



Hotel Booking System API Manual & Automation Testing (Restful-Booker Project)

API Testing using Postman, Java, Rest Assured, TestNG & Allure

Prepared & Presented by: Eng. Omar Zayed

@ ITI Software Testing Track

System Overview – Restful-Booker API

Restful-Booker is a public REST API for hotel booking management.

It simulates a real booking system used in hotel businesses.

The system allows:

- Creating bookings
- Retrieving booking details
- Updating bookings
- Deleting bookings

Official Documentation Link available in references

Requirements Analysis – API Endpoints & Features

Core API Endpoints

- Authentication POST /auth → Generate Token
- Booking Management
 - POST /booking → Create Booking
 - GET /booking/{id} → Get Booking by ID
 - PUT /booking/{id} → Update Booking
 - DELETE /booking/{id} → Delete Booking
- Health Check GET /ping → Verify API is running

Business Features Identified

- User authentication is required for update and delete operations.
- Booking must include mandatory fields:
 - firstname
 - lastname
 - totalprice
 - bookingdates
- totalprice must be a numeric value.
- The system should validate input data.
- Error handling should return proper HTTP status codes.

Test Plan & Test Strategy

Test Objectives

- Validate core booking functionalities.
- Verify input validation & error handling.
- Ensure secured endpoints enforce authentication.
- Detect functional defects through negative testing.
- Verify end-to-end business flows.

Test Scope

In-Scope:

- Authentication
- Booking Management APIs
 - Positive & Negative scenarios
- End-to-End business flows

Test Types

- Functional API testing:
 - Positive testing (Happy path)
 - Negative testing (Validation & handling)
 - End-to-end testing

Test Environment

- Public Demo API: Restful-Booker
- Toolset:
 - Postman (Manual Testing)
 - Java + Rest Assured (Automation)
 - TestNG (Test execution)
 - Allure (Reporting)

Test Design & Test Cases Structure

Test Case Organization

- Manual test cases were first designed using Postman.
- Test cases were grouped by functionality:
 - Authentication
 - Booking Operations
 - Negative Scenarios
 - End-to-End Scenarios

Design Approach

- Each test represents a real business scenario.
- Test data is varied to output system behavior.
- Expected results were defined based on requirements analysis.

Naming Convention

- Pxx → Positive Test Cases
- Nxx → Negative Test Cases
- Sxx → End-to-End Scenarios

Example:

- P01_CreateBooking_Valid
- N04_CreateBooking_Totalprice_As_Text
- S01_Create_Update_Delete_Booking

+ Search collections

▼ Hotel_Booking_Restful_Booker

 ▼ HealthCheck

 GET Ping

 ▼ Setup

 POST GenerateToken

 ▼ Bookings

 POST P01_CreateBooking

 GET P02_GetBookingById_Valid

 PUT P03_UpdateBooking_Valid

 DEL P04_DeleteBooking_Valid

 GET N01_GetBookingById_InvalidId

 PUT N02_UpdateBooking_WithoutAuth

 POST N03_CreateBooking_MissingFields

 POST N04_CreateBooking_Totalprice_As_Text

 POST N05_CreateBooking_Empty_Fname

 ▼ E2E_Scenarios

▼ E2E_Scenarios

 ▼ S01_Create_Update_Delete_Booking

 POST S01_01_GenerateToken

 POST S01_02_CreateBooking

 GET S01_03_GetBooking_AfterCreate

 PUT S01_04_UpdateBooking

 GET S01_05_GetBooking_AfterUpdate

 DEL S01_06_DeleteBooking

 GET S01_07_VerifyBookingDeleted

 ▼ S02_UpdateWithoutAuth

 POST S02_01_CreateBooking

 PUT S02_02_UpdateBooking_WithoutAuth

 GET S02_03_GetBooking_AfterUnauthorizedUp...

