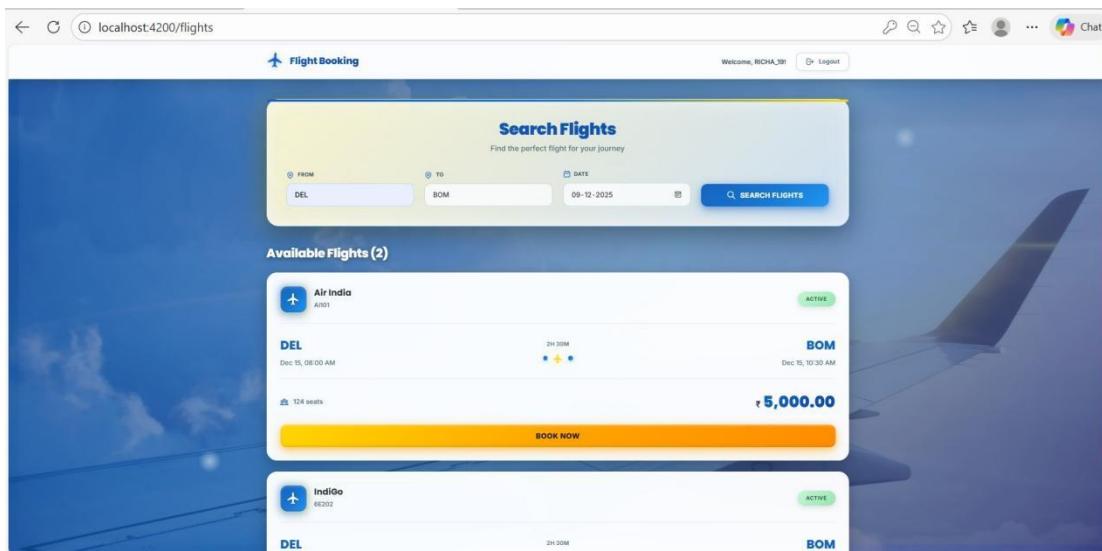
LOGIN PAGE



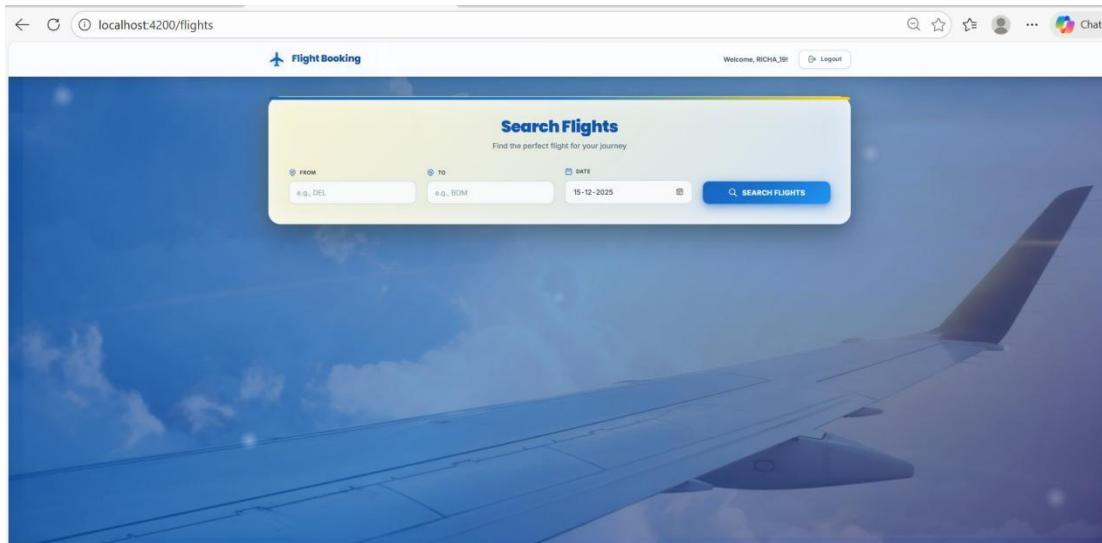
CLEAN LOGIN PAGE



SEARCH FLIGHTS

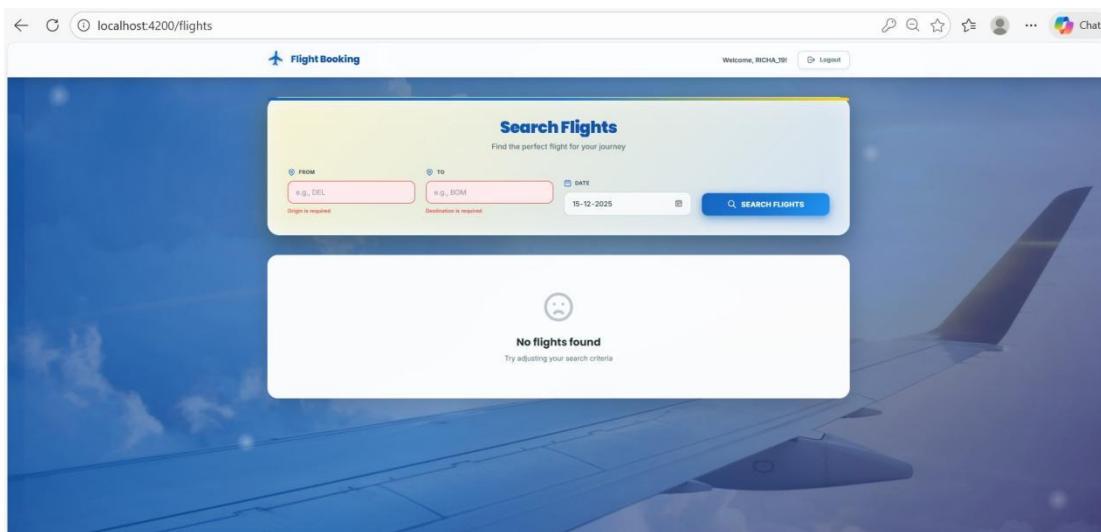


CLEAN SEARCH FLIGHTS

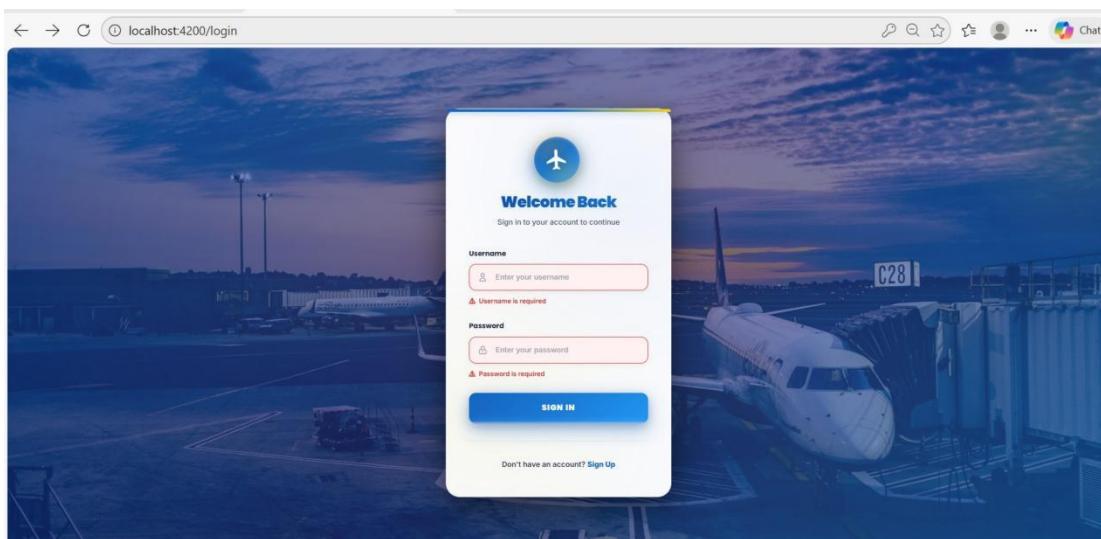


VALIDATION ERRORS

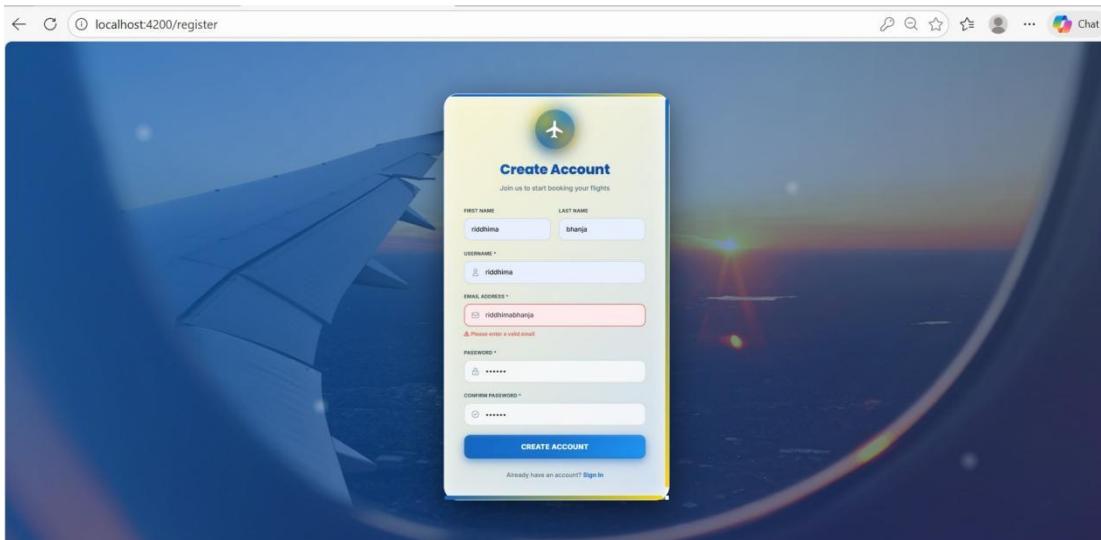
SEARCH FLIGHT VALIDATION ERROR



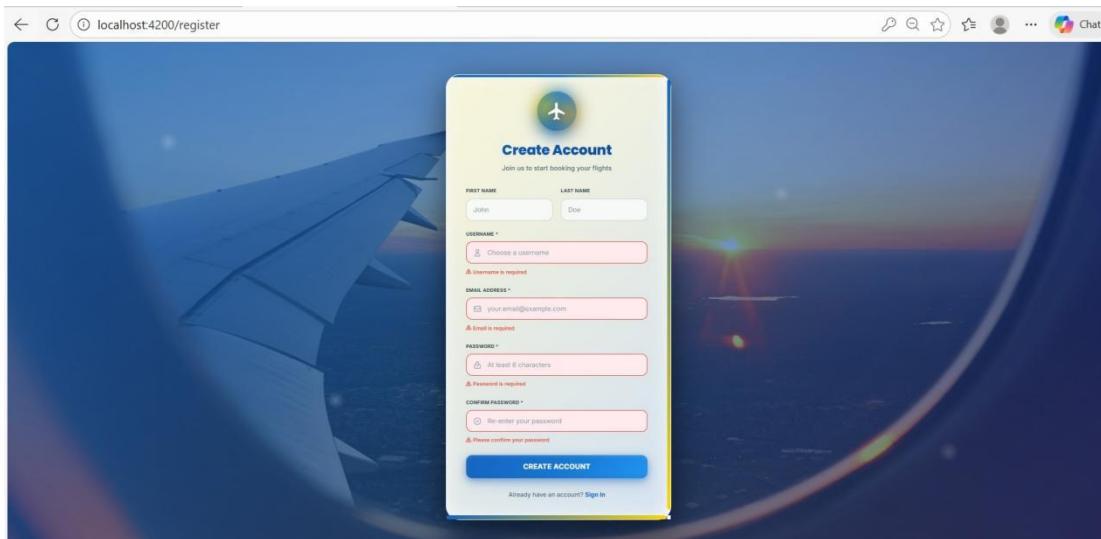
LOGIN VALIDATION ERROR



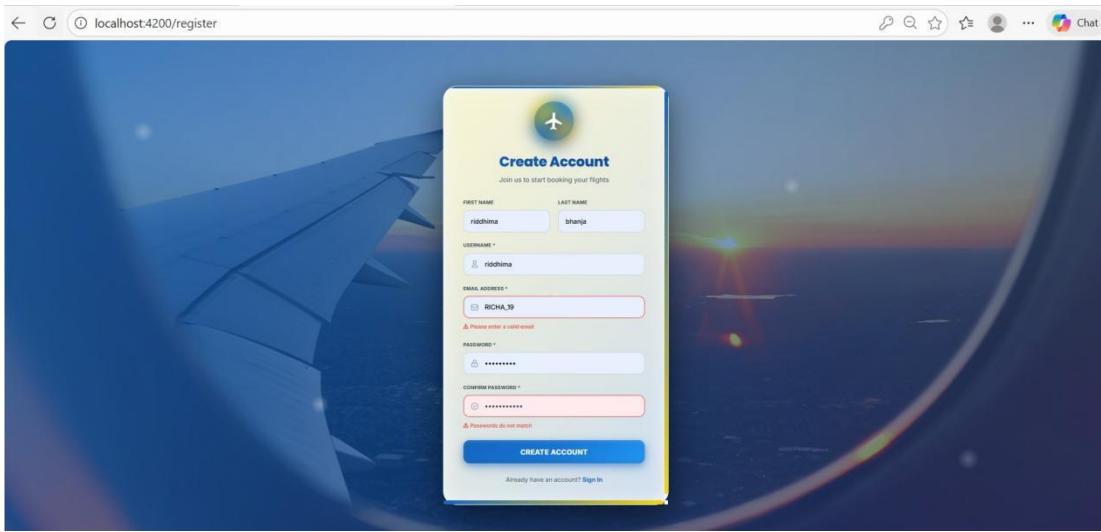
EMAIL VALIDATION ERROR



REGISTRATION VALIDATION ERROR



PASSWORD MISMATCH ERROR

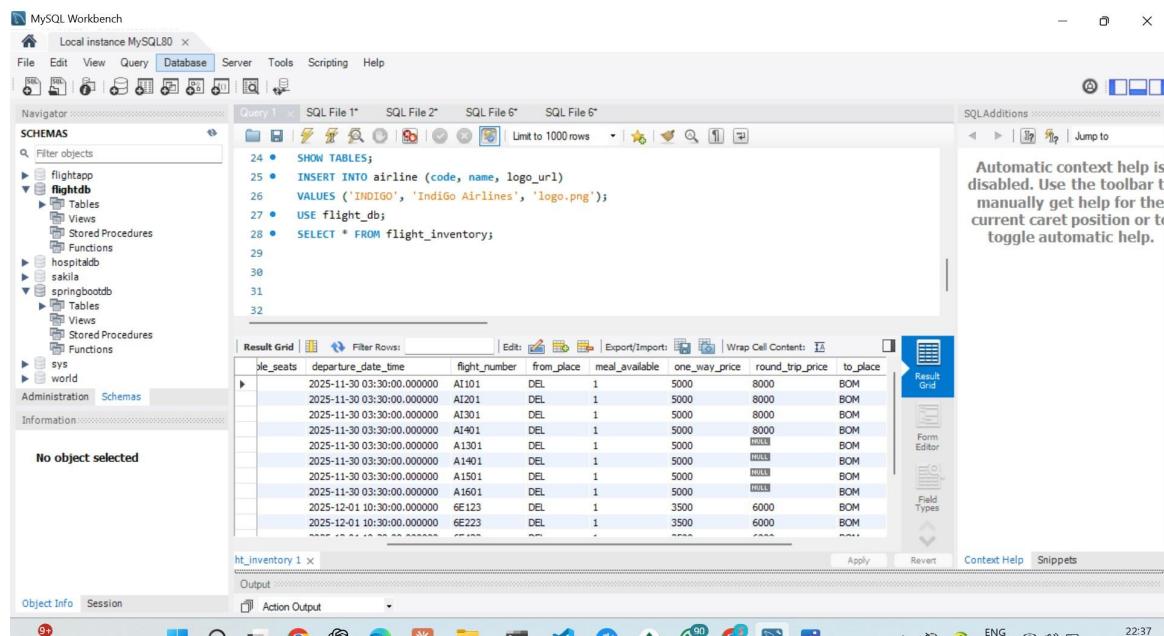


MONGODB DATA

localhost:27017 > flightdbwebflux > flight_inventory

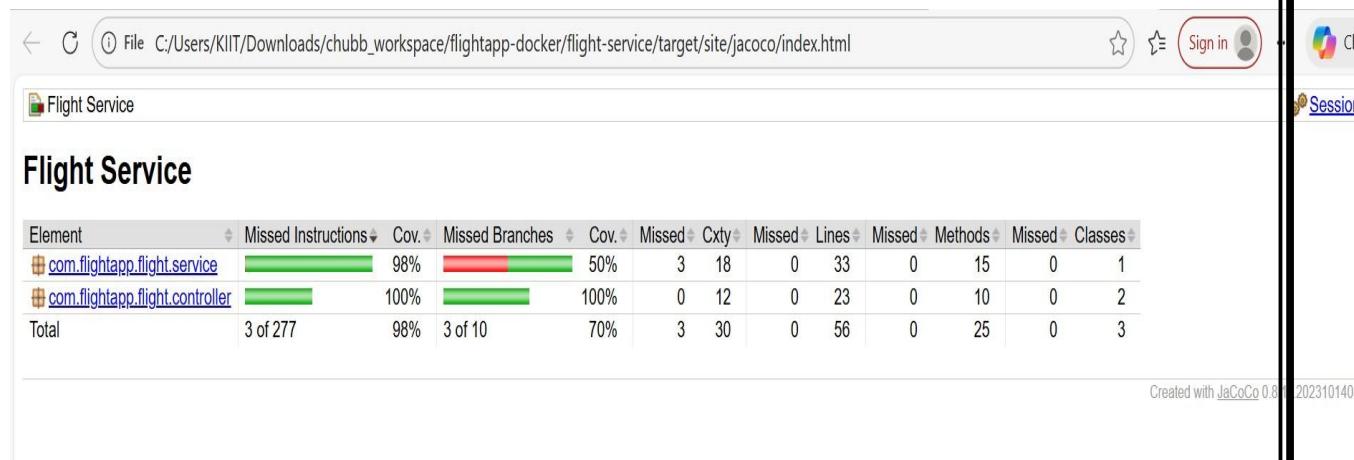
| _id | flightNumber | fromPlace | toPlace | departureDateTime | totalSeats | status | _class |
|---|--------------|-----------|---------|-------------------------------|------------|----------|--|
| <code>ObjectId('6922b0d421145447dfb631c0')</code> | "6E673" | "DEL" | "BOM" | 2025-12-01T05:00:00.000+00:00 | 50 | "ACTIVE" | "com.flightapp.entity.FlightInventory" |
| <code>ObjectId('6922b1ac21145447dfb631c1')</code> | "6E673" | "DEL" | "BOM" | 2025-12-01T05:00:00.000+00:00 | 50 | "ACTIVE" | "com.flightapp.entity.FlightInventory" |
| <code>ObjectId('6922b28621145447dfb631c2')</code> | "6E973" | "DEL" | "BOM" | 2025-12-01T05:00:00.000+00:00 | 50 | "ACTIVE" | "com.flightapp.entity.FlightInventory" |

