

FLIGHT MANAGEMENT SYSTEM

(Webflux + MongoDB)

Debashrita Mandal

JACOCO REPORT:

Code Coverage: 89%

The screenshot shows a browser window with the URL C:/Users/KIIT/Documents/workspace-spring-tools-for-eclipse-4.32.2.RELEASE/FMS_reactive_mongodb/target/site/jacoco/index.html. The page title is "FMS_reactive_mongodb". Below the title is a table of code coverage data:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cqty	Missed	Lines	Missed	Methods	Missed	Classes
com.fms.reactive.service	86%	86%	64%	64%	4	19	15	93	0	12	0	2
com.fms.reactive	94%	94%	100%	100%	1	6	2	17	1	5	0	2
com.fms.reactive.controller	83%	83%	n/a	n/a	1	6	1	6	1	6	0	1
com.fms.reactive.model	100%	100%	n/a	n/a	0	4	0	17	0	4	0	4
Total	63 of 595	89%	5 of 16	68%	6	35	18	133	2	27	0	9

Created with JaCoCo 0.8.11 2023-10-14 08:53

SONARQUBE REPORT

Code Coverage: 84.6%

The screenshot shows a browser window with the URL sonarcloud.io/summary/overall?id=com.chubb%3AFMS_reactive_mongodb&branch=master. The page title is "Debashrita Mandal > FMS_reactive_mongodb > master". Below the title is a navigation bar with "My Projects", "My Issues", and "Explore". On the left, there is a sidebar with project details: "FMS_reactive_mongodb Project", "Private", "Bind project", "Overview", "Main Branch" (selected), "Pull Requests", "Branches", "Information", and "Administration". The main content area is titled "Summary" and contains several cards: "Security" (0 Open issues, A grade), "Reliability" (0 Open issues, A grade), "Maintainability" (17 Open issues, A grade), "Accepted Issues" (0), "Coverage" (84.6%, 84.6% coverage on 133 lines), and "Duplications" (0.0%, 0.0% duplications on 804 lines). There is also a "Security Hotspots" section with 0 results.

SONARQUBE INITIAL MAINTAINABILITY ISSUES:

Total Count: 17

The screenshot shows the SonarQube interface for the project "FMS_reactive_mongodb". The main navigation bar includes "My Projects", "My Issues", and "Explore". The left sidebar shows the project details, including its status as "Private" and "Bind project". The "Main Branch" tab is selected. The "Issues" tab is active, displaying a list of maintainability issues. A filter for "Maintainability" is applied, showing 17 results. The first issue listed is "Remove this unused import 'jakarta.validation.constraints.NotBlank'." It is categorized under "Software quality" and has a severity of "Low". The second issue is "Remove this unused import 'org.springframework.data.mongodb.core.mapping.Document'." It is also under "Software quality" and "Low" severity. The third issue is "Remove this unused import 'com.fms.reactive.model.TripStatus'." This one is under "Severity" and "Medium" severity. The fourth issue is "Remove this useless assignment to local variable 'sourceCity'." It is under "Software quality" and "Medium" severity. The fifth issue is "Remove this unused 'sourceCity' local variable." It is under "Software quality" and "Low" severity.

This screenshot shows the SonarQube interface for the same project, "FMS_reactive_mongodb". The layout is identical to the first screenshot, with the "Main Branch" tab selected. The "Issues" tab is active, showing 17 maintainability issues. The issues listed are identical to those in the first screenshot, including the removal of unused imports and the cleanup of local variable assignments.

SonarQube cloud

My Projects My Issues Explore

Upgrade

F FMS_reactive_mongodb Project

Private Bind project

Overview Main Branch Pull Requests Branches

Information Administration

Debashrita Mandal > FMS_reactive_mongodb > master

Issues Security Hotspots More

Filters Clear All Filters

Software quality 1

- Security 0
- Reliability 0
- Maintainability 17

Add to selection Ctrl + click

Severity 2

- Blocker 0
- High 0
- Medium 3
- Low 11
- Info 3

Remove this useless assignment to local variable "destinationCity". Intentionality

Maintainability Medium cert cwe ... +

Open Debashrita Mandal L30 1min effort 5 hours ago Code Smell Major

Remove this unused "destinationCity" local variable. Intentionality

Maintainability Low java22 unused +

Open Debashrita Mandal L30 5min effort 5 hours ago Code Smell Minor

Replace generic exceptions with specific library exceptions or a custom exception. Intentionality

Maintainability Medium cert cwe ... +

Open Debashrita Mandal L59 20min effort 50 minutes ago Code Smell Major

src.../java/com/fms/reactive/test/BookingControllerTest.java

Remove this unused import 'java.time.LocalDateTime'. Intentionality

Maintainability Low unused +

Open Debashrita Mandal L27 2min effort 1 hour ago Code Smell Info

SonarQube cloud

My Projects My Issues Explore

Upgrade

F FMS_reactive_mongodb Project

Private Bind project

Overview Main Branch Pull Requests Branches

Information Administration

Debashrita Mandal > FMS_reactive_mongodb > master

Issues Security Hotspots More

Filters Clear All Filters

Software quality 1

- Security 0
- Reliability 0
- Maintainability 17

Add to selection Ctrl + click

Severity 2

- Blocker 0
- High 0
- Medium 3
- Low 11
- Info 3

Remove this unused import 'com.fms.reactive.model.Passenger'. Intentionality

Maintainability Low unused +

Open Debashrita Mandal L17 1min effort 1 hour ago Code Smell Minor

Remove this 'public' modifier. Intentionality

Maintainability Info junit tests +

Open Debashrita Mandal L27 2min effort 1 hour ago Code Smell Info

src.../java/com/fms/reactive/test/FlightControllerTest.java

Remove this unused import 'com.fms.reactive.controller.FlightController'. Intentionality

Maintainability Low unused +

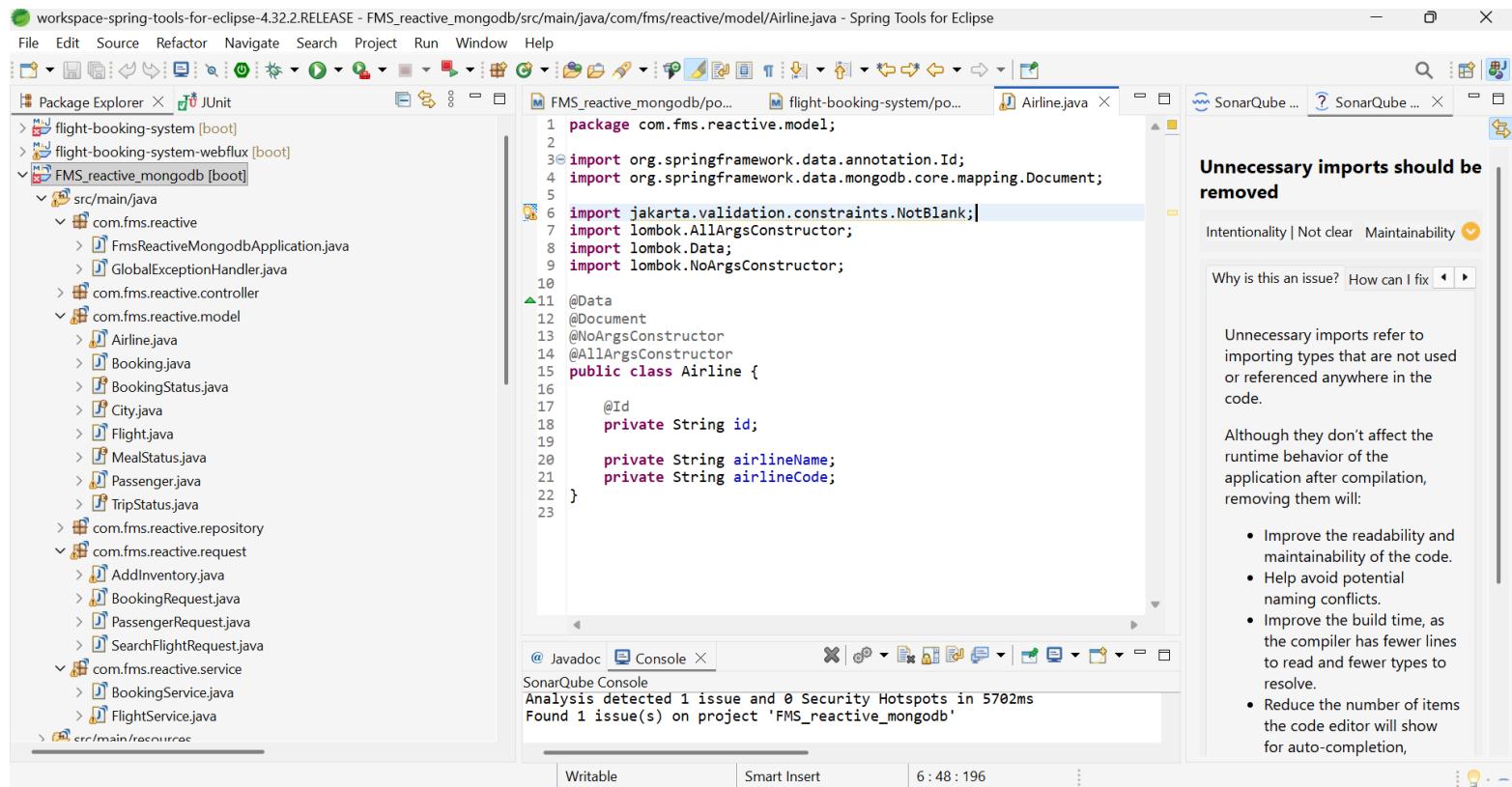
Open Debashrita Mandal L14 1min effort 1 hour ago Code Smell Minor

