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

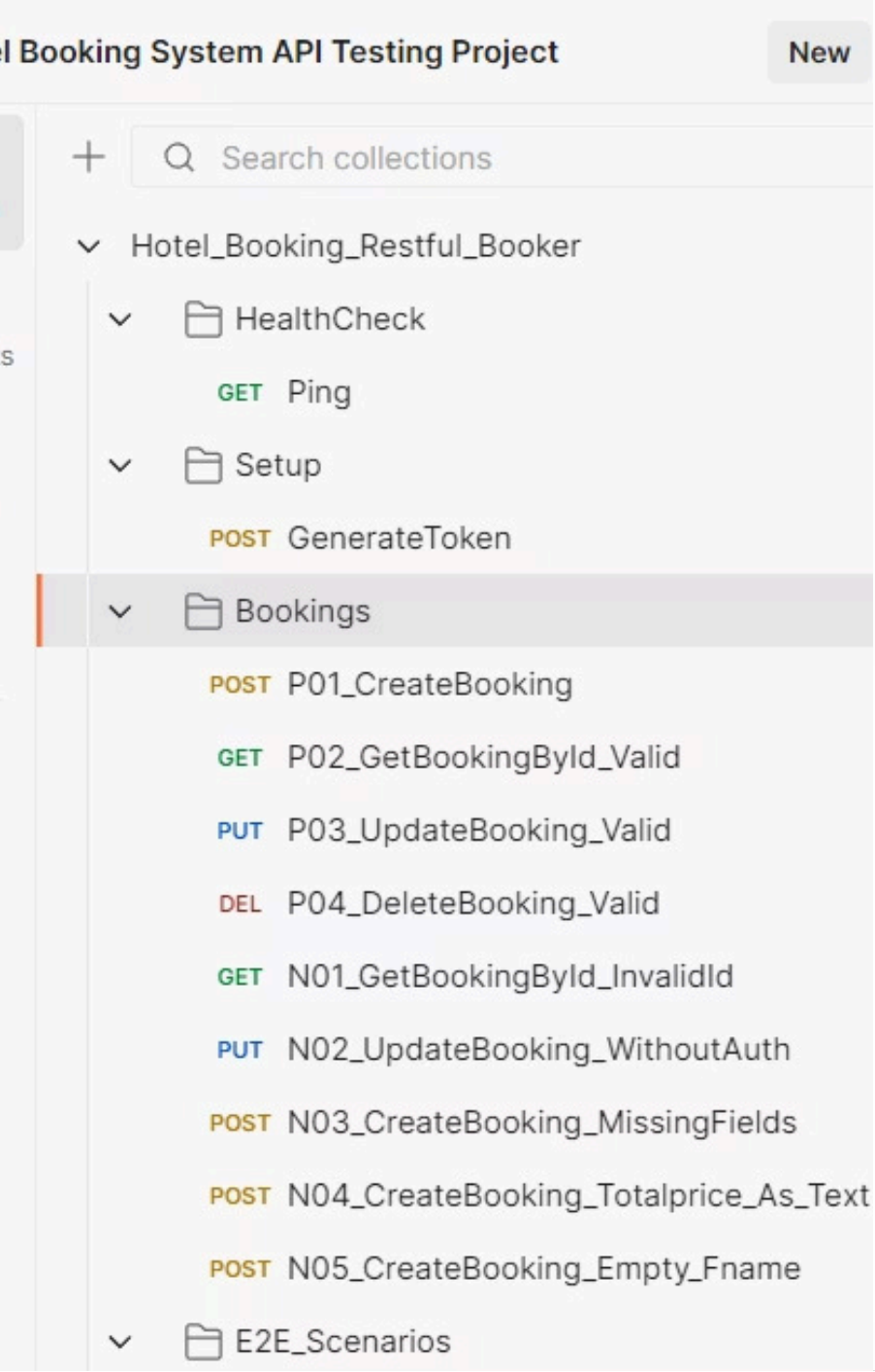
- Each test represents a real business scenario.
- Test data is varied to output system behavior.
- Expected results were defined based on requirements analysis.