MYSQL WORKBENCH



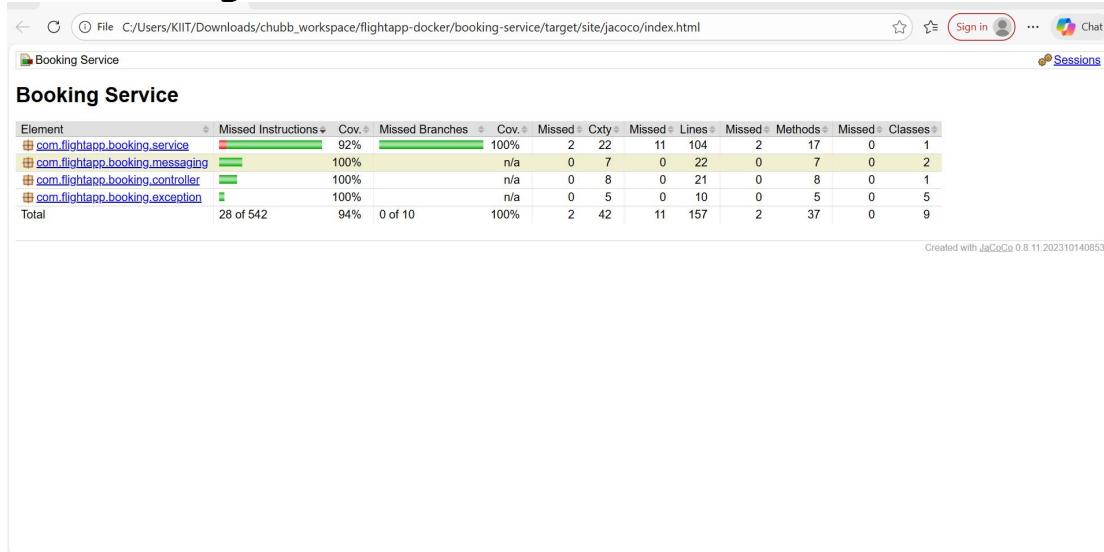
JACOCO REPORTS

**FLIGHT SERVICE:
98% COVERAGE**

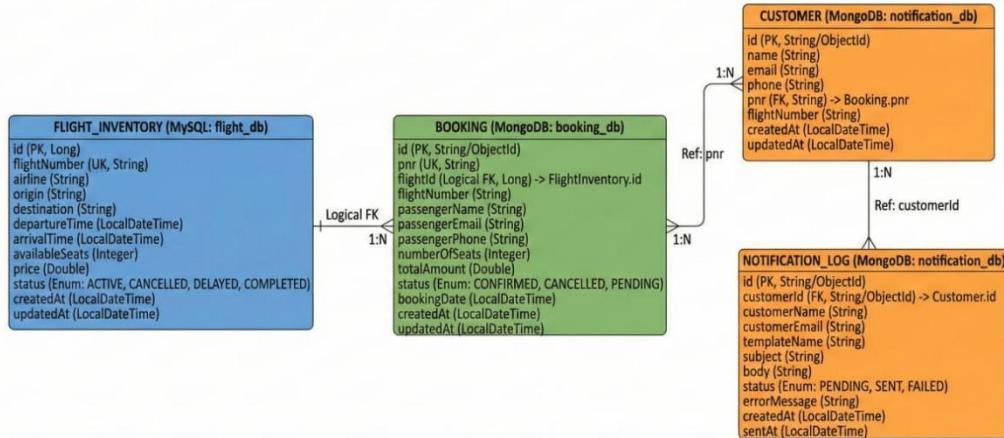


BOOKING SERVICE

94% coverage



ER DIAGRAM



1. SonarQUBE Code Coverage

The screenshot shows the SonarQUBE Cloud dashboard for the project 'Riddhima Bhanja > API Gateway > master'. The top navigation bar includes 'My Projects', 'My Issues', and 'Explore'.

The main summary section displays the following metrics:

- Security:** 0 Open issues (A)
- Reliability:** 0 Open issues (A)
- Maintainability:** 0 Open issues (A)
- Accepted Issues:** 0 (grey circle)
- Coverage:** 83.3% (green circle)
No conditions set on 12 Lines to cover
- Duplications:** 0.0% (green circle)
No conditions set on 100 Lines

Below these metrics, there is a section for 'Security Hotspots'.

2. SonarQUBE Issues

Before fixing:

The screenshot shows the SonarQUBE Cloud Issues page for the project 'Riddhima Bhanja > docker > main'. The left sidebar shows the project structure with 'Main Branch' selected.

The 'Issues' tab is active, displaying a list of 32 open and confirmed issues. The filters section on the left allows filtering by severity (Blocker, High, Medium, Low, Info) and type (Code attribute, Type).