Remove this 'public' modifier. Intentionality

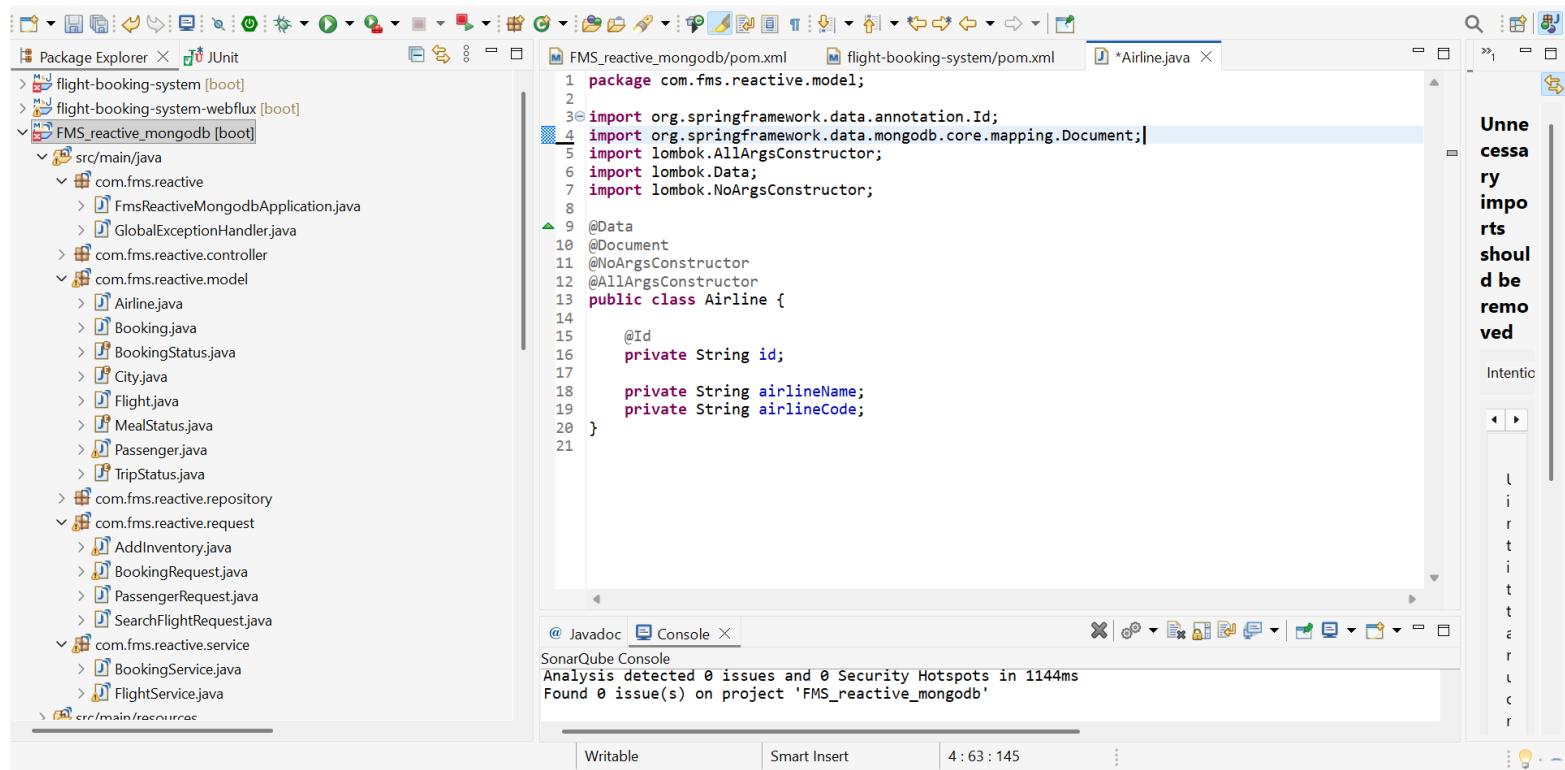
Maintainability Info junit tests +

HANDLING THE ISSUES

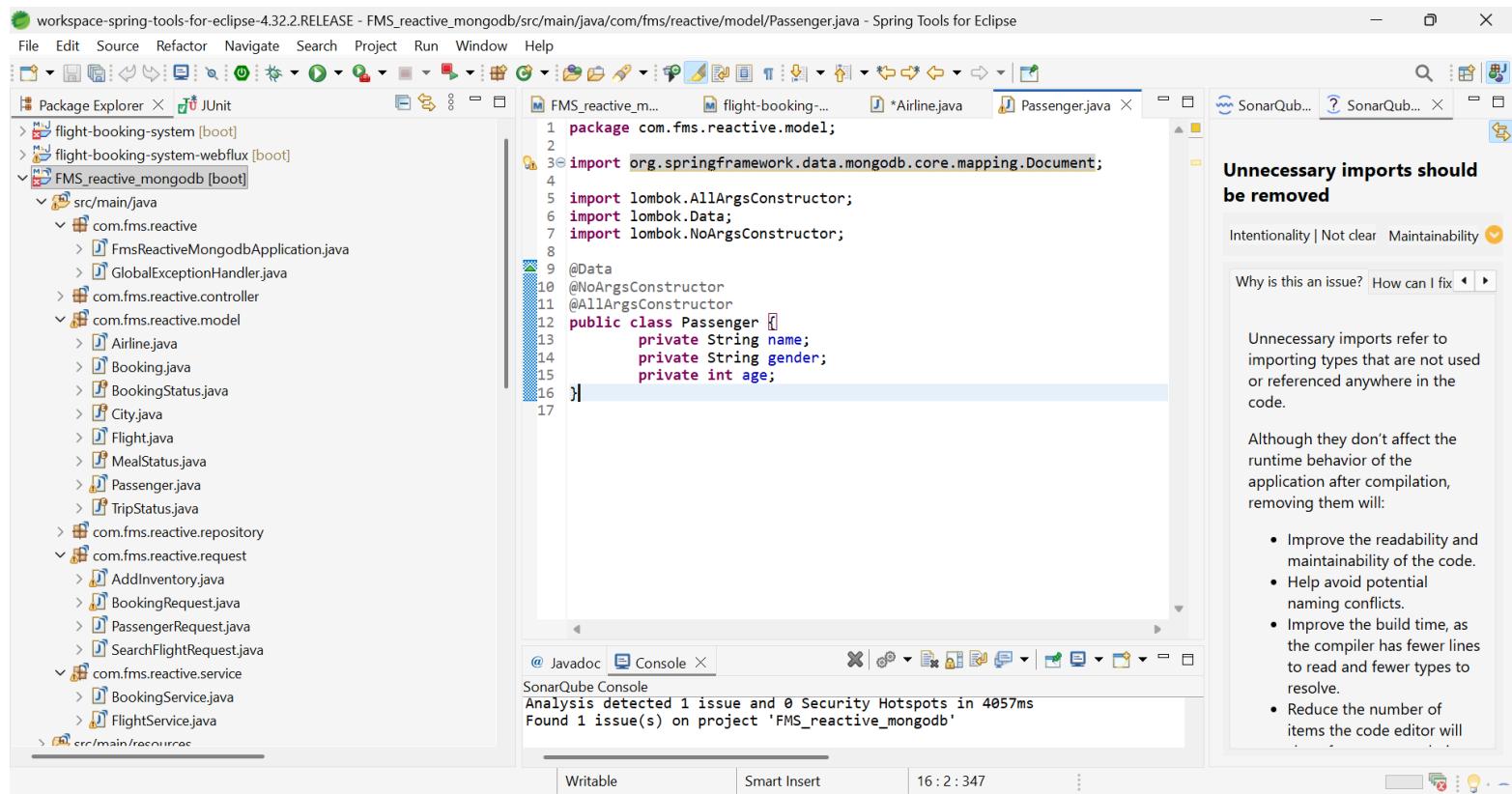
1. Remove this unused import 'jakarta.validation.constraints.NotBlank'.