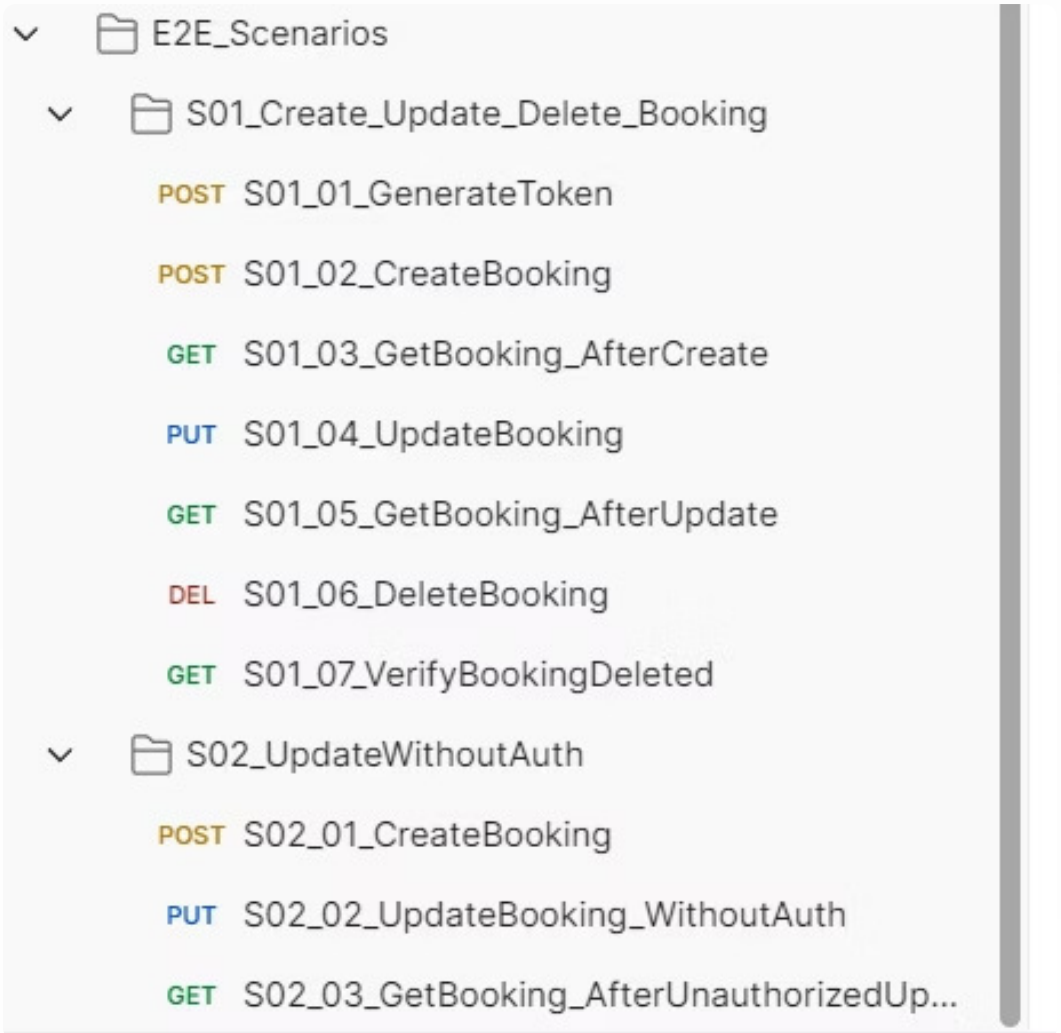
Naming Convention

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

Example:

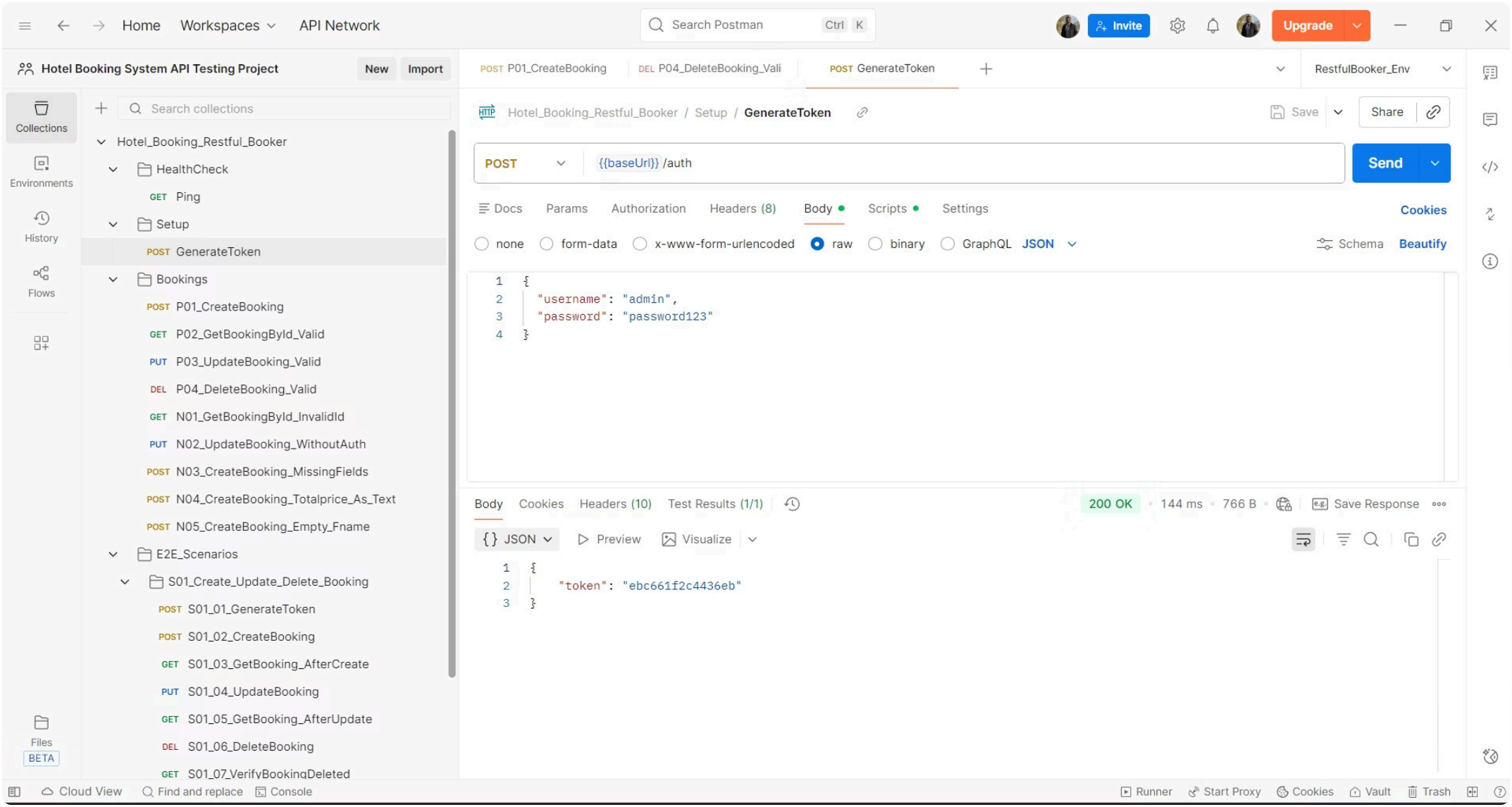
- P01_CreateBooking_Valid
- N04_CreateBooking_Totalprice_As_Text
- S01_Create_Update_Delete_Booking



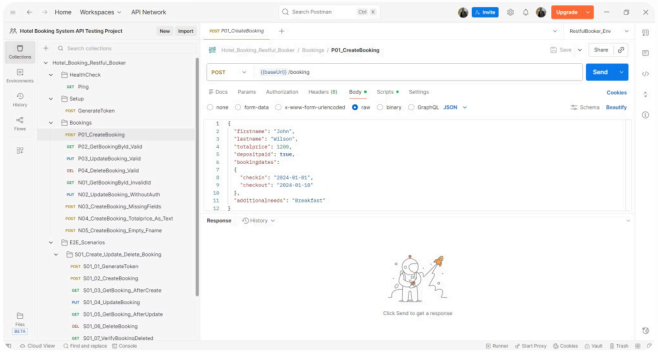


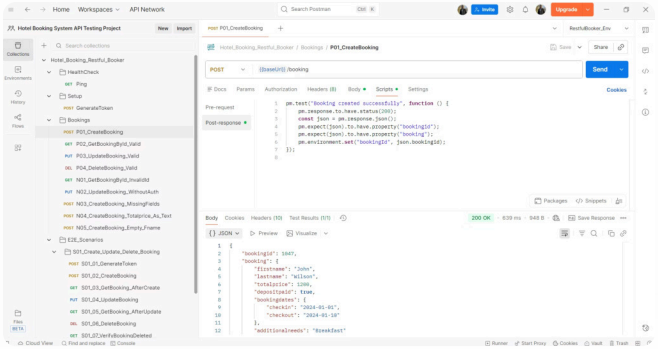
RestfulBooker_Env

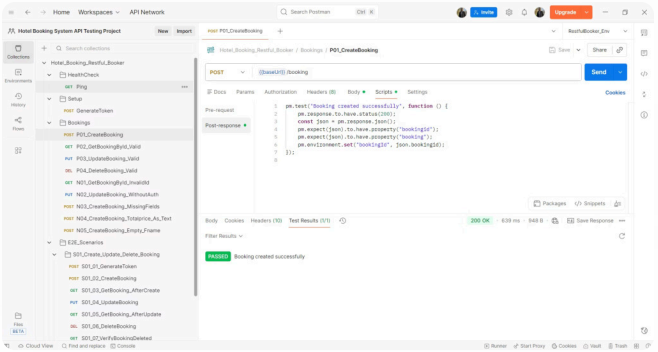
Variable	Value
baseUrl	https://restful-booker.herokuapp.com
bookingId	401
token	cc8b03114216022
originalLastname	S2_BeforeUpdate
Add variable	