The right side of the screen lists three specific issues with their details:

- Remove this field injection and use constructor injection instead.**
Reliability: Medium, Maintainability: Medium
L13 · 5min effort · 6 minutes ago · Code Smell · Major
- Inject this field value directly into "customRouteLocator", the only method that uses it.**
Maintainability: High
L14 · 5min effort · 6 minutes ago · Code Smell · Critical
- Define a constant instead of duplicating this literal "lb://flight-service" 4 times.**
Maintainability: High
L23 · 10min effort · 6 minutes ago · Code Smell · Critical

Screenshot of the SonarCloud project summary for the 'docker' project. The main statistics are:

| Category | Value |
|-------------------|---|
| Security | 1 Open issues (E) |
| Reliability | 4 Open issues (C) |
| Maintainability | 29 Open issues (A) |
| Accepted Issues | 0 |
| Coverage | A few extra steps are needed for SonarQube Cloud to analyze your code coverage. Set up coverage analysis |
| Duplications | 0.7% No conditions set on 3k Lines |
| Security Hotspots | 0 |

The sidebar shows the project structure: Riddhima Bhanja > docker > main. The navigation bar includes Summary, Issues, Security Hotspots, and More.

After fixing:

Screenshot of the SonarCloud project summary for the 'FlightApp_Docker-Assigned7' project. The main statistics are:

| Category | Value |
|------------------|--------------------------------------|
| Software quality | 0 |
| Reliability | 0 |
| Maintainability | 0 |
| Severity | 0 (Blocker, High, Medium, Low, Info) |
| Code attribute | 0 |
| Type | 0 |

The sidebar shows the project structure: Riddhima Bhanja > FlightApp_Docker-Assigned7 > main. The navigation bar includes Summary, Issues, Security Hotspots, and More. A message in the center says "No Issues. Hooray!"

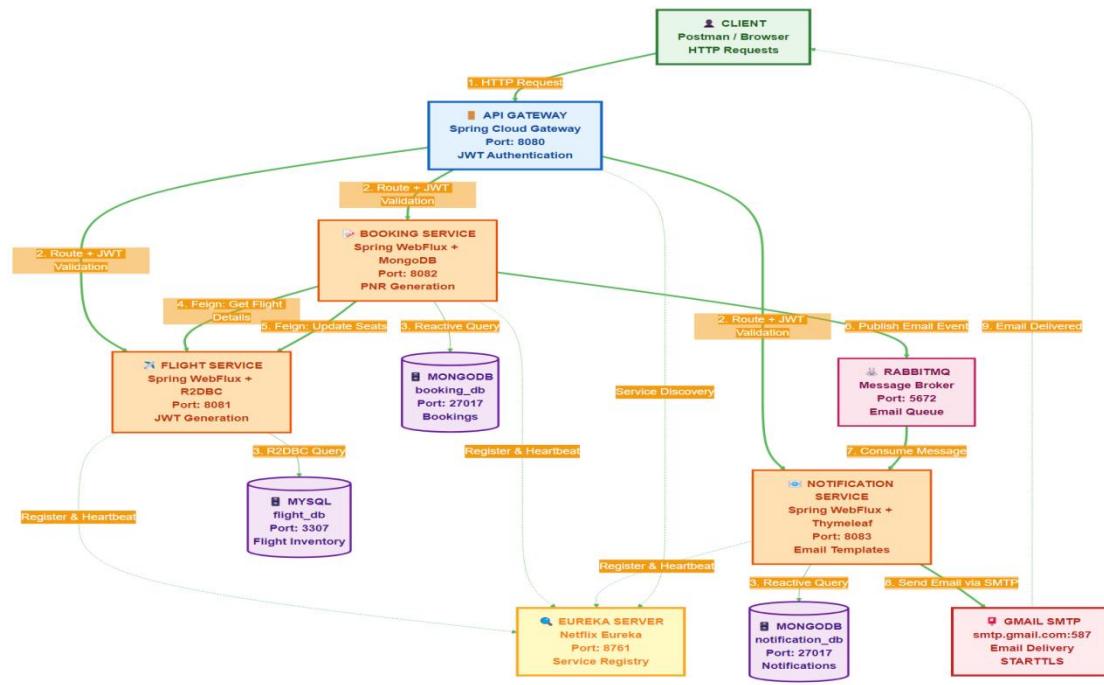
Screenshot of the SonarCloud project summary page for 'FlightApp_Docker-Assignment7' (main branch). The page displays various metrics and analysis results.

Project Summary:

- Security:** 0 Open issues (A)
- Reliability:** 0 Open issues (A)
- Maintainability:** 0 Open issues (A)
- Accepted Issues:** 0
- Coverage:** A few extra steps are needed for SonarQube Cloud to analyze your code coverage. Set up coverage analysis.
- Duplications:** 0.8% (No conditions set on 2.7k Lines)
- Security Hotspots:** 0

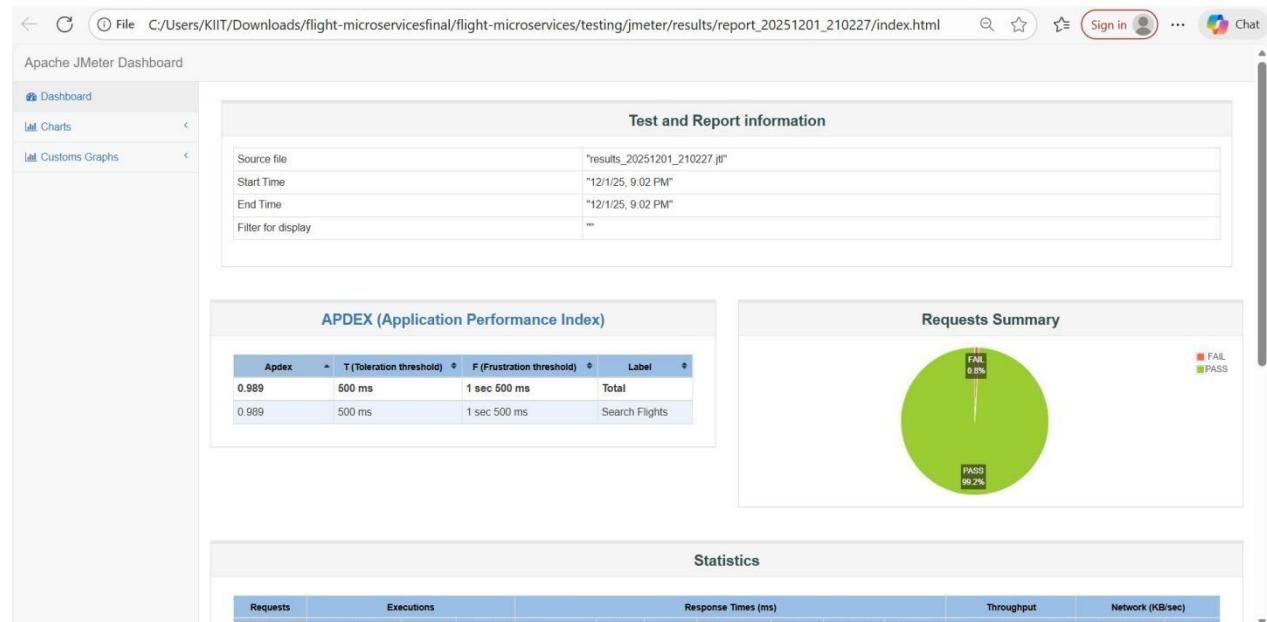
Navigation: Overview, Main Branch (selected), Pull Requests, Branches (1), Information, Administration.

3. System Architecture



4. Jmeter

- Apache Jmeter Dashboard**

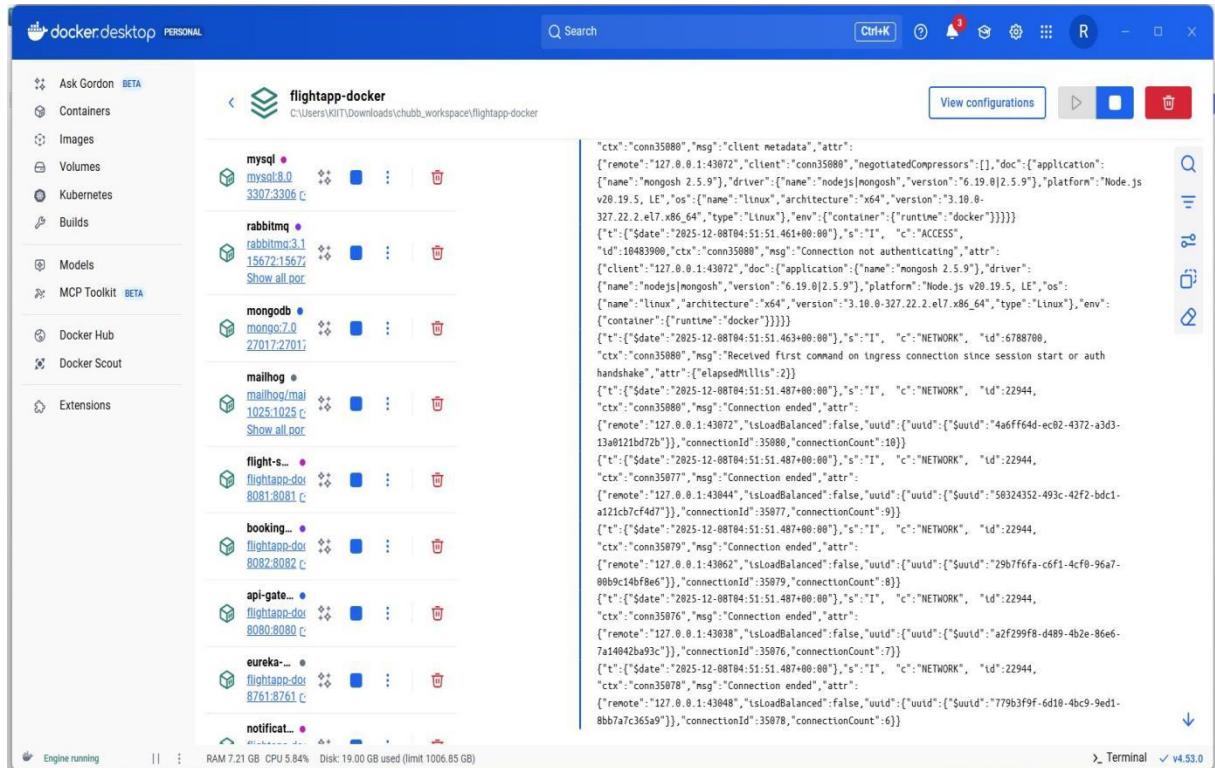


This screenshot provides a detailed view of the 'Statistics' section from the previous dashboard. It includes three tables:

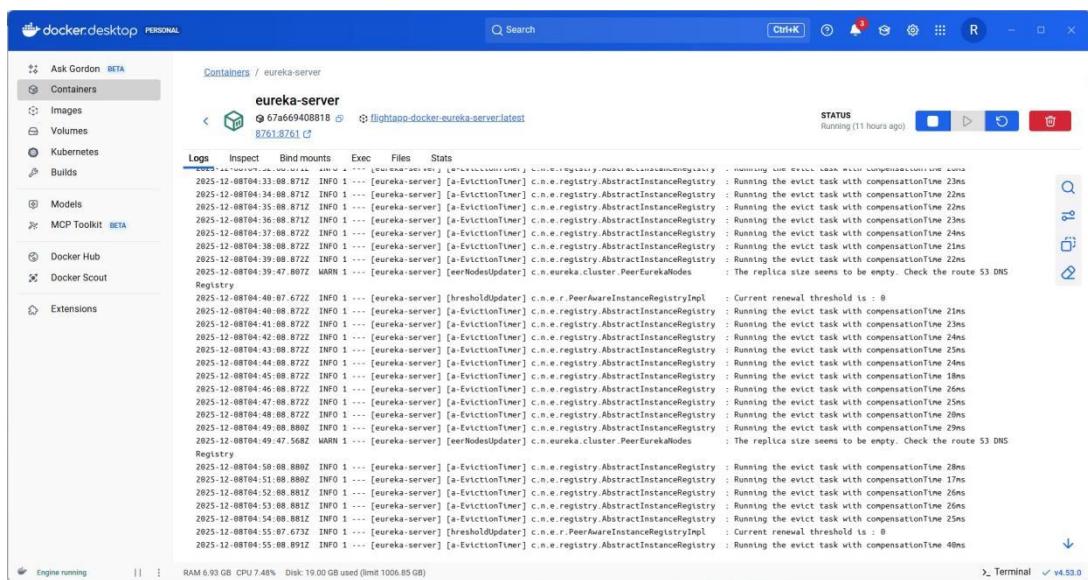
- Requests:** Compares 'Total' and 'Search Flights' across metrics like #Samples, #FAIL, Error %, Average, Min, Max, Median, 90th pct, 95th pct, 99th pct, Transactions/s, Received, and Sent.
- Errors:** Shows errors by type (e.g., 405/Method Not Allowed) with their counts and percentages.
- Top 5 Errors by sampler:** Lists errors by sample (Total and Search Flights) and error type.