RestfulBooker_Env

Variable	Value
baseUrl	https://restful-booker.herokuapp.com
bookingId	401
token	cc8b03114216022
originalLastname	S2_BeforeUpdate
Add variable	

☰ ← → Home Workspaces API Network

Search Postman Ctrl K

Upgrade

Hotel Booking System API Testing Project

New Import

POST P01_CreateBooking | DEL P04_DeleteBooking_Val | POST GenerateToken +

HTTP Hotel_Booking_Restful_Booker / Setup / GenerateToken

Save Share ↗

Collections Environments History Flows

+

Search collections

Hotel_Booking_Restful_Booker

- HealthCheck
- GET Ping
- Setup
- POST GenerateToken

Bookings

- POST P01_CreateBooking
- GET P02_GetBookingById_Valid
- PUT P03_UpdateBooking_Valid
- DEL P04_DeleteBooking_Valid
- GET N01_GetBookingById_InvalidId
- PUT N02_UpdateBooking_WithoutAuth
- POST N03_CreateBooking_MissingFields
- POST N04_CreateBooking_Totalprice_As_Text
- POST N05_CreateBooking_Empty_Fname

E2E Scenarios

- S01_Create_Update_Delete_Booking
 - POST S01_01_GenerateToken
 - POST S01_02_CreateBooking
 - GET S01_03_GetBooking_AfterCreate
 - PUT S01_04_UpdateBooking
 - GET S01_05_GetBooking_AfterUpdate
 - DEL S01_06_DeleteBooking
 - GET S01_07_VerifyBookingDeleted

Cloud View Find and replace Console

Runner Start Proxy Cookies Vault Trash

POST {{baseUrl}}/auth

Docs Params Authorization Headers (8) Body Scripts Settings Cookies

Body (raw JSON)

```
1 {  
2   "username": "admin",  
3   "password": "password123"  
4 }
```

200 OK 144 ms 766 B

Body Cookies Headers (10) Test Results (1/1)

{ } JSON Preview Visualize

```
1 {  
2   "token": "ebc661f2c4436eb"  
3 }
```

Made with GAMMA

Create booking

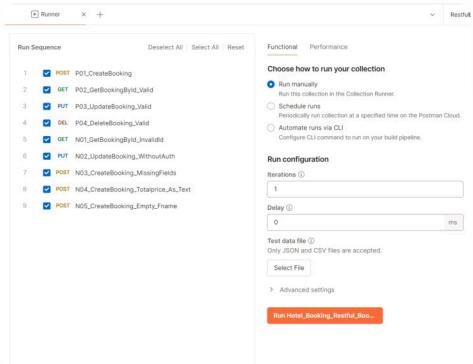
This screenshot shows the Postman interface for a POST request to the '/bookings' endpoint. The request body is a JSON object with fields: name, lastnames, totalprice, checkin, and checkout. A preview of the JSON body is shown below the body field. The response status is 200 OK, and the response body contains a booking object with an id of 1.

This screenshot shows the Postman interface for a POST request to the '/bookings' endpoint. The request body is identical to the one in the previous screenshot. The response status is 200 OK, and the response body contains a booking object with an id of 1. The 'Test Results' tab shows a green success message: 'Booking created successfully'.

This screenshot shows the Postman interface for a POST request to the '/bookings' endpoint. The request body is identical to the ones above. The response status is 200 OK, and the response body contains a booking object with an id of 1. The 'Test Results' tab shows a green success message: 'Booking created successfully'.

This screenshot shows the Postman interface for a GET request to the '/bookings/{bookingId}' endpoint with an invalid booking ID. The response status is 404 Not Found, and the response body is empty. The 'Test Results' tab shows a green success message: 'Booking deleted'.

Booking_Collection_Run



The screenshot shows the Postman Runner interface with the following details:

- Run Sequence:** A list of 11 API requests (POST, GET, PUT, DELETE) with checkboxes next to them.
- Functional Performance:** A section titled "Choose how to run your collection" with options for "Run this collection in the Collection Runner" (selected), "Schedule runs", and "Automate runs via CLI".
- Run configuration:** Set to 1 iteration, 0 ms delay, and a test data file (JSON or CSV).
- Test data file:** Only JSON and CSV files are accepted.
- Run Hotel_Booking_Restful_Bo...** button at the bottom.