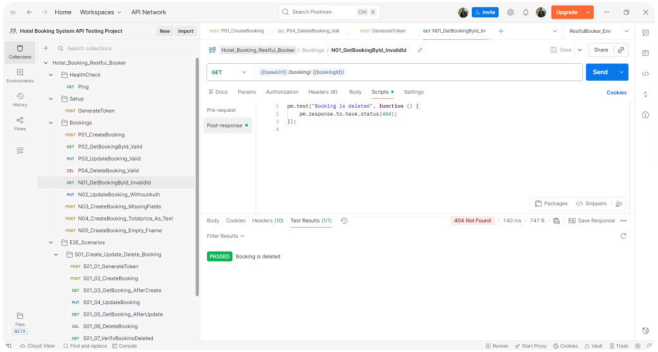


Create booking

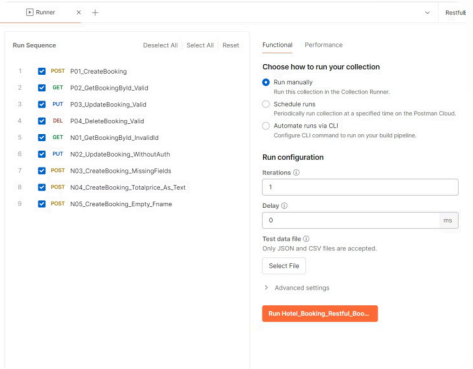


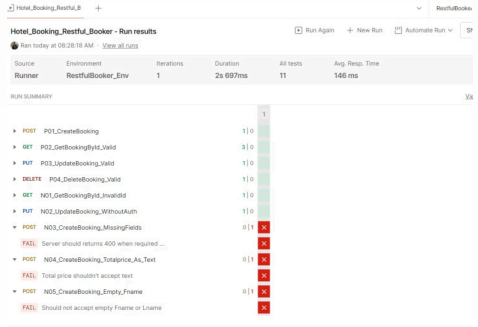


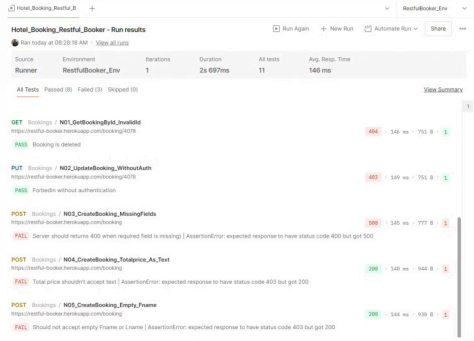




Booking_Collection_Run







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.