LOGS SCREENSHOTS

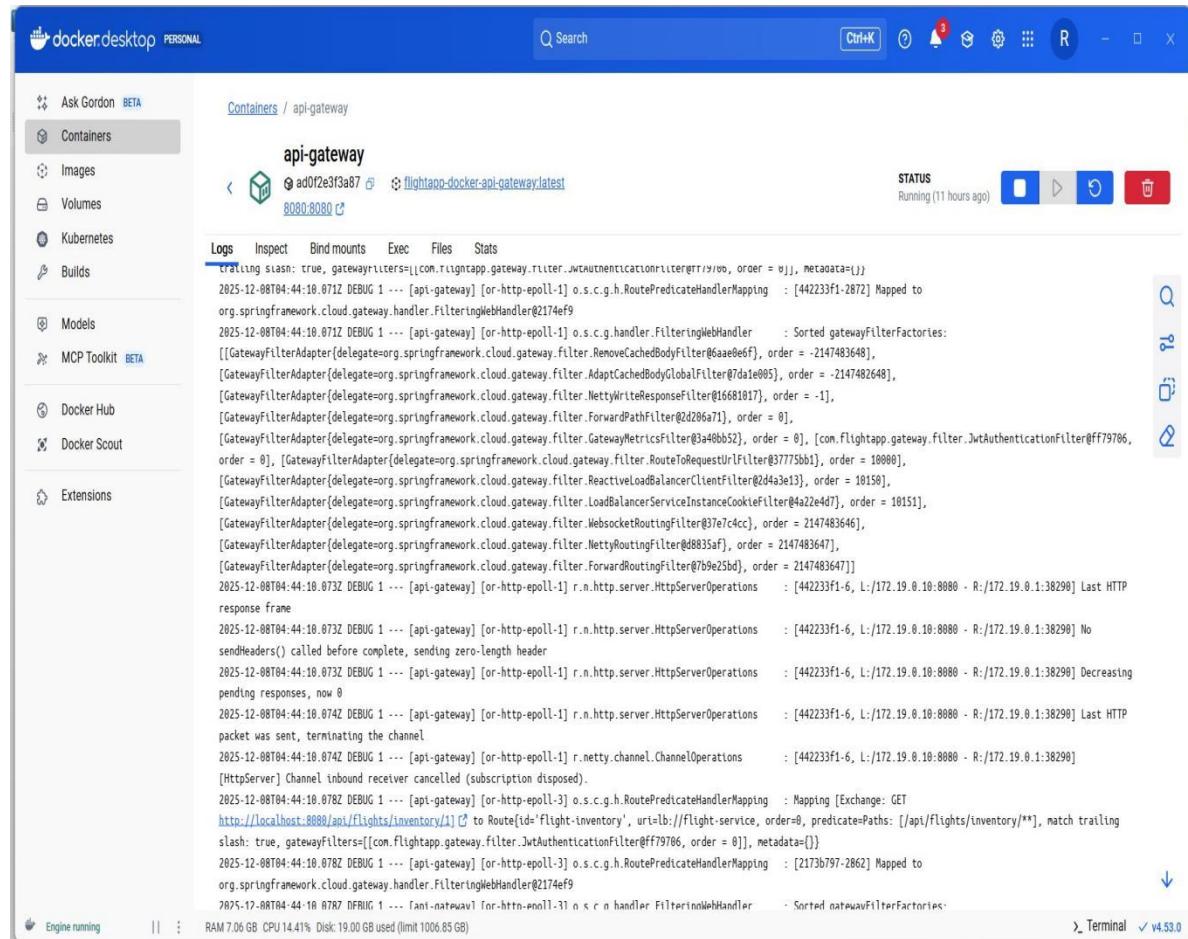
DASHBOARD



EUREKA SERVER LOGS



API GATEWAY



Docker Desktop Personal

Containers / api-gateway

api-gateway

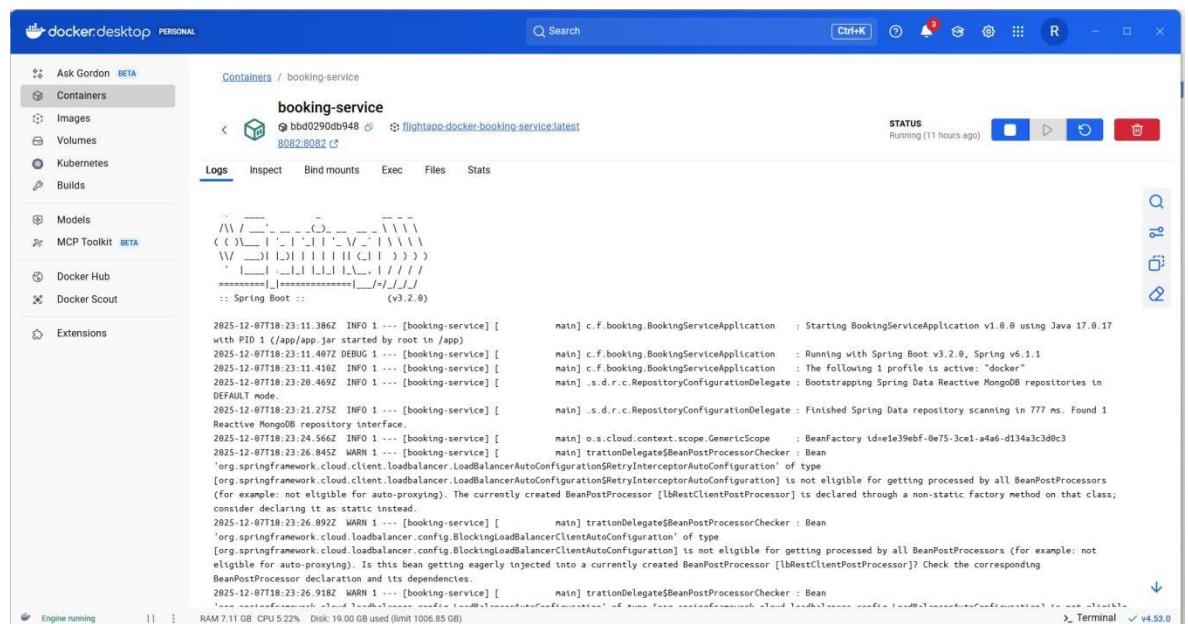
Status: Running (11 hours ago)

Logs Inspect Bind mounts Exec Files Stats

```
2025-12-08T04:44:10.071Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] o.s.c.g.h.RoutePredicateHandlerMapping : [442233f1-2872] Mapped to org.springframework.cloud.gateway.handler.FilteringWebHandler@2174ef9
2025-12-08T04:44:10.071Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] o.s.c.g.h.RoutePredicateHandlerMapping : Sorted gatewayFilterFactories: [[GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.RemoveCachedBodyFilter@6aae0ef], order = -2147483648], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.AdaptCachedBodyGlobalFilter@7da1e005], order = -2147482648], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.NettyUrlResponseFilter@16681017], order = -1], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.ForwardPathFilter@2d206a71], order = 0], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.GatewayMetricsFilter@3a4bb52], order = 0], [com.flighthapp.gateway.filter.JwtAuthenticationFilter@ff7978, order = 0], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.RouteToRequestUrlFilter@37775bb1], order = 10000], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.ReactiveLoadBalancerFilter@2d4a3e13], order = 18150], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.LoadBalancerServiceInstanceCookieFilter@4a22e4d7], order = 18151], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.WebsocketRoutingFilter@37e74cc], order = 2147483646], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.NettyRoutingFilter@08835af], order = 2147483647], [GatewayFilterAdapter[delegate=org.springframework.cloud.gateway.filter.ForwardRoutingFilter@79e25bd], order = 2147483647]]]
2025-12-08T04:44:10.073Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] r.n.http.server.HttpServerOperations : [442233f1-6, L:/172.19.0.1:38290] Last HTTP response frame
2025-12-08T04:44:10.073Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] r.n.http.server.HttpServerOperations : [442233f1-6, L:/172.19.0.1:38290] No sendHeaders() called before complete, sending zero-length header
2025-12-08T04:44:10.073Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] r.n.http.server.HttpServerOperations : [442233f1-6, L:/172.19.0.1:38290] Decreasing pending responses, now 0
2025-12-08T04:44:10.074Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] r.n.http.server.HttpServerOperations : [442233f1-6, L:/172.19.0.1:38290] Last HTTP packet was sent, terminating the channel
2025-12-08T04:44:10.074Z DEBUG 1 --- [api-gateway] [or-http-epoll-1] r.netty.channel.ChannelOperations : [442233f1-6, L:/172.19.0.1:38290] [HttpServer] Channel inbound receiver cancelled (subscription disposed).
2025-12-08T04:44:10.078Z DEBUG 1 --- [api-gateway] [or-http-epoll-3] o.s.c.g.h.RoutePredicateHandlerMapping : Mapping [Exchange: GET http://localhost:8080/api/flights/inventory/{id}] to Route[id='flight-inventory', uri=/b://flight-service, order=0, predicatePaths: [/api/flights/inventory/**], match trailing slash=true, gatewayFilters=[[com.flighthapp.gateway.filter.JwtAuthenticationFilter@ff7978, order = 0]], metadata={}]
2025-12-08T04:44:10.078Z DEBUG 1 --- [api-gateway] [or-http-epoll-3] o.s.c.g.h.RoutePredicateHandlerMapping : [2173b797-2862] Mapped to org.springframework.cloud.gateway.handler.FilteringWebHandler@2174ef9
2025-12-08T04:44:10.078Z DEBUG 1 --- [api-gateway] [or-http-epoll-3] o.s.c.g.h.RoutePredicateHandlerMapping : Sorted gatewayFilterFactories:
```

Engine running | RAM: 7.06 GB CPU: 14.41% Disk: 19.00 GB used (limit 1006.65 GB) Terminal v4.53.0

BOOKING SERVICE LOGS



Docker Desktop Personal

Containers / booking-service

booking-service

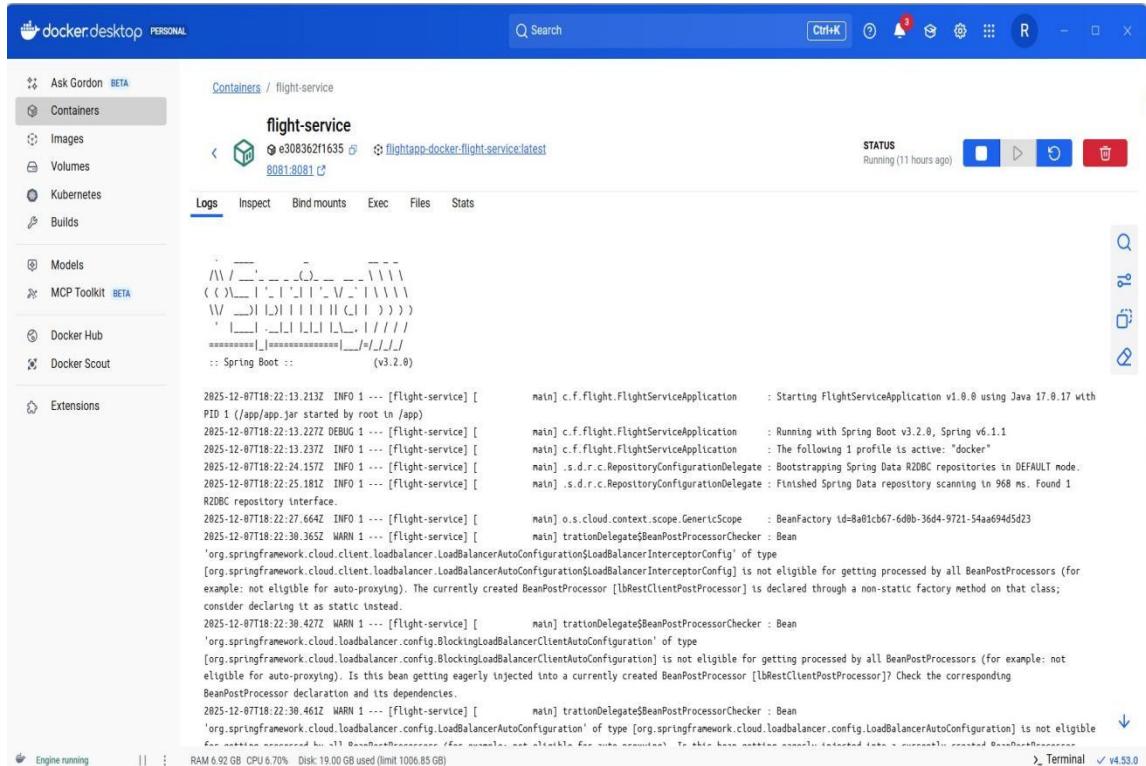
Status: Running (11 hours ago)