Hotel_Booking_Restful_B - Run results

Run today at 08:28:18 AM | View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	RestfulBooker_Env	1	2s 697ms	11	146 ms

RUN SUMMARY

Test	Status	Count
POST P01_CreateBooking	PASS	1
GET P02_GetBookingById_Valid	PASS	3
PUT P03_UpdateBooking_Valid	PASS	1
DELETE P04_DeleteBooking_Valid	PASS	1
GET N01_GetBookingById_InvalidId	FAIL	0
PUT N02_UpdateBooking_WithoutAuth	FAIL	0
POST N03_CreateBooking_MissingFields	FAIL	0
POST N04_CreateBooking_Totprice_A_Text	FAIL	0
POST N05_CreateBooking_Empty_Fname	FAIL	0

Hotel_Booking_Restful_B - Run results

Run today at 08:28:18 AM | View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	RestfulBooker_Env	1	2s 697ms	11	146 ms

AI Test: Passed (3) Failed (3) Skipped (0)

Test Results:

- GET Bookings - N01_GetBookingById_InvalidId: PASS
- PUT Bookings - N02_UpdateBooking_WithoutAuth: PASS
- POST Bookings - N03_CreateBooking_MissingFields: FAIL
- POST Bookings - N04_CreateBooking_Totprice_A_Text: FAIL
- POST Bookings - N05_CreateBooking_Empty_Fname: FAIL

Newman Run Dashboard

Thursday, 04 December 2025 01:37:15

Total Iterations	Total Assertions	Total Failed Tests	Total Skipped Tests
1	23	3	0

FILE INFORMATION

- Collection: Hotel_Booking_Restful_Booker
- Environment: RestfulBooker_Env

TIMINGS AND DATA

- Total run duration: 4.9s
- Total data received: 2.16KB
- Average response time: 177ms

From Manual Testing to Automation

Why Automation?

- Manual API testing becomes time-consuming.
- Regression testing is difficult manually.
- Automation reduces human error.
- Ensures consistent execution of test cases.
- Allows easy re-testing after changes.

Automation Goals

- Convert core scenarios into automated tests.
- Ensure repeatability and stability.
- Detect defects automatically.
- Generate professional reports.
- Build a reusable framework.

Selected Technology Stack

Java

Rest Assured

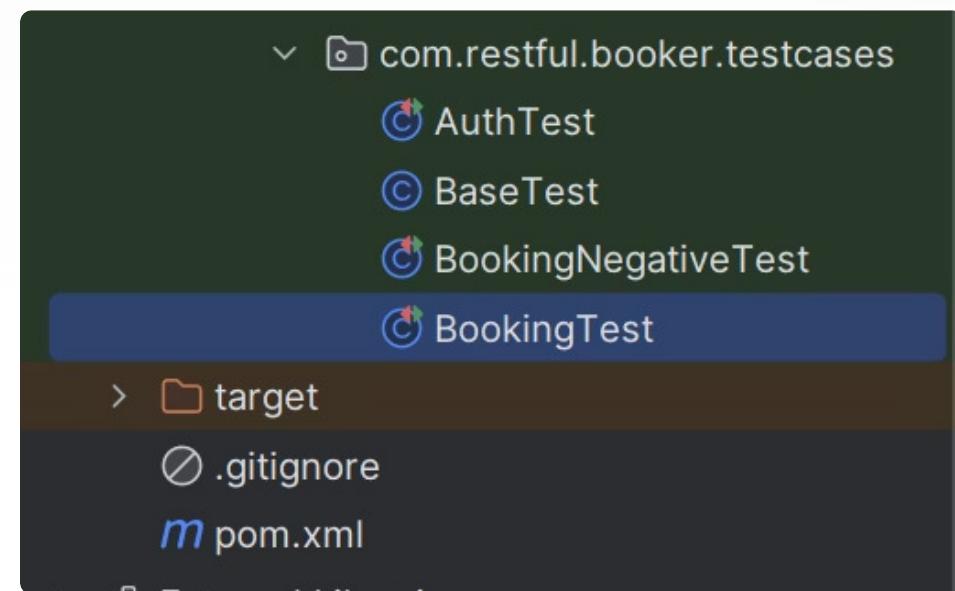
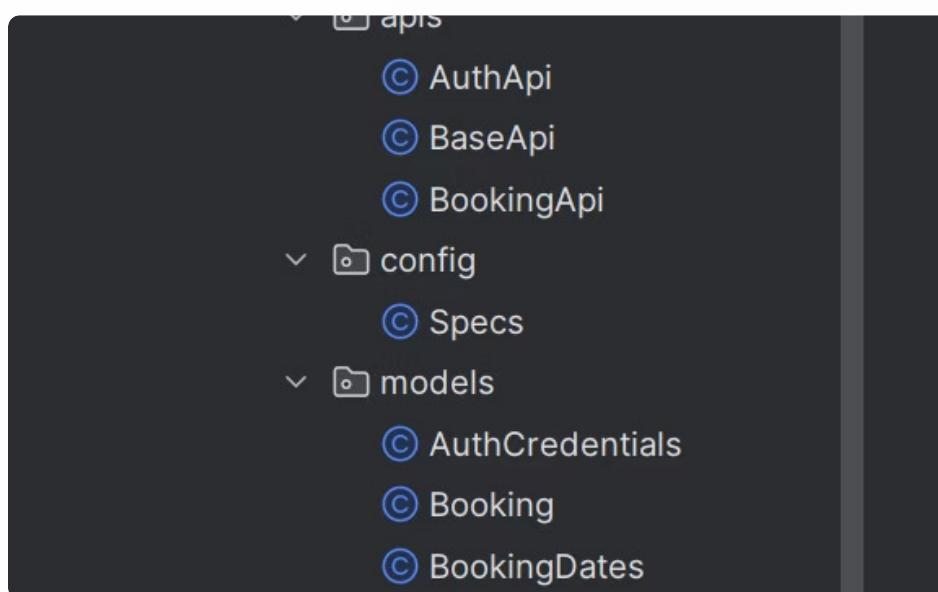
TestNG