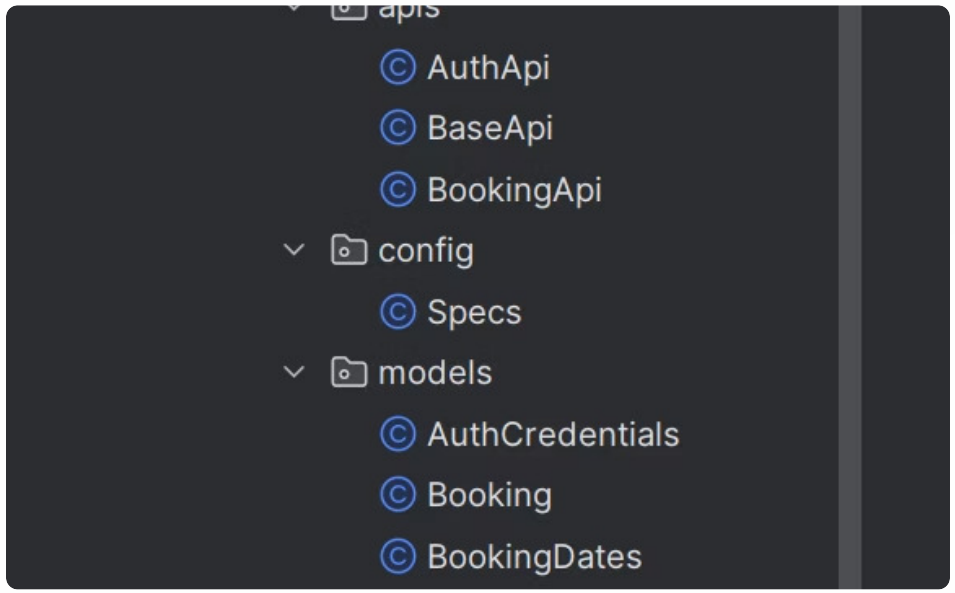
Selected Technology Stack

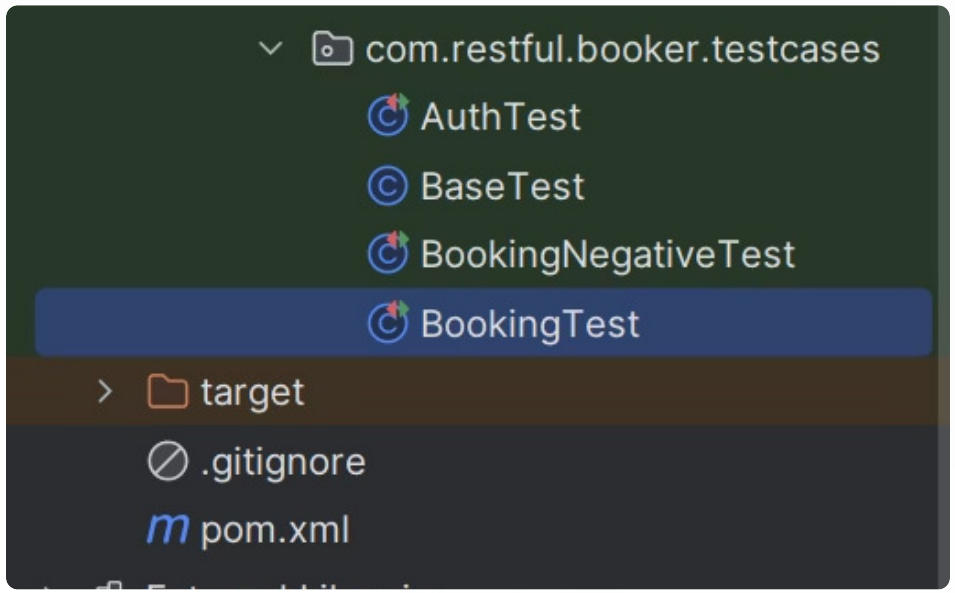
Java	Rest Assured
TestNG	Allure Report

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

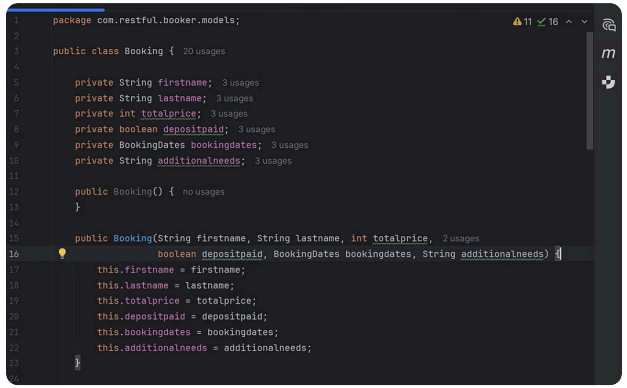
Outcome

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

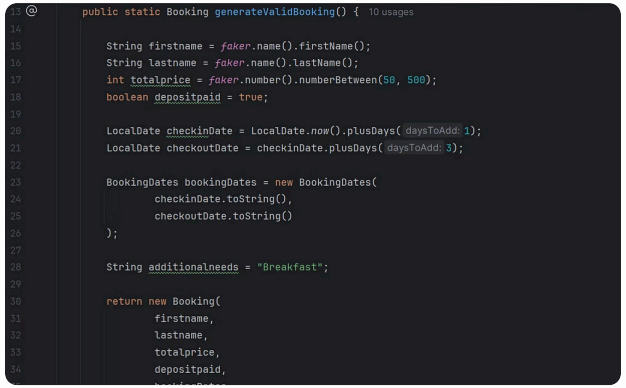
POM–Structure–Tests

```
<dependencies>

    <!-- Rest Assured -->
    <dependency>
        <groupId>io.rest-assured</groupId>
        <artifactId>rest-assured</artifactId>
        <version>5.5.6</version>
    </dependency>

    <!-- Override vulnerable commons-codec -->
    <dependency>
        <groupId>commons-codec</groupId>
        <artifactId>commons-codec</artifactId>
        <version>1.17.0</version>
    </dependency>

    <!-- TestNG -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
```