Logs Inspect Bind mounts Exec Files Stats

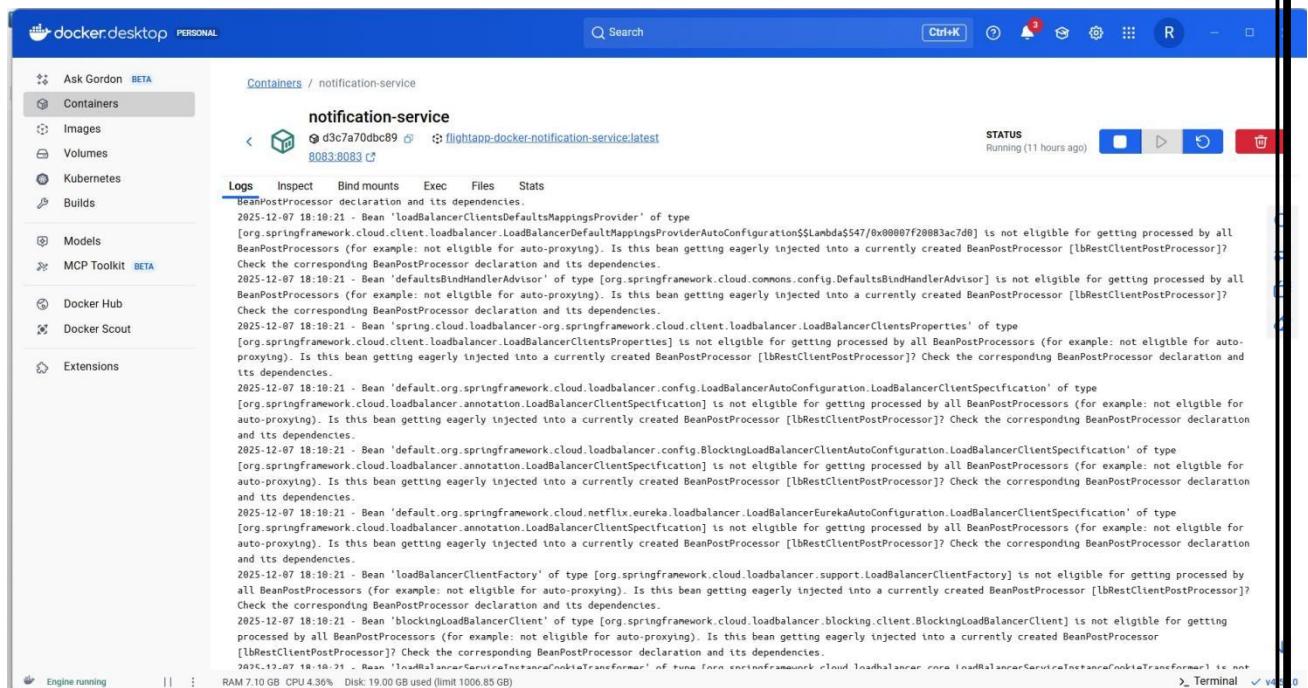
```
2025-12-07T18:23:11.386Z INFO 1 --- [booking-service] [main] c.f.booking.BookingServiceApplication : Starting BookingServiceApplication v1.0.0 using Java 17.0.17
with PID 1 in /app/app.jar started by root in /app
2025-12-07T18:23:11.407Z DEBUG 1 --- [booking-service] [main] c.f.booking.BookingServiceApplication : Running with Spring Boot v3.2.0, Spring v6.1.1
2025-12-07T18:23:11.410Z INFO 1 --- [booking-service] [main] c.f.booking.BookingServiceApplication : The following 1 profile is active: "docker"
2025-12-07T18:23:20.469Z INFO 1 --- [booking-service] [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Reactive MongoDB repositories in DEFAULT mode.
2025-12-07T18:23:21.275Z INFO 1 --- [booking-service] [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 777 ms. Found 1 Reactive MongoDB repository interface.
2025-12-07T18:23:24.566Z INFO 1 --- [booking-service] [main] o.s.cloud.context.scope.GenericScope : BeanFactory id=d4e139ebf-0e75-3c61-a4a6-d134a3c3d0c3
2025-12-07T18:23:26.845Z WARN 1 --- [booking-service] [main] trationDelegatedBeanPostProcessorChecker : Bean 'org.springframework.cloud.client.loadbalancer.config.BlockingLoadBalancerClientAutoConfiguration' of type [org.springframework.cloud.client.loadbalancer.config.BlockingLoadBalancerClientAutoConfiguration] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying). The currently created BeanPostProcessor [lbRestClientPostProcessor] is declared through a non-static factory method on that class; consider declaring it as static instead.
2025-12-07T18:23:26.892Z WARN 1 --- [booking-service] [main] trationDelegatedBeanPostProcessorChecker : Bean 'org.springframework.cloud.loadbalancer.config.BlockingLoadBalancerClientAutoConfiguration' of type [org.springframework.cloud.loadbalancer.config.BlockingLoadBalancerClientAutoConfiguration] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying). Is this bean getting eagerly injected into a currently created BeanPostProcessor [lbRestClientPostProcessor]? Check the corresponding BeanPostProcessor declaration and its dependencies.
2025-12-07T18:23:26.918Z WARN 1 --- [booking-service] [main] trationDelegatedBeanPostProcessorChecker : Bean 'org.springframework.cloud.loadbalancer.config.BlockingLoadBalancerClientAutoConfiguration' of type [org.springframework.cloud.loadbalancer.config.BlockingLoadBalancerClientAutoConfiguration] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying). Is this bean getting eagerly injected into a currently created BeanPostProcessor [lbRestClientPostProcessor]? Check the corresponding BeanPostProcessor declaration and its dependencies.
```

Engine running | RAM: 7.11 GB CPU: 5.22% Disk: 19.00 GB used (limit 1006.65 GB) Terminal v4.53.0

FLIGHT SERVICE LOGS



NOTIFICATION SERVICE LOGS



RabbitMQ logs

The screenshot shows the Docker Desktop interface. On the left, the sidebar includes 'Ask Gordon BETA', 'Containers' (selected), 'Images', 'Volumes', 'Kubernetes', 'Builds', 'Models', 'MCP Toolkit BETA', 'Docker Hub', 'Docker Scout', and 'Extensions'. The main area displays a 'Containers / rabbitmq' list with one item: 'rabbitmq 85b7ec64e87 1562/15672 5672/5672'. The container status is 'Running (23 hours ago)'. Below the container details, there are tabs for Logs, Inspect, Bind mounts, Exec, Files, and Stats. The Logs tab shows several log entries from the container, such as:

```
2025-12-07 07:06:26 630003400:00 [warning] <0.2196.0> client unexpectedly closed TCP connection
2025-12-07 07:07:21 60854400:00 [info] <0.2196.0> accepting AMQP connection <0.2196.0> (<172.19.0.7:46814 -> 172.19.0.5:5672)
2025-12-07 07:07:22 320297:00:00 [info] <0.2196.0> connection <0.2196.0> (<172.19.0.7:46814 -> 172.19.0.5:5672) has a client-provided name: rabbitConnectionFactory#29d563bd:0
2025-12-07 07:07:22.414028:00:00 [info] <0.2196.0> connection <0.2196.0> (<172.19.0.7:46814 -> 172.19.0.5:5672 - rabbitConnectionFactory#29d563bd:0) user 'guest' authenticated and granted access to vhost '/'
2025-12-07 07:24 07:079347:00:00 [warning] <0.2196.0> closing AMQP connection <0.2196.0> (<172.19.0.7:46814 -> 172.19.0.5:5672 - rabbitConnectionFactory#29d563bd:0, vhost: '/', user: 'guest')
2025-12-07 07:24 07:0879347:00:00 [warning] <0.2196.0> client unexpectedly closed TCP connection
2025-12-07 07:26:42 981083:00:00 [info] <0.2458.0> accepting AMQP connection <0.2458.0> (<172.19.0.7:58892 -> 172.19.0.5:5672)
2025-12-07 07:26:43 298546:00:00 [info] <0.2458.0> connection <0.2458.0> (<172.19.0.7:58892 -> 172.19.0.5:5672) has a client-provided name: rabbitConnectionFactory#22dc9d46:0
2025-12-07 07:26:43 344423:00:00 [info] <0.2458.0> connection <0.2458.0> (<172.19.0.7:58892 -> 172.19.0.5:5672 - rabbitConnectionFactory#22dc9d46:0) user 'guest' authenticated and granted access to vhost '/'
2025-12-07 15:26:36.190475:00:00 [warning] <0.2458.0> closing AMQP connection <0.2458.0> (<172.19.0.7:58892 -> 172.19.0.5:5672 - rabbitConnectionFactory#22dc9d46:0, vhost: '/', user: 'guest')
2025-12-07 15:26:36.190475:00:00 [warning] <0.2458.0> client unexpectedly closed TCP connection
2025-12-07 15:29:38 303612:00:00 [info] <0.13218.0> accepting AMQP connection <0.13218.0> (<172.19.0.8:50522 -> 172.19.0.5:5672)
2025-12-07 15:29:38 517457:00:00 [info] <0.13218.0> connection <0.13218.0> (<172.19.0.8:50522 -> 172.19.0.5:5672) has a client-provided name: rabbitConnectionFactory#6a7ea7c:0
2025-12-07 15:29:38.561617:00:00 [info] <0.13218.0> connection <0.13218.0> (<172.19.0.8:50522 -> 172.19.0.5:5672 - rabbitConnectionFactory#6a7ea7c:0) user 'guest' authenticated and granted access to vhost '/'
2025-12-07 18:08:24 475789:00:00 [warning] <0.13218.0> closing AMQP connection <0.13218.0> (<172.19.0.8:50522 -> 172.19.0.5:5672 - rabbitConnectionFactory#6a7ea7c:0, vhost: '/', user: 'guest')
2025-12-07 18:08:24 475789:00:00 [warning] <0.13218.0> client unexpectedly closed TCP connection
2025-12-07 18:23:59 274857:00:00 [info] <0.17222.0> accepting AMQP connection <0.17222.0> (<172.19.0.8:51264 -> 172.19.0.5:5672)
2025-12-07 18:23:59 580867:00:00 [info] <0.17222.0> connection <0.17222.0> (<172.19.0.8:51264 -> 172.19.0.5:5672) has a client-provided name: rabbitConnectionFactory#46a0ef6f:0
2025-12-07 18:23:59.622377:00:00 [info] <0.17222.0> connection <0.17222.0> (<172.19.0.8:51264 -> 172.19.0.5:5672 - rabbitConnectionFactory#46a0ef6f:0) user 'guest' authenticated and granted access to vhost '/'
2025-12-08 04:02:22 973671:00:00 [warning] <0.25948.0> HTTP access denied: user 'admin' - invalid credentials
2025-12-08 04:02:28 2608544:00:00 [warning] <0.25955.0> HTTP access denied: user 'admin' - invalid credentials
2025-12-08 04:02:33.750186:00:00 [warning] <0.25959.0> HTTP access denied: user 'admin' - invalid credentials
```

At the bottom, system status shows 'Engine running' and resource usage: RAM 7.10 GB, CPU 41.79%, Disk 19.00 GB used (limit 1006 85 GB). On the right, there's a terminal window and a status bar with icons for Ctrl+K, search, notifications, and other system status.