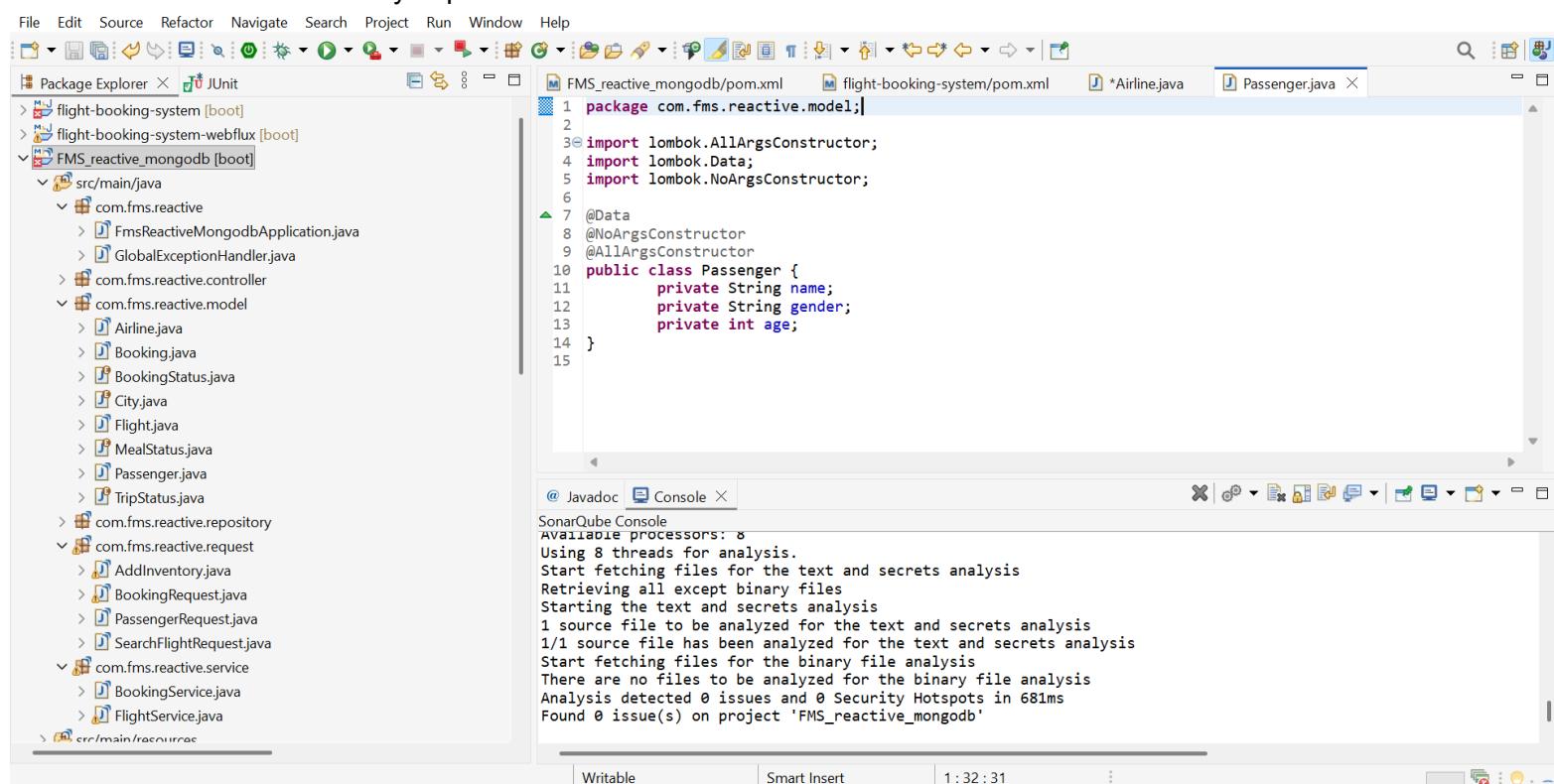
Fix: Removed the unnecessary import



2. Remove this unused import 'org.springframework.data.mongodb.core.mapping.Document'.



Fix: Removed the unnecessary import.



3. Remove this unused import 'com.fms.reactive.model.TripStatus'.

The screenshot shows an IDE interface with a project structure on the left and a code editor on the right. In the code editor, the file `AddInventory.java` is open, showing Java code. A specific import statement, `import com.fms.reactive.model.TripStatus;`, is highlighted in red, indicating it is unnecessary. The SonarQube panel on the right shows a warning titled "Unnecessary imports should be removed" with the following message:

Unnecessary imports refer to importing types that are not used or referenced anywhere in the code.

Although they don't affect the runtime behavior of the application after compilation, removing them will:

- Improve the readability and maintainability of the code.
- Avoid potential naming conflicts.
- Improve the build time, as the compiler has fewer lines to read and fewer types to resolve.
- Reduce the number of items the code editor will

Fix: Removed the unnecessary import.

The screenshot shows the same IDE interface after the unnecessary import was removed. The code editor now shows the corrected code without the `TripStatus` import. The SonarQube panel on the right now shows 1 item, indicating the fix has been applied.

4. Remove this unused import 'com.fms.reactive.model.TripStatus'.

The screenshot shows the Eclipse IDE interface with the SonarQube plugin active. The code editor displays `BookingRequest.java` containing the following code:

```

1 package com.fms.reactive.request;
2
3 import java.util.List;
4
5 import com.fms.reactive.model.MealStatus;
6 import com.fms.reactive.model.TripStatus;
7
8 import jakarta.validation.constraints.Email;
9 import jakarta.validation.constraints.Min;
10 import jakarta.validation.constraints.NotBlank;
11 import jakarta.validation.constraints.NotEmpty;
12 import jakarta.validation.constraints.NotNull;
13 import lombok.AllArgsConstructor;
14 import lombok.Data;
15 import lombok.NoArgsConstructor;
16
17 @Data
18 @NoArgsConstructor
19 @AllArgsConstructor
20 public class BookingRequest {

```

The right-hand margin of the code editor highlights the import statement `import com.fms.reactive.model.TripStatus;` with a red squiggle under the word `TripStatus`, indicating it is unused. A tooltip above the squiggle says "Unnecessary imports should be removed".

The SonarQube console window shows the following output:

```

SonarQube Console
Using 8 threads for analysis.
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
Analysis detected 1 issue and 0 Security Hotspots in 2721ms
Found 1 issue(s) on project 'FMS_reactive_mongodb'

```

Fix: Removed the unnecessary import.

The screenshot shows the Eclipse IDE interface with the SonarQube plugin active. The code editor displays `BookingRequest.java` with the unnecessary import removed:

```

1 package com.fms.reactive.request;
2
3 import java.util.List;
4
5 import com.fms.reactive.model.MealStatus;
6
7 import jakarta.validation.constraints.Email;
8 import jakarta.validation.constraints.Min;
9 import jakarta.validation.constraints.NotBlank;
10 import jakarta.validation.constraints.NotEmpty;
11 import jakarta.validation.constraints.NotNull;
12 import lombok.AllArgsConstructor;
13 import lombok.Data;
14 import lombok.NoArgsConstructor;
15
16 @Data
17 @NoArgsConstructor
18 @AllArgsConstructor
19 public class BookingRequest {
20
21     @NotBlank(message = "User name is required")
22     private String name;
23
24     @Email(message = "Invalid email")

```

The right-hand margin of the code editor now shows a green checkmark icon next to the import statement `import com.fms.reactive.model.TripStatus;`, indicating it is no longer flagged as unused.