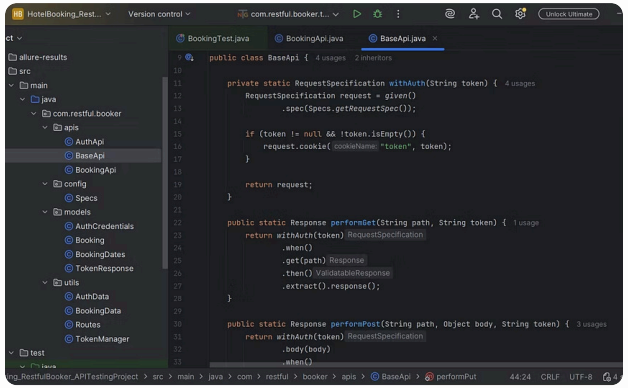
```
    <!-- Jackson (for JSON ↔ POJO) -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.20.1</version>
    </dependency>

    <!-- Faker for test data -->
    <dependency>
        <groupId>net.datafaker</groupId>
        <artifactId>datafaker</artifactId>
        <version>2.5.2</version>
    </dependency>

    <!-- Allure + TestNG integration -->
    <dependency>
        <groupId>io.qameta.allure</groupId>
```



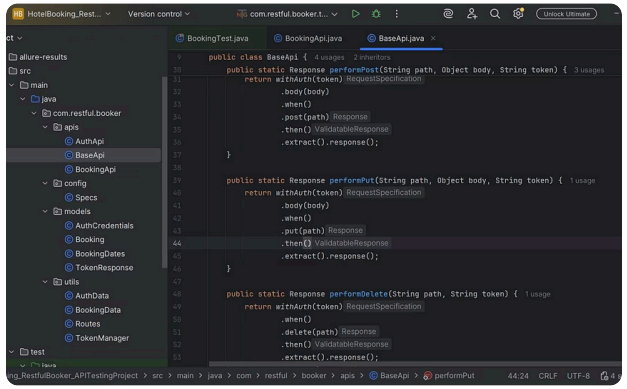
```java
package com.restful.booker.models;

public class Booking {  20 usages

    private String firstname;  3 usages
    private String lastname;  3 usages
    private int totalprice;  3 usages
    private boolean depositpaid;  3 usages
    private BookingDates bookingdates;  3 usages
    private String additionalneeds;  3 usages

    public Booking() {  no usages
    }

    public Booking(String firstname, String lastname, int totalprice,  2 usages
                   boolean depositpaid, BookingDates bookingdates, String additionalneeds) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.totalprice = totalprice;
        this.depositpaid = depositpaid;
        this.bookingdates = bookingdates;
        this.additionalneeds = additionalneeds;
    }
```



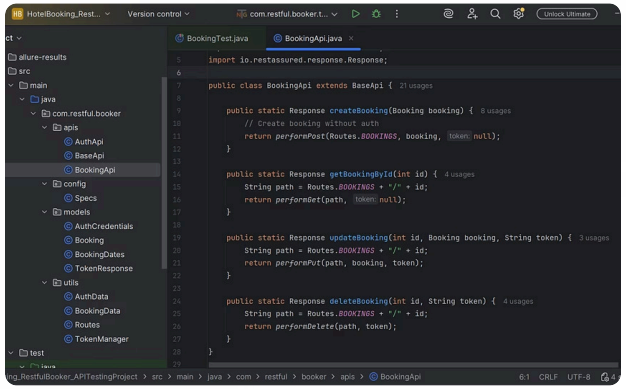
```java
public static Booking generateValidBooking() {  10 usages

    String firstname = faker.name().firstName();
    String lastname = faker.name().lastName();
    int totalprice = faker.number().numberBetween(50, 500);
    boolean depositpaid = true;

    LocalDate checkinDate = LocalDate.now().plusDays( daysToAdd: 1);
    LocalDate checkoutDate = checkinDate.plusDays( daysToAdd: 3);

    BookingDates bookingDates = new BookingDates(
            checkinDate.toString(),
            checkoutDate.toString()
    );

    String additionalneeds = "Breakfast";

    return new Booking(
            firstname,
            lastname,
            totalprice,
            depositpaid,
```



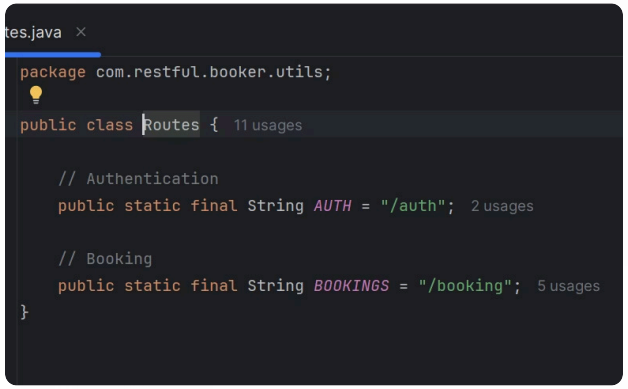
```java
public class BaseApi {  4 usages  2 inheritors

    private static RequestSpecification withAuth(String token) {  4 usages
        RequestSpecification request = given()
                .spec(Specs.getRequestSpec());

        if (token != null && !token.isEmpty()) {
            request.cookie( cookieName: "token", token);
        }

        return request;
    }

    public static Response performGet(String path, String token) {  1 usage
        return withAuth(token) RequestSpecification
                .when()
                .get(path) Response
                .then() ValidatableResponse
                .extract().response();
    }

    public static Response performPost(String path, Object body, String token) {  3 usages
        return withAuth(token) RequestSpecification
                .body(body)
                .when()
```