Mongodb logs

The screenshot shows the Docker Desktop interface. On the left, a sidebar lists various services: Ask Gordon (Beta), Containers (selected), Images, Volumes, Kubernetes, Builds, Models, MCP Toolkit (Beta), Docker Hub, Docker Scout, and Extensions. The main area displays a single container named 'booking-mongodb'. The container's status is 'Running (23 hours ago)'. It has a green icon and a URL '8f3f1e0917eb'. The container's logs show several entries, all starting with '["remote": {"\$date": "2025-12-08T04:55:32.262+00:00"}]'. One entry includes the message 'msg": "Connection ended", "attr": {"remote": {"\$date": "127.0.0.1:41986", "isLoadBalanced": false, "uid": {"\$oid": "655086fc-ab44-4fa4-9f57-aa9b9e6fa875"}}, "connectionId": 35172, "connectionCount": 7}}'. Another entry shows 'msg": "Connection accepted", "attr": {"remote": {"\$date": "127.0.0.1:41986", "isLoadBalanced": false, "uid": {"\$oid": "655086fc-ab44-4fa4-9f57-aa9b9e6fa875"}}, "connectionId": 35171, "connectionCount": 6}}'. The logs also mention 'client metadata' and 'driver' information. The container's configuration includes 'os': 'Node.js v20.19.5', 'cpu': 'RAM 9.9 GB CPU 4.42%', 'memory': 'Disk 19.0 GB used / 1006.85 GB', and 'platform': 'Node.js v20.19.5, LE'. The bottom right corner shows a terminal window with the text 'v4.53.0'.

5.Mongodb screenshots

- Booking db**

The screenshot shows the MongoDB Compass interface for the `booking_db` database. The left sidebar lists connections and databases, with `booking_db` selected. Under `booking_db`, the `booking` collection is highlighted. The main pane displays two documents in the `booking` collection:

```
_id: ObjectId('692d8254fd1536048c1d1b37')
pnr : "PNR252A0809"
flightId : "692d822651cef85aec2564cb"
flightNumber : "AI101"
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.flightapp.booking.entity.Booking"

_id: ObjectId('692d8280fd1536048c1d1b38')
pnr : "PREBFDS45C"
flightId : "692d822651cef85aec2564cb"
flightNumber : "AI101"
airline : "Air India"
fromPlace : "Delhi"
```

- Flight db**

The screenshot shows the MongoDB Compass interface for the `flight_db` database. The left sidebar lists connections and databases, with `flight_db` selected. Under `flight_db`, the `flight_inventory` collection is highlighted. The main pane displays one document in the `flight_inventory` collection:

```
_id: ObjectId('692d822651cef85aec2564cb')
airline : "Air India"
flightNumber : "AI101"
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.flightapp.flight.entity.FlightInventory"
```

- Notification logs

```

MongoDB Compass - localhost:27017/flightdbwebflux.notification_log
Connections Edit View Collection Help
Compass
My Queries
Data Modeling
CONNECTIONS (4)
localhost:27017
  admin
  booking_db
    booking
    config
  config
  flight_db
    flight_inventory
  flightdbwebflux
    booking
    flight_inventory
    notification_log
    passenger
  library
  local
  quizdb
  test
  localhost:27017
  localhost:27017

localhost:27017 > flightdbwebflux > notification_log
Documents 3 Aggregations Schema Indexes 1 Validation
Type a query: { field: 'value' } or Generate query
ADD DATA EXPORT DATA UPDATE DELETE
status : "SUCCESS"
details : "Email sent"
createdAt : 2023-11-29T19:16:22.289+00:00
_class : "com.flighthapp.entity.NotificationLog"

_id: ObjectId('692d8533e195fb66d11e1466')
passengerId: "692d85c4e03f4e58590436"
email: "riddhimabhanja2003@gmail.com"
subject: "Your e-ticket and booking details"
type: "EMAIL"
status: "SUCCESS"
details: "Email sent"
createdAt: 2023-12-01T12:08:19.565+00:00
_class: "com.flighthapp.entity.NotificationLog"

_id: ObjectId('693880f63a1c8a7745cc4c3')
passengerId: "692d85c4e03f4e58590436"
email: "riddhimabhanja2003@gmail.com"
subject: "Your e-ticket and booking details"
type: "TICKET"
status: "SUCCESS"
details: "Email sent"
createdAt: 2023-12-03T18:27:02.142+00:00
_class: "com.flighthapp.entity.NotificationLog"

```

5. Email

- Welcome Email

mail.google.com/mail/u/0/?tab=rm&oqlb#inbox/FFMfcgzQdzcdBBWCgDpKlmXdpkcXIMQX

Welcome Test

riddhimabhanja2003@gmail.com
to me • Sun, Dec 7, 11:55 PM (10 hours ago)

Welcome Aboard!

Dear Test User,

Welcome to our Flight Booking Service! We're excited to have you with us.

Your Customer ID: CUST789

With our service, you can:

- Search and book flights to destinations worldwide
- Manage your bookings easily
- Receive instant email confirmations
- Access 24/7 customer support

Start exploring our flight options and book your next journey today!

Best regards,
Flight Booking Team

© 2025 Flight Booking Service. All rights reserved.

- Confirm Booking Email

The screenshot shows a Gmail inbox with 95 messages. A specific email from "riddhimabhanja2003@gmail.com" is selected, titled "Flight Booking Confirmation - PNR12345678". The subject line is "Flight Booking Confirmation". The body of the email starts with "Flight Booking Confirmed!" in a green header. It then addresses the recipient with "Dear riddhima bhanja," and states "Your flight booking has been successfully confirmed. Below are your booking details:". It includes a "Booking Information" section with the following details:

- PNR: PNR12345678
- Flight Number: AI101
- Number of Seats: 2
- Total Amount: Rs. 10000
- Booking Date: 2025-12-07

Below this, there is an "Important Notes:" section with three bullet points:

- Please arrive at the airport at least 2 hours before departure
- Carry a valid photo ID for check-in
- Check-in opens 2 hours before departure

The email concludes with "Thank you for choosing our service!", "Best regards," and "Flight Booking Team".

Flight cancel email

The screenshot shows a Gmail inbox with 96 messages. A specific email from "riddhimabhanja2003@gmail.com" is selected, titled "Flight Booking Cancelled". The subject line is "Flight Booking Cancelled". The body of the email starts with "Dear John Doe," and states "Your flight booking has been cancelled." It includes a "Booking Details" section with the following information:

- PNR: PNR45010076
- Flight: AI101
- Route: Delhi to Mumbai
- Departure: 2024-12-15T10:00
- Total Amount: \$10000.0

Below this, there is a "Thank you for choosing our service!" message, "Best regards," and "Flight Booking Team".

RabbitMQ email-queue dashboard

This screenshot shows the RabbitMQ Queue 'email.queue' dashboard. At the top, there are navigation links for Overview, Connections, Channels, Exchanges, Queues and Streams, and Admin. The 'Queues and Streams' tab is selected. Below the tabs, there are two main sections: 'Queued messages last minute' and 'Message rates last minute'. The 'Queued messages last minute' section shows a chart with three bars: Ready (yellow), Unacked (light blue), and Total (red). The 'Message rates last minute' section shows a chart with several bars representing different message types: Publish (yellow), Deliver (auto ack) (light blue), Consumer ack (green), Redelivered (purple), Get (auto ack) (light green), Get (empty) (light blue), Deliver (manual ack) (red), and Get (empty) (light blue). Below these charts, there is a table titled 'Details' with columns for Feature, durable, true, State, and various message counts. A sidebar on the left lists Consumers (1), Bindings (2), Publish message, Get message, Move messages, Delete, and Purge. At the bottom, there are links for HTTP API, Documentation, Tutorials, New releases, Commercial edition, Commercial support, Discussions, Discord, Slack, Plugins, and GitHub.

RabbitMQ overview dashboard

This screenshot shows the RabbitMQ overview dashboard. At the top, there are navigation links for Overview, Connections, Channels, Exchanges, Queues and Streams, and Admin. The 'Overview' tab is selected. Below the tabs, there is a section titled 'Global counts' with a table showing the count of various entities: Dimensions (1), Channels (2), Exchanges (8), Queues (1), and Consumers (1). There is also a 'Nodes' section with a table showing system statistics for a single node: Name (rabbit@53.7.142.47), File descriptors (26 / 104775 available), Socket descriptors (1 / 104775 available), Erlang processes (420 / 100000 available), Memory (84 MB / 3.2 GB high watermark), Disk space (937.68 / 40.0 GB free), Uptime (22h 2m), Info (basic, doc, 2, msi), Reset stats, and a link to 'This node' or 'All nodes'. At the bottom, there are links for HTTP API, Documentation, Tutorials, New releases, Commercial edition, Commercial support, Discussions, Discord, Slack, Plugins, and GitHub.

6. Postman screenshots

Obtain JWT TOKEN

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Endpoint:** http://localhost:8080/api/auth/login (POST method)
- Body (JSON):**

```
1 {
2   "username": "admin",
3   "password": "password"
4 }
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbkiIiMlhdCI6MTc2NTEzNDM5MiwiZXhwIjoxNzY1MjIwNzkyfQ.8FY9Cw6dLfuUb0uQ00zZvk8LuhsCdUhvJhm1k3FRVxrtor66TBWEAM4nWso4tx01V0zhKnVTQUFuTbk1_87w",
3   "username": "admin",
4   "message": "Login successful"
5 }
```

Unauthorized access

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Endpoint:** http://localhost:8080/api/flights/search (GET method)
- Body (JSON):**

```
1 {
2   "origin": "DEL",
3   "destination": "BOM",
4   "travelDate": "2025-12-15"
5 }
```
- Response Status:** 401 Unauthorized
- Response Body (Raw):**

```
1
```

Same endpoint with auth bearer token

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request URL:** `http://localhost:8080/api/flights/search`
- Method:** POST
- Authorization:** Bearer Token (selected in the dropdown)
- Body:** JSON (empty)
- Response:** 200 OK (6.05 s, 736 B)
The response body is a JSON array of flight details:

```
[{"id": 1, "flightNumber": "AI101", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T08:00:00", "arrivalTime": "2025-12-15T10:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 2, "flightNumber": "AI102", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T09:00:00", "arrivalTime": "2025-12-15T11:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 3, "flightNumber": "AI103", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T10:00:00", "arrivalTime": "2025-12-15T12:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 4, "flightNumber": "AI104", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T11:00:00", "arrivalTime": "2025-12-15T13:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 5, "flightNumber": "AI105", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T12:00:00", "arrivalTime": "2025-12-15T14:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 6, "flightNumber": "AI106", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T13:00:00", "arrivalTime": "2025-12-15T15:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 7, "flightNumber": "AI107", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T14:00:00", "arrivalTime": "2025-12-15T16:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 8, "flightNumber": "AI108", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T15:00:00", "arrivalTime": "2025-12-15T17:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 9, "flightNumber": "AI109", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T16:00:00", "arrivalTime": "2025-12-15T18:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 10, "flightNumber": "AI110", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T17:00:00", "arrivalTime": "2025-12-15T19:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 11, "flightNumber": "AI111", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T18:00:00", "arrivalTime": "2025-12-15T20:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 12, "flightNumber": "AI112", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T19:00:00", "arrivalTime": "2025-12-15T21:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}, {"id": 13, "flightNumber": "AI113", "airline": "Air India", "origin": "DEL", "destination": "BOM", "departureTime": "2025-12-15T20:00:00", "arrivalTime": "2025-12-15T22:30:00", "availableSeats": 134, "price": 5000, "status": "ACTIVE", "createdAt": "2025-12-07T06:02:42"}]
```