The SonarQube console window shows the following output:

```

SonarQube Console
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis

```

5. Remove this useless assignment to local variable "sourceCity".
6. Remove this useless assignment to local variable "destinationCity".
7. Remove this unused "sourceCity" local variable.
8. Remove this unused "destinationCity" local variable.

SonarQube Console output:

```

6 source files to be analyzed for the text and secrets analysis
6/6 source files have been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
Analysis detected 8 issues and 0 Security Hotspots in 11765ms
Found 8 issue(s) on project 'FMS_reactive_mongodb'.

```

Fix: Removed the unused variables.

SonarQube On-The-Fly output:

Date	Description
1 hour ago	
7 hours ago	

SonarQube Console output:

```

Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis

```

9. Replace generic exceptions with specific library exceptions or a custom exception.

The screenshot shows an IDE interface with several windows:

- File Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Package Explorer:** Shows projects: flight-booking-system [boot], flight-booking-system-webflux [boot], FMS_reactive_mongodb [boot].
- Editor:** Displays Java code in `FlightService.java`. A specific line of code is highlighted:

```
50     return flightRepository.save(flight);
51 }
52 }
53 }
54 private City validateCity(String value) {
55     try {
56         return City.valueOf(value.toUpperCase());
57     } catch (IllegalArgumentException ex) {
58         throw new RuntimeException("Invalid city");
59     }
60 }
61 }
```
- SonarQube Console:** Shows the analysis process:

```
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
```
- Right Panel:** A SonarQube rule detail:

Generic exceptions should never be thrown

This rule raises an issue when a generic exception (such as `Error`, `RuntimeException`, `Throwable`, or `Exception`) is thrown.

Why is this an issue? How can I fix ▾

Throwing generic exceptions such as `Error`, `RuntimeException`, `Throwable`, and `Exception` will have a negative impact on any code trying to catch these exceptions.

From a consumer perspective, it is generally a best practice to only catch exceptions you intend to handle. Other exceptions

Fix: Working on the fix, which will require a full restructuring of the Global Error Handling system.

10. Remove this unused import 'java.time.LocalDateTime'.

The import `java.time.LocalDateTime` is never used

```
1 package com.fms.reactive.test;
2
3 import static org.mockito.ArgumentMatchers.any;
4 import static org.mockito.Mockito.when;
5
6 import java.time.LocalDateTime;
7 import java.util.List;
8
9 import org.junit.jupiter.api.Test;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.boot.test.context.SpringBootTest;
12 import org.springframework.boot.test.mock.mockito.MockBean;
13 import org.springframework.test.web.reactive.server.WebTestClient;
14
15 import com.fms.reactive.model.Booking;
16 import com.fms.reactive.model.MealStatus;
17 import com.fms.reactive.model.Passenger;
18 import com.fms.reactive.request.BookingRequest;
19 import com.fms.reactive.request.PassengerRequest;
20 import com.fms.reactive.service.BookingService;
21 import com.fms.reactive.service.FlightService;
22
23 import reactor.core.publisher.Flux;
24 import reactor.core.publisher.Mono;
```

SonarQube Console

```
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
```

Unnecessary imports refer to importing types that are not used or referenced anywhere in the code.

Although they don't affect the runtime behavior of the application after compilation, removing them will:

- Improve the readability and maintainability of the code.
- Help avoid potential naming conflicts.
- Improve the build time, as the compiler has fewer lines to read and fewer types to resolve.
- Reduce the number of items the code editor will

Fix: Removed the unnecessary import.

SonarQube On-The-Fly

Date	Description
2 hours ago	Unused import
2 hours ago	Unused import
2 hours ago	Unused import

Issues reported "on the fly" on files you have recently opened/edited. [Learn more](#)

The import `java.time.LocalDateTime` is never used

```
1 package com.fms.reactive.test;
2
3 import static org.mockito.ArgumentMatchers.any;
4 import static org.mockito.Mockito.when;
5 import java.util.List;
6
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.boot.test.mock.mockito.MockBean;
11 import org.springframework.test.web.reactive.server.WebTestClient;
12
13 import com.fms.reactive.model.Booking;
14 import com.fms.reactive.model.MealStatus;
15 import com.fms.reactive.model.Passenger;
16 import com.fms.reactive.request.BookingRequest;
17 import com.fms.reactive.request.PassengerRequest;
18 import com.fms.reactive.service.BookingService;
19 import com.fms.reactive.service.FlightService;
20
21 import reactor.core.publisher.Flux;
22 import reactor.core.publisher.Mono;
23
24 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
```

SonarQube Console

```
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
```

11. Remove this unused import 'com.fms.reactive.model.Passenger'.

```
1 package com.fms.reactive.test;
2
3@import static org.mockito.ArgumentMatchers.any;
4 import static org.mockito.Mockito.when;
5 import java.util.List;
6
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.boot.test.mock.mockito.MockBean;
11 import org.springframework.test.web.reactive.server.WebTestClient;
12
13 import com.fms.reactive.model.Booking;
14 import com.fms.reactive.model.MealStatus;
15 import com.fms.reactive.model.Passenger;
16 import com.fms.reactive.request.BookingRequest;
17 import com.fms.reactive.request.PassengerRequest;
18 import com.fms.reactive.service.BookingService;
19 import com.fms.reactive.service.FlightService;
20
21 import reactor.core.publisher(Flux;
22 import reactor.core.publisher.Mono;
23
24 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
```

SonarQube Console

```
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
```

Unnecessary imports refer to importing types that are not used or referenced anywhere in the code.

Although they don't affect the runtime behavior of the application after compilation, removing them will:

- Improve the readability and maintainability of the code.
- Help avoid potential naming conflicts.
- Improve the build time, as the compiler has fewer lines to read and fewer types to resolve.
- Reduce the number of items the code editor will

Fix: Removed the unnecessary import.

```
1 package com.fms.reactive.test;
2
3@import static org.mockito.ArgumentMatchers.any;
4 import static org.mockito.Mockito.when;
5 import java.util.List;
6
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.boot.test.mock.mockito.MockBean;
11 import org.springframework.test.web.reactive.server.WebTestClient;
12
13 import com.fms.reactive.model.Booking;
14 import com.fms.reactive.model.MealStatus;
15 import com.fms.reactive.request.BookingRequest;
16 import com.fms.reactive.request.PassengerRequest;
17 import com.fms.reactive.service.BookingService;
18 import com.fms.reactive.service.FlightService;
19
20 import reactor.core.publisher(Flux;
21 import reactor.core.publisher.Mono;
22
23 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
24 public class BookingControllerTest {
```

SonarQube Console

```
Start fetching files for the text and secrets analysis
Retrieving all except binary files
Starting the text and secrets analysis
1 source file to be analyzed for the text and secrets analysis
1/1 source file has been analyzed for the text and secrets analysis
Start fetching files for the binary file analysis
There are no files to be analyzed for the binary file analysis
```

Issues reported "on the fly" on files you have recently opened/edited. [Learn more](#)

12. Remove this 'public' modifier.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like com.fms.reactive.test, com.fms.reactive.repository, com.fms.reactive.request, com.fms.reactive.service, and src/test/java.
- Code Editor:** Displays the file GlobalExceptionHandlerTest.java. The code includes a public class declaration:

```
1 package com.fms.reactive.test;
2
3+import static org.junit.jupiter.api.Assertions.assertEquals;
4
5+public class GlobalExceptionHandlerTest {
```
- SonarQube On-The-Fly:** A panel on the right showing three recent analysis items, all dated "1 hour ago".
- Console:** Shows the SonarQube analysis process: "Start fetching files for the text and secrets analysis", "Retrieving all except binary files", "Starting the text and secrets analysis", "1 source file to be analyzed for the text and secrets analysis", "1/1 source file has been analyzed for the text and secrets analysis", "Start fetching files for the binary file analysis", and "There are no files to be analyzed for the binary file analysis".