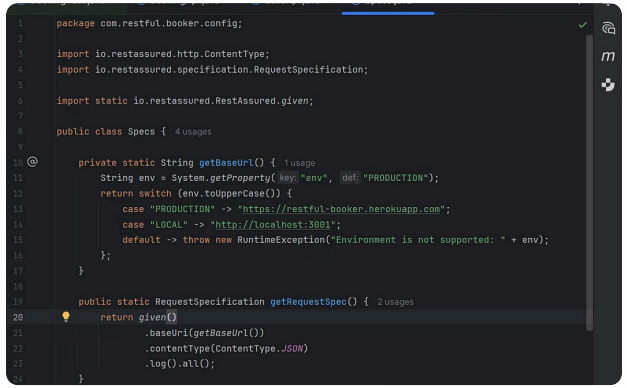
```java
public class BaseApi {  4 usages  2 inheritors
    public static Response performPost(String path, Object body, String token) {  3 usages
        return withAuth(token) RequestSpecification
                .body(body)
                .when()
                .post(path) Response
                .then() ValidatableResponse
                .extract().response();
    }

    public static Response performPut(String path, Object body, String token) {  1 usage
        return withAuth(token) RequestSpecification
                .body(body)
                .when()
                .put(path) Response
                .then() ValidatableResponse
                .extract().response();
    }

    public static Response performDelete(String path, String token) {  1 usage
        return withAuth(token) RequestSpecification
                .when()
                .delete(path) Response
                .then() ValidatableResponse
                .extract().response();
```



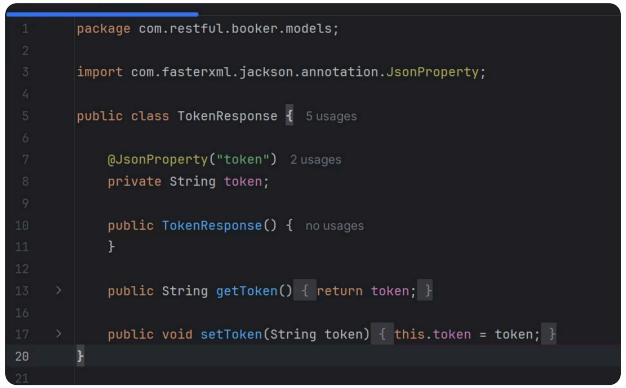
```java
import io.restassured.response.Response;

public class BookingApi extends BaseApi {  21 usages

    public static Response createBooking(Booking booking) {  8 usages
        // Create booking without auth
        return performPost(Routes.BOOKINGS, booking, token: null);
    }

    public static Response getBookingById(int id) {  4 usages
        String path = Routes.BOOKINGS + "/" + id;
        return performGet(path, token: null);
    }

    public static Response updateBooking(int id, Booking booking, String token) {  3 usages
        String path = Routes.BOOKINGS + "/" + id;
        return performPut(path, booking, token);
    }

    public static Response deleteBooking(int id, String token) {  4 usages
        String path = Routes.BOOKINGS + "/" + id;
        return performDelete(path, token);
    }
}
```



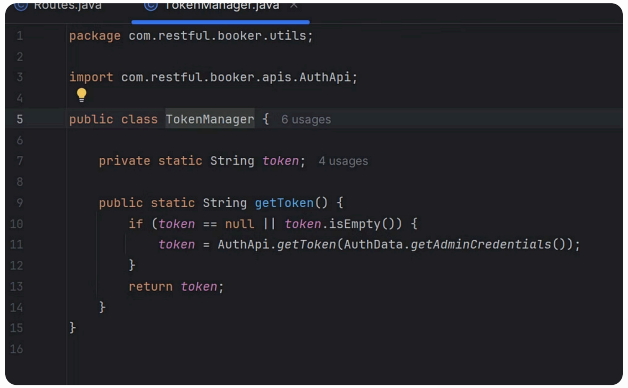
```java
tes.java
package com.restful.booker.utils;

public class Routes {  11 usages

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

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



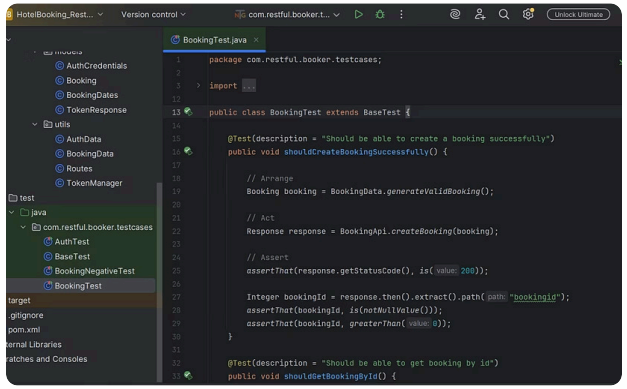
```java
package com.restful.booker.config;