Circuit Breaker

Events:

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request URL:** `http://localhost:8080/actuator/circuitbreakerevents`
- Method:** GET
- Authorization:** None
- Body:** JSON (empty)
- Response:** 200 OK (95 ms, 1.4 KB)
The response body is a JSON object containing circuit breaker events:

```
{ "circuitBreakerEvents": [ { "circuitBreakerName": "bookingServiceCircuitBreaker", "type": "ERROR", "creationTime": "2025-12-01T21:55:37.209614500+05:30[Asia/Calcutta]", "errorMessage": "java.util.concurrent.TimeoutException: Did not observe any item or terminal signal within 1000ms in 'circuitBreaker' (and no fallback has been configured)" }, { "durationInMs": 1000, "stateTransition": null } ], { "circuitBreakerName": "flightServiceCircuitBreaker", "type": "ERROR", "creationTime": "2025-12-01T21:55:45.179704400+05:30[Asia/Calcutta]", "errorMessage": "java.util.concurrent.TimeoutException: Did not observe any item or terminal signal within 1000ms in 'circuitBreaker' (and no fallback has been configured)" } ] }
```

Status:

The screenshot shows the Postman interface with a workspace titled "riddhima bhanja's Workspace". On the left sidebar, under "Collections", there is a section for "circuit breaker" which contains "GET status" and "GET events". The "GET status" item is selected and highlighted in orange. The main panel displays a GET request to "http://localhost:8080/actuator/circuitbreakers". The response status is 200 OK, with a response time of 62 ms and a body size of 618 B. The response body is a JSON object representing the state of the "FlightServiceCircuitBreaker":

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

• Health Check

API gateway health:

The screenshot shows the Postman interface with a workspace titled "riddhima bhanja's Workspace". On the left sidebar, under "Collections", there is a section for "Flight Booking Microservices" which contains "POST Add Flight Inventory", "POST Search Flights", "POST Book Flight", "GET Get Booking by PNR", "GET Get Booking History", "DEL Cancel Booking", and "GET Eureka Server". Below this, there is a section for "Health Checks" which contains "GET API Gateway Health", "GET Flight Service Health", and "GET Booking Service Health". The "GET API Gateway Health" item is selected and highlighted in orange. The main panel displays a GET request to "http://localhost:8080/actuator/health". The response status is 200 OK, with a response time of 14 ms and a body size of 991 B. The response body is a JSON object representing the overall system status: "UP". It also includes a detailed breakdown of components: "discoveryComposite" (status: UP), "discoveryClient" (status: UP), and "services" (listing "api-gateway", "flight-service", and "booking-service").

```
1 {  
2   "status": "UP",  
3   "components": [  
4     "discoveryComposite": {  
5       "status": "UP",  
6       "components": [  
7         "discoveryClient": {  
8           "status": "UP",  
9           "details": {  
10             "services": [  
11               "api-gateway",  
12               "flight-service",  
13               "booking-service"  
14             ]  
15           ]  
16         }  
17       }  
18     }  
19   ]  
20 }  
21
```

Booking Service Health:

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Request:** GET http://localhost:8082/actuator/health
- Status:** 200 OK
- Body (JSON):**

```
1 {  
2   "status": "UP",  
3   "components": {  
4     "circuitBreakers": {  
5       "status": "UP",  
6       "details": {  
7         "flightService": {  
8           "status": "UP",  
9           "details": {  
10              "failureRate": "-1.0%",  
11              "failureRateThreshold": "50.0%",  
12              "slowCallRate": "-1.0%",  
13              "slowCallRateThreshold": "100.0%",  
14              "bufferedCalls": 0,  
15              "slowCalls": 0,  
16            }  
17          }  
18        }  
19      }  
20    }  
21 }
```

Eureka server:

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Request:** GET http://localhost:8761/
- Status:** 200 OK
- Body (HTML):**

```
1 <!doctype html>  
2 <html lang="en">  
3 <head>  
4   <base href="/">  
5   <meta charset="utf-8">  
6   <title>Eureka</title>  
7   <meta name="description" content="">  
8   <meta name="viewport" content="width=device-width">  
9   <link rel="stylesheet" href="eureka/css/wro.css">  
10 </head>  
11 <body>  
12   <h1>Eureka</h1>  
13   <p>Welcome to the Eureka Server</p>  
14 </body>  
15 </html>
```

Flight Service Health:

The screenshot shows the Postman interface with a successful response from the Flight Service Health endpoint. The response body is a JSON object indicating all components are up.

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

ADD FLIGHT

The screenshot shows the Postman interface with a successful POST request to add a flight inventory entry. The response body is a JSON object with flight details and a status message.

```
{
  "airline": "Air India",
  "flightNumber": "AI101",
  "fromPlace": "Delhi",
  "toPlace": "Mumbai",
  "departureDateTime": "2024-12-15T10:00:00",
  "arrivalDateTime": "2024-12-15T12:30:00",
  "totalSeats": 180,
  "availableSeats": 180,
  "ticketPrice": 5000.0,
  "flightStatus": "ACTIVE",
  "oneWayPrice": 5000.0,
  "roundTripPrice": 9000.0,
  "mealAvailable": true
}
```

BOOK FLIGHT

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Collection:** Flight Booking Microservices
- Request:** POST /api/v1/booking/book/:flightId
- Method:** POST
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2   "pn": "PNR45018076",
3   "flightId": "692d822651cef85aec2564cb",
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"
```

SEARCH FLIGHTS

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Collection:** Flight Booking Microservices
- Request:** POST /api/v1/flight/search
- Method:** POST
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 [
2   {
3     "id": "692d822651cef85aec2564cb",
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,
16    ...
17  }
18]
```

CANCEL BOOKING

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Request Type:** DELETE
- URL:** http://localhost:8080/api/v1/booking/cancel/:pnr
- Headers:** (8)
Key Value
Content-Type application/json
Authorization Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adGwuDw
- Body:** ({} JSON)
A JSON object representing a booking cancellation request with fields: pnr, flightId, flightNumber, airline, fromPlace, toPlace, departureDateTime, arrivalDateTime, userName, userEmail, journeyDate, noOfSeats, mealType, and totalAmount.
- Response:** 200 OK (201 ms, 712 B)

GET BOOKING BY PNR

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Request Type:** GET
- URL:** http://localhost:8080/api/v1/booking/:pnr
- Headers:** (7)
Key Value
Content-Type application/json
Authorization Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adGwuDw
- Path Variables:**

| Key | Value | Description |
|-----|-------------|-------------|
| pnr | PNR252ABD90 | Description |
- Body:** ({} JSON)
A JSON object representing a booking with fields: pnr, flightId, flightNumber, airline, fromPlace, toPlace, departureDateTime, arrivalDateTime, and userName.
- Response:** 200 OK (27 ms, 713 B)

GET BOOKING HISTORY

The screenshot shows the Postman interface with a collection named "riddhima bhanja's Workspace". The "Flight Booking Microservices" collection is expanded, and the "Get Booking History" endpoint is selected. The request URL is `http://localhost:8080/api/v1/booking/history/:email`. The response status is 200 OK, and the JSON body contains the following data:

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

FALLBACK case

The screenshot shows the Postman interface with a collection named "riddhima bhanja's Workspace". The "BookingService API(docker)" collection is expanded, and the "fallback(circuit breaker)" endpoint is selected. The request URL is `http://localhost:8080/api/bookings/book`. The response status is 201 Created, and the JSON body contains the following error message:

```
1  {
2    "pnr": null,
3    "flightNumber": null,
4    "passengerName": null,
5    "numberOfSeats": null,
6    "totalAmount": null,
7    "status": "FAILED",
8    "bookingDate": null,
9    "message": "Service temporarily unavailable. Please try again later."
10 }
```

SEND EMAIL

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request Method:** POST
- Request URL:** <http://localhost:8083/api/notifications/send>
- Body (raw JSON):**

```
6 "subject": "Welcome Test",
7   "templatedData": {
8     "customerName": "Test User",
9     "customerId": "CUST789"
10   }
11 }
```
- Response Status:** 201 Created
- Response Body (JSON):**

```
1 {
2   "notificationId": "6935c6a955627f03b8b65198",
3   "customerId": "CUST789",
4   "customerEmail": "riddhimabhanja2003@gmail.com",
5   "status": "SENT",
6   "message": "Email sent successfully",
7   "timestamp": "2025-12-07T18:25:49.718726912"
8 }
```

GET BY ID

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request Method:** GET
- Request URL:** <http://localhost:8083/api/notifications/6935a9a61ee8f9036919f375>
- Query Params:**

| Key | Value | Description |
|-----|-------|-------------|
| Key | Value | Description |
- Response Status:** 200 OK
- Response Body (HTML Preview):**

```
1 <li>\n    <li>Access 24/7 customer support</li>\n    </ul>\n    <p>Start\nexploring our flight options and book your next journey today!</p>\n    <p>Best regards,\n<br>\n    <strong>Flight Booking Team</strong></p>\n    </div>\n    <div\n        class='\\'footer'\\'>\n        <p>&copy; 2025 Flight Booking Service. All rights reserved.</p>\n    </div>\n</div>\n</body>\n</html>\n",\n\n"status": "SENT",\n"errorMessage": null,\n"createdAt": "2025-12-07T16:21:58.004",\n"sentAt": "2025-12-07T16:21:59.059"
```

BOOKING CONFIRMATION EMAIL

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request Type:** POST
- URL:** http://localhost:8083/api/notifications/send
- Body (raw JSON):**

```
5  "templateName": "booking-confirmation",
6  "subject": "Flight Booking Confirmation - PNR12345678",
7  "templateData": [
8    "customerName": "riddhima bhanja",
9    "pnr": "PNR12345678",
10   "flightNumber": "AI101",
```
- Response:** 201 Created

GET ALL NOTIFICATION

The screenshot shows the Postman interface with the following details:

- Workspace:** riddhima bhanja's Workspace
- Request Type:** GET
- URL:** http://localhost:8083/api/notifications/all
- Query Params:** None
- Response:** 200 OK

The response body is a JSON object containing a single element:

```
28 {
29   "id": "69359bdbcbab6bb290bcb8712",
30   "customerId": "CUST123",
31   "customerName": "riddhima bhanja",
32   "customerEmail": "riddhimbhanja2003@example.com",
33   "templateName": "booking-confirmation",
34   "subject": "Booking Confirmation",
35   "body": "<!DOCTYPE html><html><head></head><meta charset=\"UTF-8\"><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\"><title>Booking Confirmation</title><style><body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; } .header { margin: 0 auto; padding: 20px; } .header h1 { background-color: #4CAF50; color: white; text-align: center; border-radius: 5px; }</style></html>"}
```

GET CUSTOMER NOTIFICATION

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Request Type:** GET
- URL:** <http://localhost:8083/api/notifications/customer/CUST123>
- Body:** JSON response (200 OK)

```
10  "status": "FAILED",
11  "errorMessage": "Failed to send email",
12  "createdAt": "2025-12-07T15:11:04.1",
13  "sentAt": null
14 }
15 {
16   "id": "69359ba1cba6bb290bcb8711",
17   "customerId": "CUST123",
18   "customerName": "riddhima bhanja",
19   "customerEmail": "riddhimabhanja2003@example.com",
20   "templateName": "booking-confirmation",
21   "subject": "Booking Confirmation",
22   "body": "Your booking has been confirmed. Your flight number is AI101. Please check your inbox for the confirmation email."}
```