Fix: Removed the unnecessary import.

The screenshot shows the Eclipse IDE interface after the fix, with the following changes:

- Project Explorer:** Same as the previous screenshot.
- Code Editor:** The code editor now shows the class definition without the public modifier:

```
1 package com.fms.reactive.test;
2
3+import static org.junit.jupiter.api.Assertions.assertEquals;
4
5+class GlobalExceptionHandlerTest {
```
- SonarQube On-The-Fly:** The panel shows two recent analysis items, both dated "1 hour ago".
- Console:** The analysis results are identical to the previous screenshot, indicating no new issues were found.

Remove this use of "getStatusCodeValue"; it is deprecated.

13. Remove this unused import 'com.fms.reactive.controller.FlightController'.

The screenshot shows the Eclipse IDE interface with the SonarQube plugin. The left sidebar displays the package explorer with various Java files. The main editor window shows the code for `FlightControllerTest.java`. A specific import statement, `import com.fms.reactive.controller.FlightController;`, is highlighted with a blue selection bar. The right-hand panel displays the SonarQube analysis results, specifically the 'Unnecessary imports should be removed' report. It includes a detailed description of unnecessary imports, a list of reasons for removal, and a summary of the impact on readability and maintainability.

```
1 package com.fms.reactive.test;
2
3 import static org.mockito.ArgumentMatchers.any;
4 import static org.mockito.Mockito.when;
5
6 import java.time.LocalDateTime;
7
8 import org.junit.jupiter.api.Test;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.boot.test.context.SpringBootTest;
11 import org.springframework.boot.test.mock.mockito.MockBean;
12 import org.springframework.test.web.reactive.server.WebTestClient;
13
14 import com.fms.reactive.controller.FlightController;
15 import com.fms.reactive.model.Flight;
16 import com.fms.reactive.model.TripStatus;
17 import com.fms.reactive.request.AddInventory;
18 import com.fms.reactive.request.SearchFlightRequest;
19 import com.fms.reactive.service.BookingService;
20 import com.fms.reactive.service.FlightService;
21
22 import reactor.core.publisher(Flux;
23 import reactor.core.publisher.Mono;
```

The import com.fms.reactive.controller.FlightController is never used

Unnecessary imports should be removed

- Improve the readability and maintainability of the code.
- Help avoid potential naming conflicts.
- Improve the build time, as the compiler has fewer lines to read and fewer types to resolve.
- Reduce the number of items the code editor will

Fix: Removed the import.

The screenshot shows the Eclipse IDE interface with the SonarQube plugin. The left sidebar displays the package explorer with various Java files. The main editor window shows the code for `FlightControllerTest.java`. The previously highlighted import statement, `import com.fms.reactive.controller.FlightController;`, is now removed. The right-hand panel displays the SonarQube analysis results, specifically the 'Unnecessary imports should be removed' report. It includes a detailed description of unnecessary imports, a list of reasons for removal, and a summary of the impact on readability and maintainability.

```
1 package com.fms.reactive.test;
2
3 import static org.mockito.ArgumentMatchers.any;
4 import static org.mockito.Mockito.when;
5
6 import java.time.LocalDateTime;
7
8 import org.junit.jupiter.api.Test;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.boot.test.context.SpringBootTest;
11 import org.springframework.boot.test.mock.mockito.MockBean;
12 import org.springframework.test.web.reactive.server.WebTestClient;
13
14 import com.fms.reactive.model.Flight;
15 import com.fms.reactive.model.TripStatus;
16 import com.fms.reactive.request.AddInventory;
17 import com.fms.reactive.request.SearchFlightRequest;
18 import com.fms.reactive.service.BookingService;
19 import com.fms.reactive.service.FlightService;
20
21 import reactor.core.publisher(Flux;
22 import reactor.core.publisher.Mono;
```

Unnecessary imports should be removed

- Improve the readability and maintainability of the code.
- Help avoid potential naming conflicts.
- Improve the build time, as the compiler has fewer lines to read and fewer types to resolve.
- Reduce the number of items the code editor will

14. Remove this 'public' modifier.

The screenshot shows the Eclipse IDE interface with the 'FlightControllerTest.java' file open in the editor. The code is as follows:

```
24 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
25 public class FlightControllerTest {
26
27     @Autowired
28     private WebTestClient client;
29
30     @MockBean
31     private FlightService flightService;
32
33     @MockBean
34     private BookingService bookingService;
35
36     @Test
37     void testAddInventory() {
38
39         AddInventory req = new AddInventory(
40             "A001",
41             "DELHI",
42             "MUMBAI",
43             "2025-12-01T10:00:00",
44             5000,
45             100
46     );
47 }
```

In the bottom right corner, there is a SonarQube inspection window titled 'JUnit5 test classes and methods should have default package visibility'. It states: 'JUnit5 test classes and methods should generally have package visibility. To fix this issue, change their visibility to the default package visibility.' Below this, there is a 'Why is this an issue?' section and a 'How can I fix it?' section.

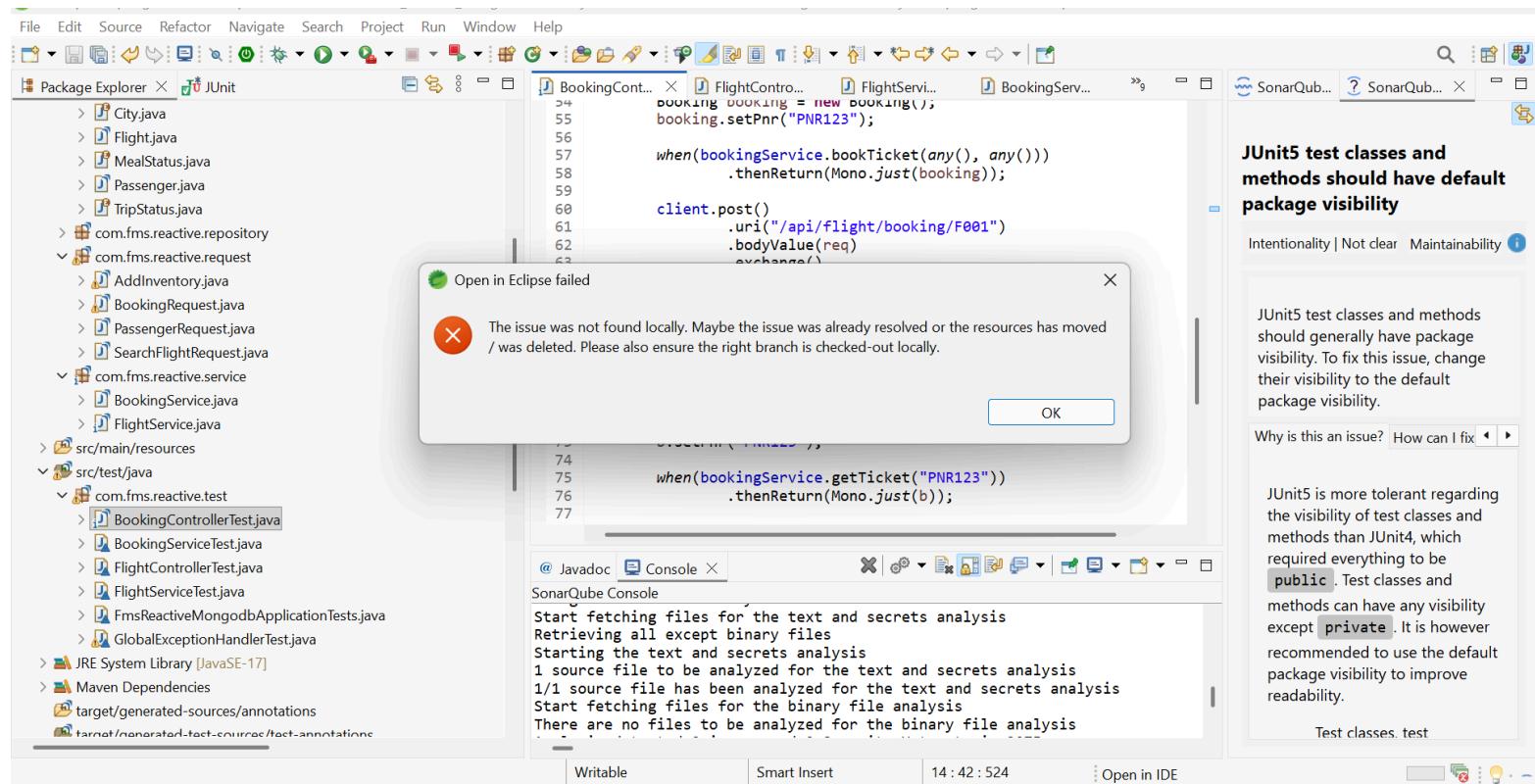
Fix: Removed the public modifier.