Allure Report

Framework Architecture & Project Structure

Design Principles

- Layered architecture
- Readable and maintainable code



Test Execution & Reporting

Test Execution

- Tests executed using TestNG.
- Automation runs through Maven.
- Suites organized by test type:
 - Positive Tests
 - Negative Tests
 - Authentication Tests
 - End-to-End Scenarios

Allure Reporting

- Generates detailed HTML reports.
- Displays:
 - Passed & Failed test cases
 - Each test's steps
 - Request & response details

Outcome

- All positive tests passed successfully.
- Negative tests revealed real system defects.
- End-to-End scenarios validated business flow.

POM–Structure–Tests

```
20 <dependencies>
21
22     <!-- Rest Assured -->
23     <dependency>
24         <groupId>io.rest-assured</groupId>
25         <artifactId>rest-assured</artifactId>
26         <version>5.5.6</version>
27     </dependency>
28
29     <!-- Override vulnerable commons-codec -->
30     <dependency>
31         <groupId>commons-codec</groupId>
32         <artifactId>commons-codec</artifactId>
33         <version>1.17.0</version>
34     </dependency>
35
36     <!-- TestNG -->
37     <dependency>
38         <groupId>org.testng</groupId>
39         <artifactId>testng</artifactId>
```

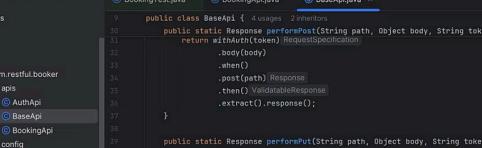
```
44     <!-- Jackson (for JSON ⇄ POJO) -->
45     <dependency>
46         <groupId>com.fasterxml.jackson.core</groupId>
47         <artifactId>jackson-databind</artifactId>
48         <version>2.20.1</version>
49     </dependency>
50
51     <!-- Faker for test data -->
52     <dependency>
53         <groupId>net.datfaker</groupId>
54         <artifactId>datafaker</artifactId>
55         <version>2.5.2</version>
56     </dependency>
57
58     <!-- Allure + TestNG integration -->
59     <dependency>
60         <groupId>io.qameta.allure</groupId>
```

```
1 package com.restful.booker.models;
2
3 public class Booking { 20 usages
4
5     private String firstname; 3 usages
6     private String lastname; 3 usages
7     private int totalprice; 3 usages
8     private boolean depositpaid; 3 usages
9     private BookingDates bookingdates; 3 usages
10    private String additionalneeds; 3 usages
11
12    public Booking() { no usages
13    }
14
15    public Booking(String firstname, String lastname, int totalprice, 2 usages
16                      boolean depositpaid, BookingDates bookingdates, String additionalneeds) {
17        this.firstname = firstname;
18        this.lastname = lastname;
19        this.totalprice = totalprice;
20        this.depositpaid = depositpaid;
21        this.bookingdates = bookingdates;
22        this.additionalneeds = additionalneeds;
23    }
24 }
```

```
13 @
14 public static Booking generateValidBooking() { 10 usages
15
16     String firstname = faker.name().firstName();
17     String lastname = faker.name().lastName();
18     int totalprice = faker.number().numberBetween(50, 500);
19     boolean depositpaid = true;
20
21     LocalDate checkinDate = LocalDate.now().plusDays(daysToAdd: 1);
22     LocalDate checkoutDate = checkinDate.plusDays(daysToAdd: 3);
23
24     BookingDates bookingDates = new BookingDates(
25         checkinDate.toString(),
26         checkoutDate.toString()
27     );
28
29     String additionalneeds = "Breakfast";
30
31     return new Booking(
32         firstname,
33         lastname,
34         totalprice,
35         depositpaid,
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "ct". It includes "main" (containing "src" and "java"), "apis" (containing "AuthApi" and "BaseApi"), "config" (containing "Specs"), "models" (containing "AuthCredentials", "Booking", "BookingDates", and "TokenResponse"), and "utils" (containing "AuthData", "BookingData", "Routes", and "TokenManager").
- Code Editor:** The main editor area displays `BookingTest.java`. The code includes methods for `BaseApi` and `AuthApi`, such as `withAuth` and `performPost`.
- Toolbars and Status Bar:** The top bar shows tabs for "BookingTest.java", "BookingApi.java", and "BaseApi.java". The status bar at the bottom right shows "44:24 CRLF UTF-8".