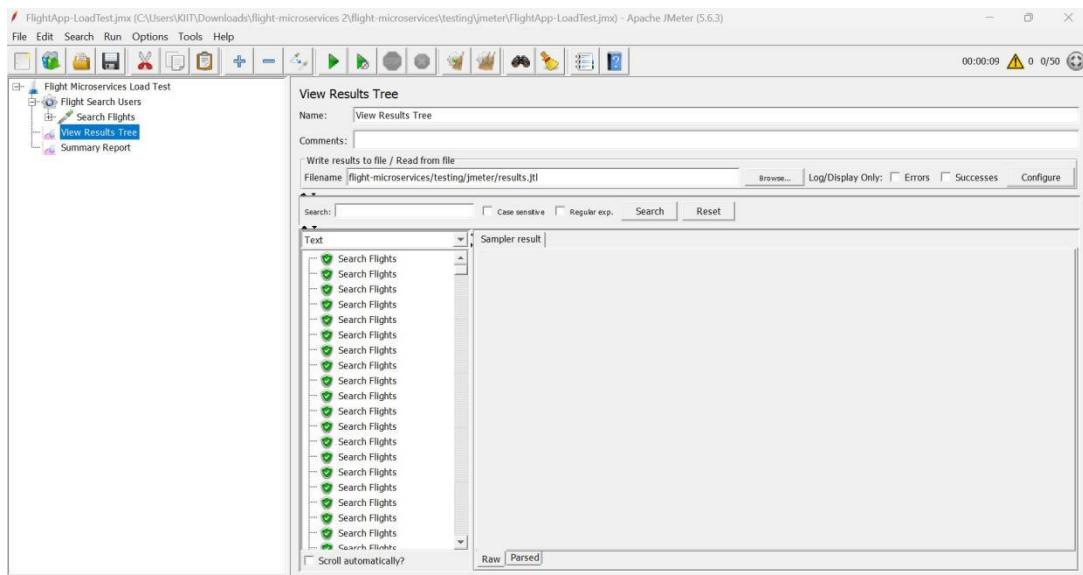
CREATE BOOKING

The screenshot shows the Postman interface with the following details:

- Collection:** riddhima bhanja's Workspace
- Request Type:** POST
- URL:** <http://localhost:8080/api/bookings/book>
- Headers:** Content-Type: application/json
- Body:** JSON response (201 Created)

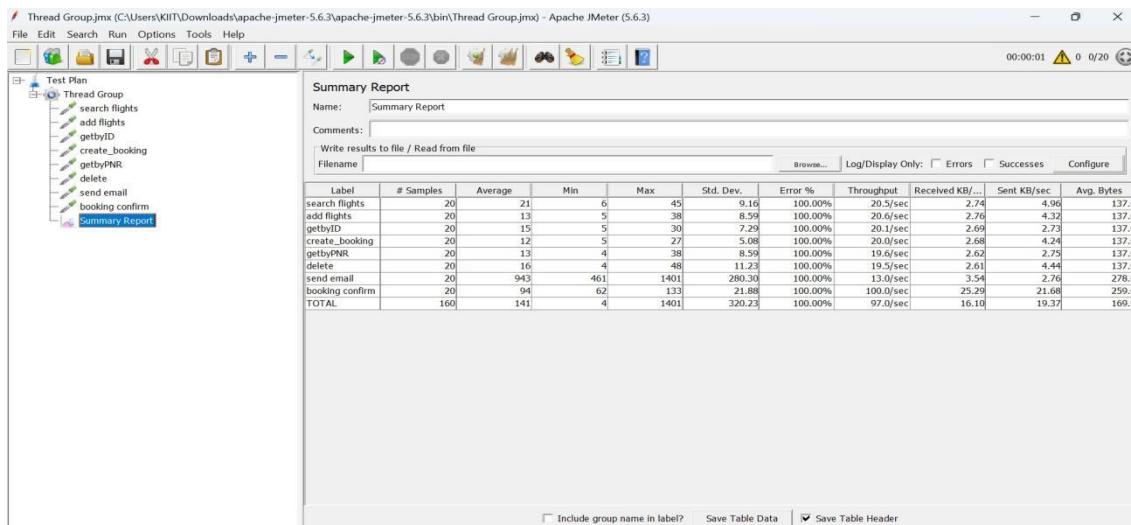
```
1 {
2   "pnr": "PNR7051E600",
3   "flightNumber": "AI101",
4   "passengerName": "riddhima",
5   "numberOfSeats": 2,
6   "totalAmount": 10000.0,
7   "status": "CONFIRMED",
8   "bookingDate": "2025-12-07T11:02:44.541433336",
9   "message": "Booking created successfully"
10 }
```

- Jmeter result tree

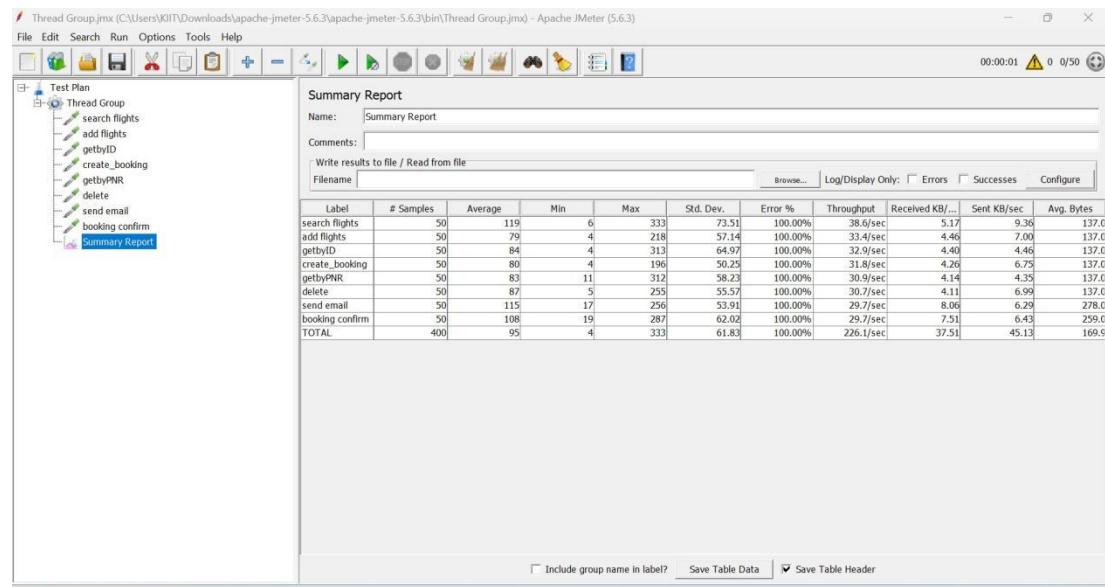


• Jmeter Summary report

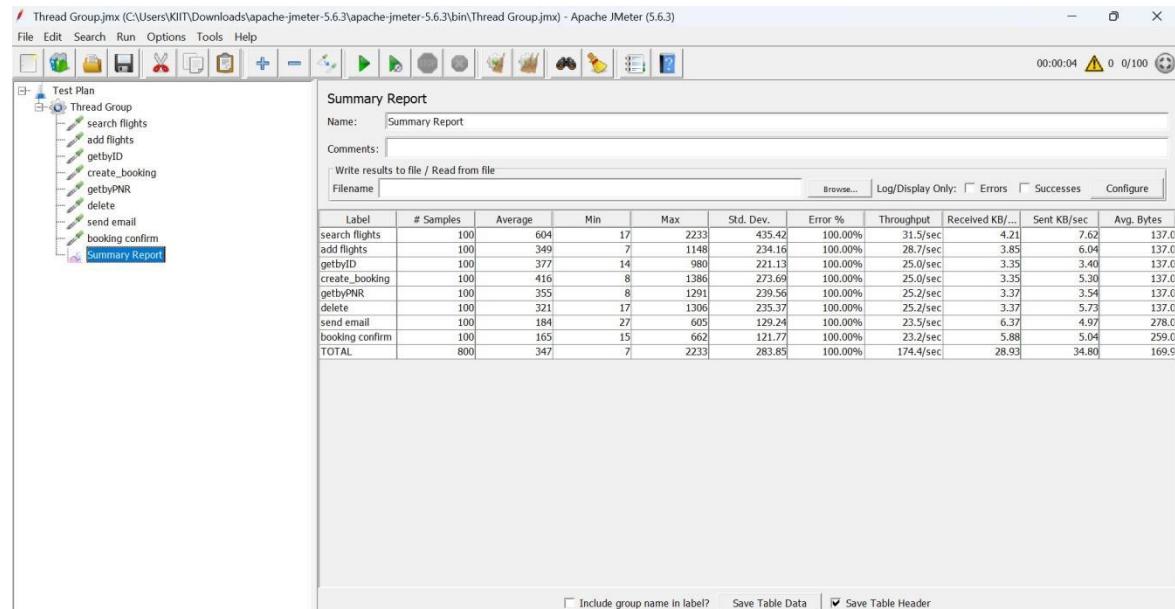
20 Requests



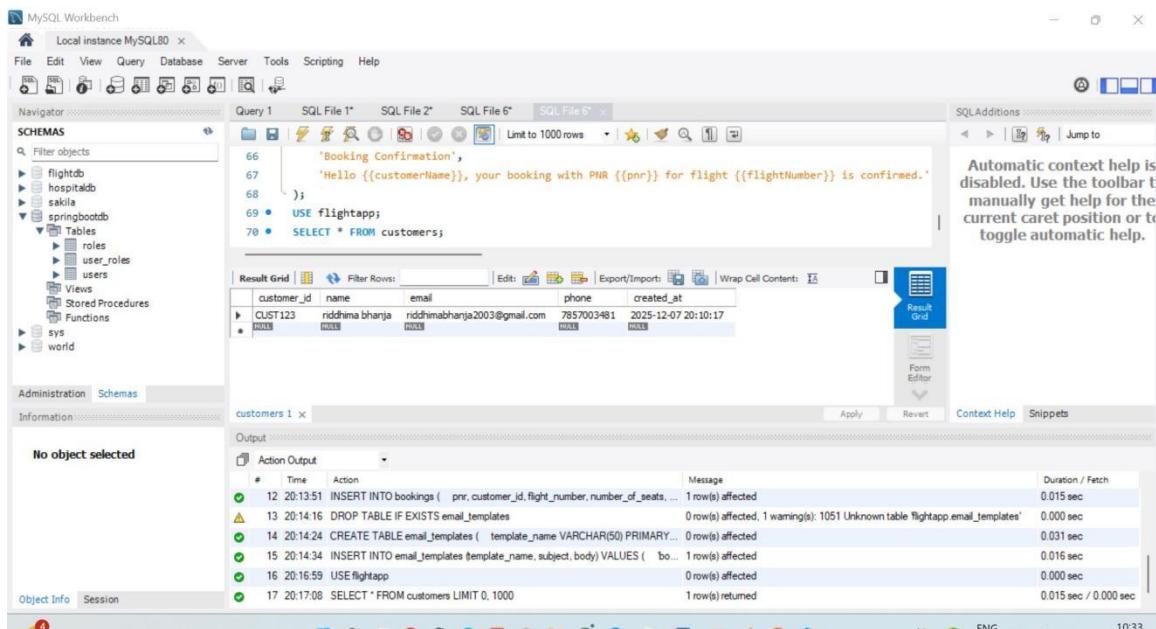
50 Requests



100 Requests



MySQL Workbench



EUREKA DASHBOARD

The screenshot shows the Eureka Dashboard at localhost:8761. The top navigation bar includes links for Home, Last 1000 since startup, Sign in, Chat, and Help.

The main dashboard has the following sections:

- System Status:** Displays environment (test), data center (default), current time (2025-12-07T16:36:21+0000), uptime (00:21), lease expiration enabled (true), renew threshold (0), and renew (last min) (8).
- DS Replicas:** A note states "THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS." Below this, it lists registered instances with their application names, AMIs, availability zones, and statuses:

| Application | AMIs | Availability zones | Status |
|----------------------|---------|--------------------|--|
| API-GATEWAY | n/a (1) | (1) | UP (1) - api-gateway:8080 172.0 ef735671c484c x4e3542cf |
| BOOKING-SERVICE | n/a (1) | (1) | UP (1) - booking-service:8504 4224351192 e2cefebf0 0xa16e443 |
| FLIGHT-SERVICE | n/a (1) | (1) | UP (1) - flight-service:95125 bc520e35391c750420a aba2cd |
| NOTIFICATION-SERVICE | n/a (1) | (1) | UP (1) - 23b5e63ac0sd.notification-service:8083 |
- General Info:** Shows various system metrics like total available memory (87mb), number of CPUs (8), current memory usage (53mb (60%)), and server uptime (00:21). It also lists registered replicas and available replicas.
- Instance Info:** A table showing instance details for each registered service.