The screenshot shows the Eclipse IDE interface with the 'FlightControllerTest.java' file open in the editor. The code has been modified to remove the 'public' modifier from the class definition:

```
22 import reactor.core.publisher.Mono;
23
24 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
25 class FlightControllerTest {
26
27     @Autowired
28     private WebTestClient client;
29
30     @MockBean
31     private FlightService flightService;
32
33     @MockBean
34     private BookingService bookingService;
35
36     @Test
37     void testAddInventory() {
38
39         AddInventory req = new AddInventory(
40             "A001",
41             "DELHI",
42             "MUMBAI",
43             "2025-12-01T10:00:00",
44             5000,
45             100
46     );
47 }
```

The SonarQube inspection window is still present, showing the same message about package visibility. The 'Why is this an issue?' and 'How can I fix it?' sections are also visible.

15. Remove this public modifier



Fix: The issue was not found locally.

16. Remove this use of "getStatusCodeValue"; it is deprecated.

The screenshot shows the Eclipse IDE interface with the SonarQube plugin active. A warning message is displayed in the status bar: "@Deprecated" code should not be used. The code editor shows a test method with a highlighted line of code: `assertEquals(400, response.getStatusCodeValue());`. A tooltip for this line indicates it is deprecated since version 6.0. The SonarQube console output at the bottom right shows the analysis progress: "Start fetching files for the text and secrets analysis", "Retrieving all except binary files", "Starting the text and secrets analysis", "1 source file to be analyzed for the text and secrets analysis", "1/1 source file has been analyzed for the text and secrets analysis", "Start fetching files for the binary file analysis", and "There are no files to be analyzed for the binary file analysis".

Fix: Added @SuppressWarnings("deprecation")

The screenshot shows the Eclipse IDE interface after the fix has been applied. The code editor now includes the annotation `@SuppressWarnings("deprecation")` above the problematic line of code. The SonarQube analysis results remain the same as in the previous screenshot, indicating the code is still being analyzed for deprecated usage.

The method getStatusCodeValue() from the...Object>> is deprecated since version 6.0 | Writable | Smart Insert | 35:56:1188

17. Remove this use of "getStatusCodeValue"; it is deprecated.

The screenshot shows an IDE interface with several windows:

- Package Explorer**: Shows the project structure with packages like com.fms.reactive.repository, com.fms.reactive.request, and com.fms.reactive.service.
- BookingControllerTest.java**: The current file being edited, containing a test method `testHandleValidationErrors()`.
- SonarQube Console**: A panel displaying SonarQube analysis results, including:
 - Start fetching files for the text and secrets analysis
 - Retrieving all except binary files
 - Starting the text and secrets analysis
 - 1 source file to be analyzed for the text and secrets analysis
 - 1/1 source file has been analyzed for the text and secrets analysis
 - Start fetching files for the binary file analysis
 - There are no files to be analyzed for the binary file analysis
- Right-hand pane**: A detailed view of the SonarQube issue for the deprecated code:
 - "@Deprecated" code should not be used**
 - Consistency | Not conventional Maintainability
 - Why is this an issue? More Info
 - Code is sometimes annotated as deprecated by developers maintaining libraries or APIs to indicate that the method, class, or other programming element is no longer recommended for use. This is typically due to the introduction of a newer or more effective alternative. For example, when a better solution has been identified, or when the existing code presents potential errors or security risks.
 - Deprecation is a good practice because it helps to phase out obsolete code in a controlled manner, without breaking existing software that may still depend on it. It is a way to warn

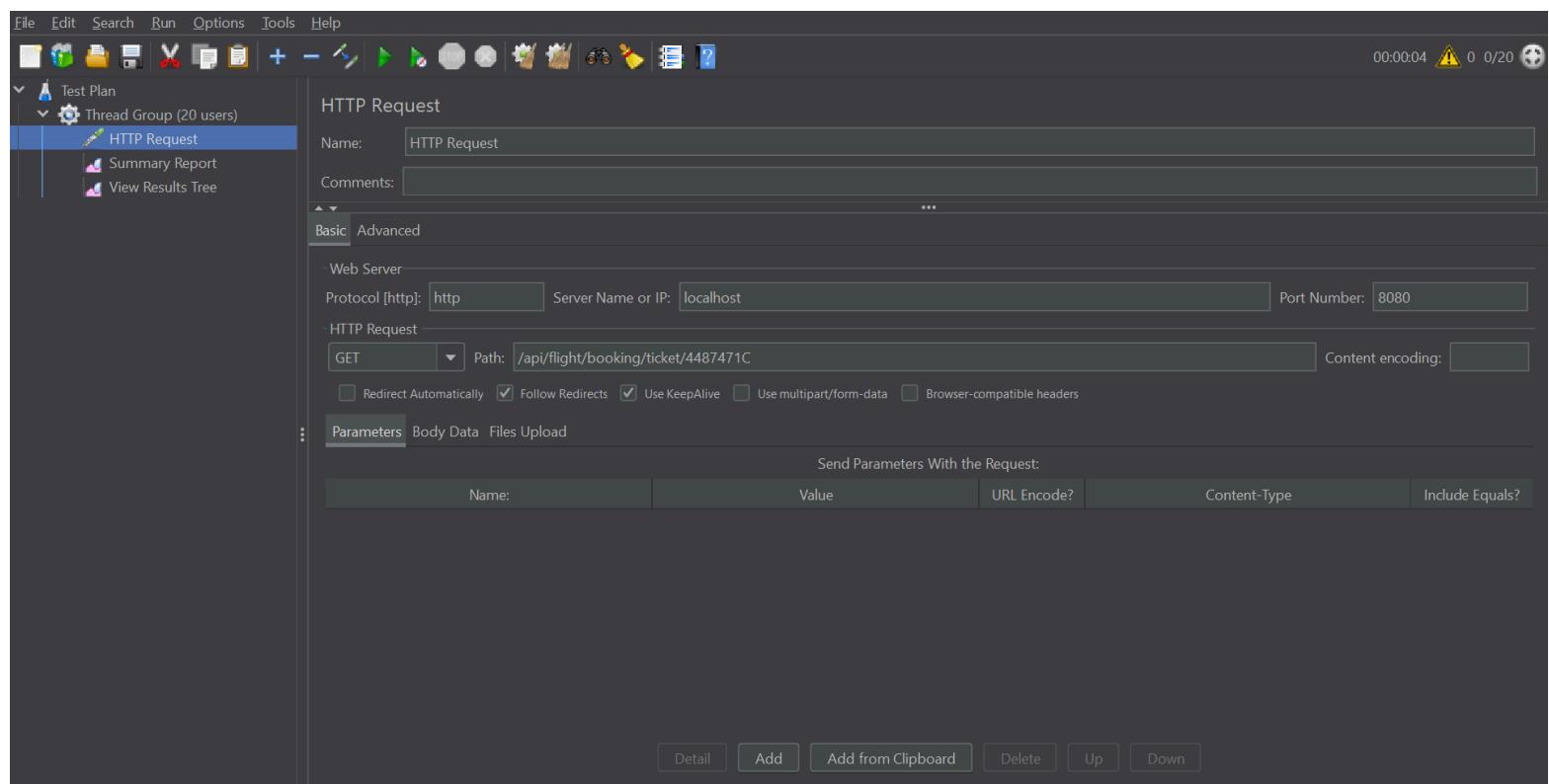
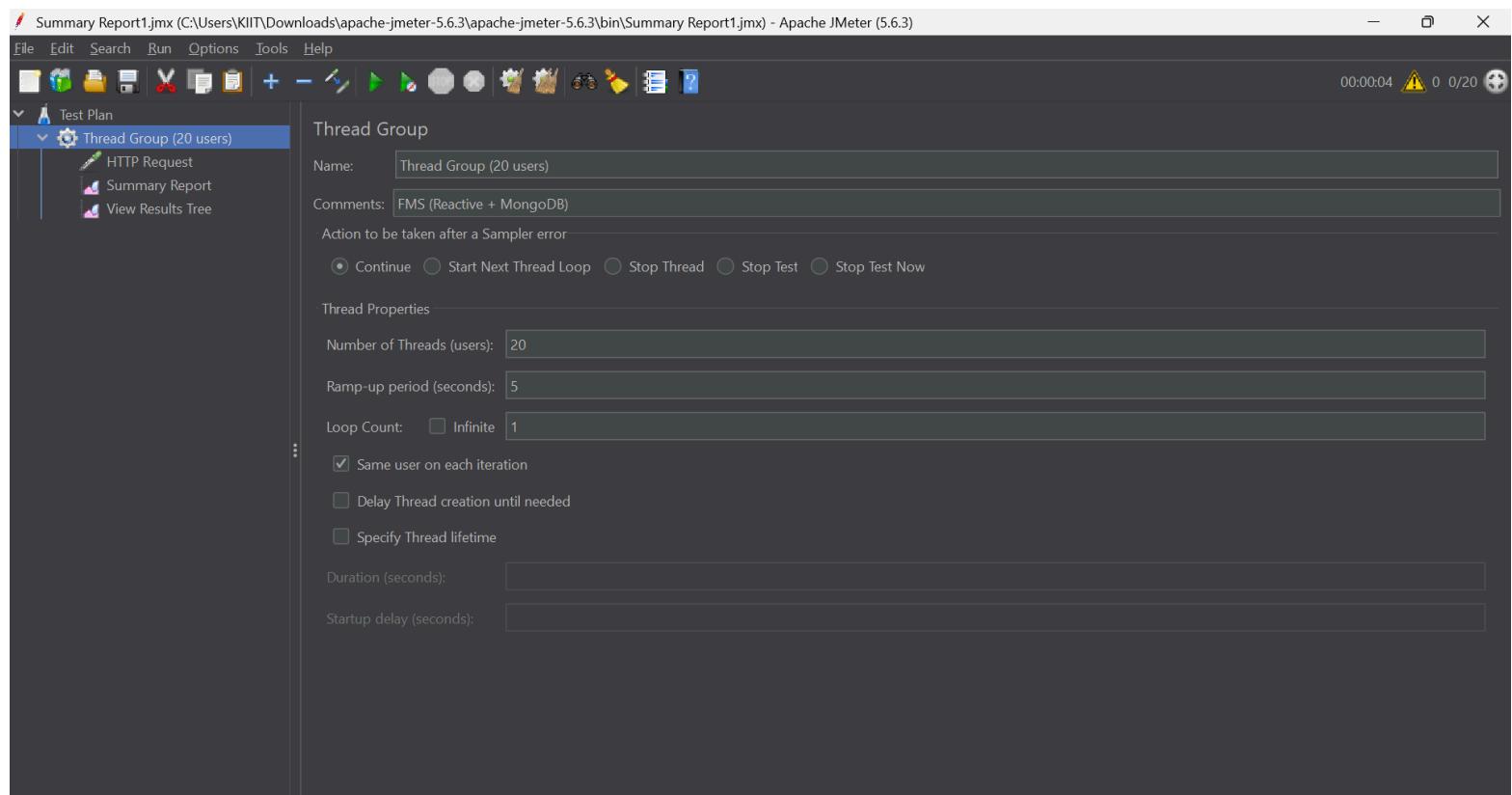
Fix: Added `@SuppressWarnings("deprecation")`