```
public class BaseApi {  
    public static Response performPost(String path, Object body, String token) {  
        return withAuth(token).requestSpec(specification)  
            .when()  
                .post(path).response()  
            .then().validatableResponse  
                .extract().response();  
    }  
  
    public static Response performPut(String path, Object body, String token) {  
        return withAuth(token).requestSpecification  
            .body(body)  
            .when()  
                .put(path).response()  
            .then().validatableResponse  
                .extract().response();  
    }  
  
    public static Response performDelete(String path, String token) {  
        return withAuth(token).requestSpecification  
            .when()  
                .delete(path).response()  
            .then().validatableResponse  
                .extract().response();  
    }  
}
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "main".
- Code Editor:** The main editor shows the `BookingTest.java` file, which includes imports for `io.restassured.response.Response` and `com.restful Booker.apis.BookingApi`. It contains several static methods for testing booking operations like creating, getting, updating, and deleting bookings.
- Code Coverage:** A vertical bar on the left indicates code coverage for the `BookingApi` class, with a red section covering the `createBooking` method and a green section covering the `getBookingId` method.
- Toolbars and Status Bar:** The top has standard IntelliJ toolbars and a status bar showing "Unlock Ultimate". The bottom status bar shows "File BestRestBooker API TestingProject src main java com restful Booker apis BookingApi 6.1 CRUI LITE-R".

```
tes.java x

package com.restful.booker.utils;
!
public class Routes { 11 usages

    // Authentication
    public static final String AUTH = "/auth";  2 usages

    // Booking
    public static final String BOOKINGS = "/booking";  5 usages
}
```

