# SparkRock Assessment

## Overview

This project aims the testing of the Restful Booker API(https://restful-booker.herokuapp.com/apidoc/index.html) using TestNG and RestAssured frameworks. The API provides endpoints for booking management, including creation, retrieval, update, and deletion of booking records.

## Testing Approach

### Test Strategy

• Functional Testing: Verify that API endpoints meet the functional requirements.  
• Positive Testing: Validate API responses with valid inputs.  
• Negative Testing: Check API behavior with invalid inputs.  
• Boundary Testing: Ensure API handles edge cases.  
• Security Testing: Validate unauthorized access scenarios.

### Tools Used

• Programming Language: Java  
• Framework: TestNG  
• API Testing Library: RestAssured  
• Build Tool: Maven  
• Version Control: Git/GitHub

### Project Structure

restful-booker-api-automation/  
├── src  
│ ├── main  
│ │ └── java  
│ ├── test  
│ │ └── java  
│ │ └── ApiTest  
│ │ ├── BookingApiCrud.java  
│ │ └── testExecution.xml  
├── pom.xml  
└── README.md

### Test Scenarios Covered

1. Health Check API - Ensure API availability with GET /ping.
2. Create Booking - Verify booking creation with valid data, handle missing and invalid input scenarios.  
   3. Get Booking - Retrieve booking by valid/invalid ID, handle missing ID cases.  
   4. Update Booking - Modify an existing booking, ensure handling of incorrect/missing data.  
   5. Delete Booking - Delete existing bookings, validate authorization checks.

### Automation Covered

1. Create Booking - Verify booking creation with valid data, handle missing and invalid input scenarios.
2. Update Booking - Modify an existing booking, ensure handling of incorrect/missing data.
3. Negative scenario for for missing field like booking id

### Execution Steps

1. Clone the repository:

https://github.com/Nkatyal93/Sparkrocktest  
2. Navigate to the project folder:  
 cd restful-booker-api-automation  
  
3. Install dependencies:  
 mvn clean install  
  
4. Execute tests using Maven:  
 mvn test

### Test Data Management

• Static test data is used in the test cases.  
• Dynamic test data such as booking IDs are extracted and reused in subsequent tests.

### Assertions and Validations

• HTTP response status codes are validated.  
• JSON response body values are verified.  
• Error handling is tested for negative scenarios.

### Logging and Reporting

• Request and response logging enabled for debugging.  
• TestNG reports are generated after execution.

## Better Approach

This code is currently designed for assessment purposes. In the future, it can be integrated into a more robust framework, such as implementing the Page Object Model (POM) to enhance code reusability, minimize redundancy, and improve the overall scalability and maintainability of the project.