The screenshot shows the same IDE interface after applying the fix:

- BookingControllerTest.java**: The code now includes the annotation `@SuppressWarnings("deprecation")` above the test method `testHandleValidationErrors()`.
- SonarQube Console**: The analysis results remain the same as in the previous screenshot.
- Right-hand pane**: The detailed view of the SonarQube issue is still present, indicating that the code is annotated as deprecated.

JMETER TESTING

CASE 1: 20 Users



Summary Report1.jmx (C:\Users\KIT\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Summary Report1.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:04 ⚠ 0 0/20

Test Plan Thread Group (20 users)
HTTP Request 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/s...	Sent KB/sec	Avg. Bytes
HTTP Request	20	15	9	55	9.47	0.00%	4.3/sec	2.15	0.63	508.0
TOTAL	20	15	9	55	9.47	0.00%	4.3/sec	2.15	0.63	508.0

Include group name in label? Save Table Header

Summary Report1.jmx (C:\Users\KIT\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Summary Report1.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:04 ⚠ 0 0/20

Test Plan Thread Group (20 users)
HTTP Request Summary Report View Results Tree

View Results Tree

Name: View Results Tree
Comments:
- Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes

Search: Case sensitive Regular exp.

Text Sampler result Request Response data

Thread Name: Thread Group (20 users) 1-1
Sample Start: 2025-11-24 20:27:46 IST
Load time: 55
Connect Time: 29
Latency: 51
Size in bytes: 508
Sent bytes: 150
Headers size in bytes: 72
Body size in bytes: 436
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"|""): text
Response code: 200
Response message: OK

HTTPSampleResult fields:
Content-Type: application/json
DataEncoding: null

CASE 2: 50 Users

The screenshot shows the Apache JMeter interface with a test plan containing two thread groups. The left sidebar lists 'Test Plan', 'Thread Group (20 users)', and 'Thread Group (50 Users)'. The 'Thread Group (50 Users)' is selected. The main panel displays its configuration:

- Name:** Thread Group (50 Users)
- Comments:** FMS (Reactive + MongoDB)
- Action to be taken after a Sampler error:** Continue (radio button selected)
- Number of Threads (users):** 50
- Ramp-up period (seconds):** 10
- Loop Count:** Infinite (checkbox selected)
- Thread Properties:**
 - Same user on each iteration
 - Delay Thread creation until needed
 - Specify Thread lifetime
- Duration (seconds):** (empty field)
- Startup delay (seconds):** (empty field)

The screenshot shows the Apache JMeter interface with a test plan containing two thread groups. The left sidebar lists 'Test Plan', 'Thread Group (20 users)', and 'Thread Group (50 Users)'. The 'Thread Group (50 Users)' is selected. The main panel displays its configuration:

- Name:** HTTP Request
- Comments:** (empty field)
- Basic Advanced** tab (Basic is selected)
- Web Server**: Protocol [http]: http, Server Name or IP: localhost, Port Number: 8080
- HTTP Request**: Method: GET, Path: /api/flight/booking/ticket/4487471C, Content encoding: (empty field)
 - Redirect Automatically
 - Follow Redirects
 - Use KeepAlive
 - Use multipart/form-data
 - Browser-compatible headers
- Parameters** tab (selected): Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
(empty)	(empty)	(empty)	(empty)	(empty)

Summary Report1.jmx (C:\Users\KIIT\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Summary Report1.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:09 0 / 0/70

Test Plan

- Thread Group (20 users)
 - HTTP Request
 - Summary Report
 - View Results Tree
- Thread Group (50 Users)
 - HTTP Request
 - 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/s...	Sent KB/sec	Avg. Bytes
HTTP Request	50	15	9	29	3.19	0.00%	5.1/sec	2.53	0.75	508.0
TOTAL	50	15	9	29	3.19	0.00%	5.1/sec	2.53	0.75	508.0

Include group name in label? Save Table Header

Summary Report1.jmx (C:\Users\KIIT\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Summary Report1.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:09 0 / 0/70

Test Plan

- Thread Group (20 users)
 - HTTP Request
 - Summary Report
 - View Results Tree
- Thread Group (50 Users)
 - HTTP Request
 - Summary Report
 - View Results Tree

View Results Tree

Name: View Results Tree

Comments:

- Write results to file / Read from file

Filename Log/Display Only: Errors Successes

Search: Case sensitive Regular exp.