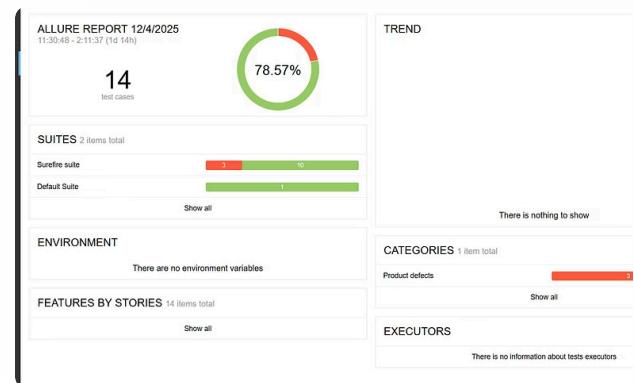
```
1 package com.restful.booker.config;
2
3 import io.restassured.http.ContentType;
4 import io.restassured.specification.RequestSpecification;
5
6 import static io.restassured.RestAssured.given;
7
8 public class Specs { 4 usages
9
10 @
11     private static String getBaseUrl() { 1 usage
12         String env = System.getProperty("key:env", def: "PRODUCTION");
13         return switch (env.toUpperCase()) {
14             case "PRODUCTION" -> "https://restful-booker.herokuapp.com";
15             case "LOCAL" -> "http://localhost:3001";
16             default -> throw new RuntimeException("Environment is not supported: " + env);
17         };
18     }
19
20     public static RequestSpecification getRequestSpec() { 2 usages
21         return given()
22             .baseUrl(getBaseUrl())
23             .contentType(ContentType.JSON)
24             .log().all();
25     }
26 }
```

```
1 package com.restful.booker.models;
2
3 import com.fasterxml.jackson.annotation.JsonProperty;
4
5 public class TokenResponse { 5 usages
6
7     @JsonProperty("token") 2 usages
8     private String token;
9
10     public TokenResponse() { no usages
11     }
12
13     public String getToken() { return token; }
14
15     public void setToken(String token) { this.token = token; }
16
17 }
18
19
20
21
```

```
1 package com.restful.booker.utils;
2
3 import com.restful.booker.apis.AuthApi;
4
5 public class TokenManager { 6 usages
6
7     private static String token; 4 usages
8
9     public static String getToken() {
10         if (token == null || token.isEmpty()) {
11             token = AuthApi.getToken(AuthData.getAdminCredentials());
12         }
13         return token;
14     }
15
16 }
```

```
BookingTest.java
1 package com.restful.booker.testcases;
2
3 import ...
4
5 public class BookingTest extends BaseTest {
6
7     @Test(description = "Should be able to create a booking successfully")
8     public void shouldCreateBookingSuccessfully() {
9
10         // Arrange
11         Booking booking = BookingData.generateValidBooking();
12
13         // Act
14         Response response = BookingApi.createBooking(booking);
15
16         // Assert
17         assertEquals(response.getStatusCode(), is(value: 200));
18
19         Integer bookingId = response.then().extract().path(path: "bookingId");
20         assertNotEquals(isNullValue());
21         assertThat(bookingId, greaterThan(value: 0));
22     }
23
24     @Test(description = "Should be able to get booking by id")
25     public void shouldGetBookingById() {
26
27     }
28 }
```

```
BookingTest.java
1 3 tests failed, 10 passed | 13 tests total, 22 sec 89 ms
2 Default Suite: 22 sec 89 ms
3 HotelBooking_Runner[main]: 89 ms
4 [C:\Program Files\Eclipse\Adoption\jdk-21.0.8-9-hotspot\bin\java.exe] ...
5 SLF4J: No SLF4J providers were found.
6 SLF4J: Defaulting to no-operation (NOP) logger implementation
7 SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
8
9 ✓ BookingTest: 13 ms
10   ✓ bug_ShouldNot[734 ms]
11   ✓ bug_ShouldNot[734 ms]
12   ✓ bug_ShouldNot[734 ms]
13   ✓ bug_ShouldNot[734 ms]
14   ✓ bug_ShouldNot[734 ms]
15   ✓ bug_ShouldNot[734 ms]
16   ✓ bug_ShouldNot[734 ms]
17   ✓ bug_ShouldNot[734 ms]
18   ✓ bug_ShouldNot[734 ms]
19   ✓ bug_ShouldNot[734 ms]
20   ✓ bug_ShouldNot[734 ms]
21   ✓ bug_ShouldNot[734 ms]
22   ✓ bug_ShouldNot[734 ms]
23   ✓ bug_ShouldNot[734 ms]
24   ✓ bug_ShouldNot[734 ms]
25   ✓ bug_ShouldNot[734 ms]
26   ✓ bug_ShouldNot[734 ms]
27   ✓ bug_ShouldNot[734 ms]
28
29 ✓ BookingTest: 1 sec 54 ms
30   ✓ E2ESome[1 sec 300 ms]
31   ✓ shouldCreate[708 ms]
32   ✓ shouldDelete[110 ms]
33   ✓ shouldGet[sec 480 ms]
34   ✓ shouldUpdate[sec 475 ms]
35
36 }
```



Defect Analysis – Discovered Issues

1

Bug #1 – Accepts Empty First Name

- Field: firstname
- Input: Empty string ""
- Expected: Validation error (4xx)
- Actual: Booking created successfully
- Severity: High

2

Bug #2 – Accepts Total Price as Text

- Field: totalprice
- Input: "150" (String instead of Integer)
- Expected: Validation error
- Actual: Booking accepted
- Severity: High

3

Bug #3 – Delete Non-Existing Booking Returns 500

- Input: Invalid booking ID
- Expected: 404 or 4xx client error
- Actual: 500 Internal Server Error
- Severity: Medium

Thank You!

Any Questions?