import io.restassured.http.ContentType;
import io.restassured.specification.RequestSpecification;

import static io.restassured.RestAssured.given;

public class Specs {

    private static String getBaseUrl() {
        String env = System.getProperty("env", "PRODUCTION");
        return switch (env.toUpperCase()) {
            case "PRODUCTION" -> "https://restful-booker.herokuapp.com";
            case "LOCAL" -> "http://localhost:3001";
            default -> throw new RuntimeException("Environment is not supported: " + env);
        };
    }

    public static RequestSpecification getRequestSpec() {
        return given()
                .baseUri(getBaseUrl())
                .contentType(ContentType.JSON)
                .log().all();
    }
```



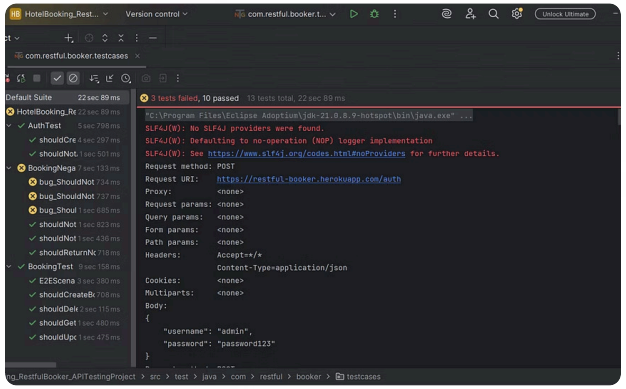
```java
package com.restful.booker.models;

import com.fasterxml.jackson.annotation.JsonProperty;

public class TokenResponse {

    @JsonProperty("token")
    private String token;

    public TokenResponse() {
    }

    public String getToken() { return token; }

    public void setToken(String token) { this.token = token; }
}
```



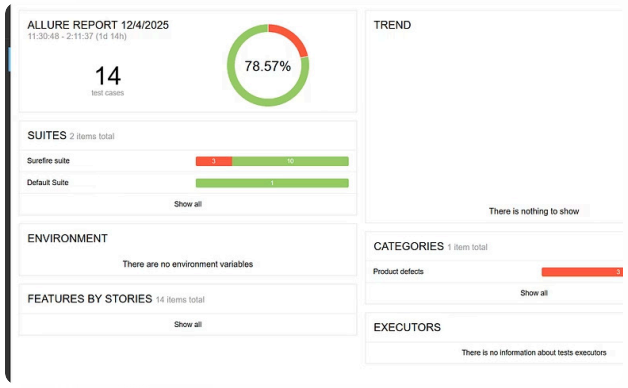
```java
package com.restful.booker.utils;

import com.restful.booker.apis.AuthApi;

public class TokenManager {

    private static String token;

    public static String getToken() {
        if (token == null || token.isEmpty()) {
            token = AuthApi.getToken(AuthData.getAdminCredentials());
        }
        return token;
    }
}
```



```java
package com.restful.booker.testcases;

import ...

public class BookingTest extends BaseTest {

    @Test(description = "Should be able to create a booking successfully")
    public void shouldCreateBookingSuccessfully() {

        // Arrange
        Booking booking = BookingData.generateValidBooking();

        // Act
        Response response = BookingApi.createBooking(booking);

        // Assert
        assertThat(response.getStatusCode(), is( value: 200));

        Integer bookingId = response.then().extract().path( path: "bookingid");
        assertThat(bookingId, is(notNullValue()));
        assertThat(bookingId, greaterThan( value: 0));
    }

    @Test(description = "Should be able to get booking by id")
    public void shouldGetBookingById() {
```



```
Request method: POST
Request URI:    https://restful-booker.herokuapp.com/auth
Proxy:          <none>
Request params: <none>
Query params:   <none>
Form params:    <none>
Path params:    <none>
Headers:        Accept=*/*
                Content-Type=application/json
Cookies:        <none>
Multiparts:     <none>
Body:
{
    "username": "admin",
    "password": "password123"
}
```



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?