Text

Sampler result Request Response data

<ul style="list-style-type: none">HTTP RequestHTTP Request	Thread Name: Thread Group (50 Users) 2-1 Sample Start: 2025-11-24 20:34:01 IST Load time: 14 Connect Time: 2 Latency: 14 Size in bytes: 508 Sent bytes: 150 Headers size in bytes: 72 Body size in bytes: 436 Sample Count: 1 Error Count: 0 Data type ("text" "bin" ""): text Response code: 200 Response message: OK HTTPSampleResult fields: Content-Type: application/json DataEncoding: null
---	---

CASE 3: 100 Users

Summary Report1.jmx (C:\Users\KIT\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Summary Report1.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:14 ⚠ 0 0/170

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

- Test Plan Tree:** The tree on the left shows three Thread Groups:
 - Thread Group (20 users)
 - Thread Group (50 Users)
 - Thread Group (100 Users)** (selected)
- Thread Group Configuration:** For the selected Thread Group (100 Users), the following settings are visible:
 - Name:** Thread Group (100 Users)
 - Comments:** FMS (Reactive + MongoDB)
 - Action to be taken after a Sampler error:** Continue (radio button selected)
 - Thread Properties:**
 - Number of Threads (users):** 100
 - Ramp-up period (seconds):** 15
 - Loop Count:** Infinite (checkbox checked)
 - Checkboxes:** Same user on each iteration (checked), Delay Thread creation until needed (unchecked), Specify Thread lifetime (unchecked)
 - Duration (seconds):** [empty input field]
 - Startup delay (seconds):** [empty input field]

Summary Report1.jmx (C:\Users\KIT\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Summary Report1.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

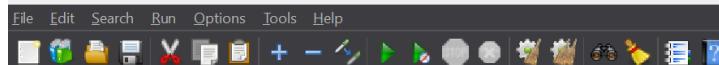
00:00:14 ⚠ 0 0/170

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

- Test Plan Tree:** The tree on the left shows three Thread Groups:
 - Thread Group (20 users)
 - Thread Group (50 Users)
 - Thread Group (100 Users)** (selected)
- HTTP Request Configuration:** For the selected Thread Group (100 Users), the following settings are visible:
 - HTTP Request Sampler:** Name: HTTP Request
 - Basic tab (Web Server):** Protocol [http]: http, Server Name or IP: localhost, Port Number: 8080
 - Advanced tab (HTTP Request):** Method: GET, Path: /api/flight/booking/ticket/4487471C, Content encoding: [empty input field]
 - Checkboxes: Redirect Automatically (unchecked), Follow Redirects (checked), Use KeepAlive (checked), Use multipart/form-data (unchecked), Browser-compatible headers (unchecked)
 - Parameters tab:** Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
[empty input field]				

Buttons at the bottom: Detail, Add, Add from Clipboard, Delete, Up, Down



Test Plan

- Thread Group (20 users)
 - HTTP Request
 - Summary Report
 - View Results Tree
- Thread Group (50 Users)
 - HTTP Request
 - Summary Report
 - View Results Tree
- Thread Group (100 Users)
 - HTTP Request
 - Summary Report
 - View Results Tree

Summary Report

Name:

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/s...	Sent KB/sec	Avg. Bytes
HTTP Request	100	13	8	32	3.27	0.00%	6.7/sec	3.34	0.99	508.0
TOTAL	100	13	8	32	3.27	0.00%	6.7/sec	3.34	0.99	508.0

Include group name in label? Save Table Header

Test Plan

- Thread Group (20 users)
 - HTTP Request
 - Summary Report
 - View Results Tree
- Thread Group (50 Users)
 - HTTP Request
 - Summary Report
 - View Results Tree
- Thread Group (100 Users)
 - HTTP Request
 - Summary Report
 - View Results Tree

View Results Tree

Name:

Comments:

- Write results to file / Read from file

Filename Log/Display Only: Errors Successes

Search: Case sensitive Regular exp.

Text

HTTP Request

Thread Name: Thread Group (100 Users) 3-1
 Sample Start: 2025-11-24 20:38:45 IST
 Load time: 27
 Connect Time: 1
 Latency: 27
 Size in bytes: 508
 Sent bytes: 150
 Headers size in bytes: 72
 Body size in bytes: 436
 Sample Count: 1
 Error Count: 0
 Data type ("text"|"bin"|""): text
 Response code: 200
 Response message: OK

HTTPSampleResult fields:
 ContentType: application/json
 DataEncoding: null

Scroll automatically?

POSTMAN API ENDPOINT TESTING

1. POST: <http://localhost:8080/api/flight/add>

The screenshot shows the Postman interface with a successful POST request to `http://localhost:8080/api/flight/add`. The response status is `201 Created` with a response body containing a flight ID.

Request Body:

```
1 {
2   "airlineId": "A003",
3   "source": "DELHI",
4   "destination": "MUMBAI",
5   "startTime": "2025-12-02T09:30",
6   "price": 4000,
7   "totalSeats": 200
}
```

Response Body:

```
1 {
2   "id": "69247945a8332f6545548853"
3 }
```

2. POST: <http://localhost:8080/api/flight/search>

The screenshot shows the Postman interface with a successful POST request to `http://localhost:8080/api/flight/search`. The response status is `200 OK` with a response body containing flight details.

Request Body:

```
2 {
3   "source": "DELHI",
4   "destination": "MUMBAI",
5   "journeyDate": "2025-12-02",
6   "tripStatus": "ONE WAY"
}
```

Response Body:

```
25 {
26   "id": "692464ecb7db76c14a9f6fef",
27   "airlineId": "A001",
28   "source": "DELHI",
29   "destination": "MUMBAI",
30   "startTime": "2025-12-02T09:30:00",
31   "endTime": "2025-12-02T11:30:00",
32   "price": 4000.0,
33   "totalSeats": 200,
34   "availableSeats": 200
35 }
```

3. POST: <http://localhost:8080/api/flight/booking/6923f4d291e3a6284777061f>

The screenshot shows the Postman interface with a successful POST request. The request URL is <http://localhost:8080/api/flight/booking/6923f4d291e3a6284777061f>. The response status is 201 Created, and the response body is:

```
1 {  
2   "name": "Debashrita",  
3   "email": "debashrita@example.com",  
4   "seats": 2,  
5   "passengers": [  
6     {  
7       "name": "Debashrita Mandal",  
8       "gender": "FEMALE",  
9       "age": 28  
10    },  
11    {  
12      "name": "Aahana Sharma",  
13    }  
14  ]  
15}  
16
```

4. GET: <http://localhost:8080/api/flight/booking/ticket/4487471C>

The screenshot shows the Postman interface with a successful GET request. The request URL is <http://localhost:8080/api/flight/booking/ticket/4487471C>. The response status is 200 OK, and the response body is:

```
1 {  
2   "pnr": "4487471C",  
3   "flightId": "6923f4d291e3a6284777061f",  
4   "name": "Debashrita",  
5   "email": "debashrita@example.com",  
6   "bookingDate": "2025-11-24T19:34:55.71",  
7   "journeyDate": "2025-12-02T09:30:00",  
8   "passengers": [  
9     {  
10        "name": "Debashrita Mandal",  
11        "gender": "FEMALE",  
12        "age": 28  
13      },  
14      {  
15        "name": "Aahana Sharma",  
16        "gender": "FEMALE",  
17        "age": 25  
18      }  
19    ]  
20}
```

5. GET: <http://localhost:8080/api/flight/booking/history/debashrita@example.com>

The screenshot shows the Postman interface with a successful response. The response body is a JSON object representing a flight booking:

```
33 |     "flightId": "6923f4d291e3a6284777061f",
34 |     "name": "Debashrita",
35 |     "email": "debashrita@example.com",
36 |     "bookingDate": "2025-11-24T19:34:55.71",
37 |     "journeyDate": "2025-12-02T09:30:00",
38 |     "passengers": [
39 |         {
40 |             "name": "Debashrita Mandal",
41 |             "gender": "FEMALE",
42 |             "age": 28
43 |         },
44 |         {
45 |             "name": "Aahana Sharma",
46 |             "gender": "FEMALE",
47 |             "age": 25
48 |         }
49 |     ],
50 |     "seatNumbers": [
51 |         "12A"
52 |     ]
53 | }
```

6. DELETE: <http://localhost:8080/api/flight/booking/cancel/4487471C>

The screenshot shows the Postman interface with a successful response after a DELETE request. The response body is a simple message:

```
1 Ticket cancelled successfully
```