**RESTful Booker API Testing**

**1. Project Overview**

**1.1 Purpose**

This document outlines the requirements of API testing for the RESTful Booker API available at [Restful Booker API](https://restful-booker.herokuapp.com/apidoc/index.html). The document serves as a guide for testers and developers involved in testing.

**1.2 Scope**

This document covers the functional, performance and security testing requirements. It includes environment setup, test cases design and development, test execution and test reporting.

**1.3 Objectives**

* Verify the functionality of all endpoints.
* Test performance under different load conditions.
* Validate security aspects like authentication and authorization.
* Provide detailed test reports and logs.

**2. API under Test (AUT)**

* **API URL:** <https://restful-booker.herokuapp.com/apidoc/index.html>
* **Description:** RESTful Booker is a sample API for practicing and learning API testing. It provides functionalities to manage bookings includes create, retrieve, modify and delete bookings.

**3. Testing Tools and Framework**

**3.1 Tools**

* **Postman**: For practice API testing and collection creation.
* **Rest Assured**: For Java-based API automation.
* **TestNG**: For test case management and execution.
* **Maven**: For project build management.
* **Jenkins**: For CI/CD integration.
* **Allure**: For test reporting.
* **Git**: For version control.

**3.2 Framework**

* **Language**: Java
* **Libraries**: Rest Assured, TestNG
* **Design Pattern**: Page Object Model (POM)
* **Reporting**: Allure

**4. Environment Setup**

* **Development Environment**: Local machine setup with JDK and IDE (IntelliJ IDEA or Eclipse).
* **Testing Environment**: Access to the API endpoints.
* Create dummy server for practice API testing using following steps :
  1. Install Node JS from [nodejs.org](http://nodejs.org)
  2. Open CMD prompt and use command as npm i -g json-server
  3. Create a db.json file in the working directory using Editor .
  4. Open CMD prompt and use command as json-server -w db.json

JSON Server is a Dummy Server based on the JSON Database

**5. Functionality**

The testing suite should cover the following functionalities of the RESTful Booker API

**Test Cases**

**5.1 Functional Test Cases**

* **Create Booking**
  + Verify that a new booking can be created.
  + Validate required fields and response codes.
* **Get/ Retrieve Booking**
  + Verify retrieval of booking details by ID.
  + Validate response structure.
* **Update Booking**
  + Verify updating an existing booking.
  + Check partial and full updates.
* **Delete Booking**
  + Verify deletion of a booking.
  + Validate response codes and message.

**5.2 Performance Test Cases**

* **Load Testing**
  + Check multiple users to test API response times.
* **Stress Testing**
  + Test API behavior under extreme conditions.

**5.3 Security Test Cases**

* **Authentication**
  + Verify access with valid and invalid credentials.
* **Authorization**
  + Test access control for different user roles.

**6. Test Data**

The test should utilize various test data scenarios to ensure comprehensive testing. This includes:

* Valid and invalid booking data (e.g., missing fields, incorrect data types).
* Existing and non-existing booking IDs for retrieval and deletion operations.
* Different combinations of filter parameters for GET requests.

**7. Expected Results**

Validate the API responses against expected results for each test case.

This includes:

* Status codes (e.g., 200 for success, 400 for bad request).
* Response body and content (e.g., presence of required fields, data types).
* Error messages for invalid requests.

**8. Reporting**

Generate comprehensive reports that include:

* Test case name and description.
* API request and response details (URL, method, headers, body).
* Pass/Fail status for each test case.
* Error messages in case of test failures.

The successful implementation of API testing will improve test coverage, detect defects early, enhance API reliability, and provide detailed reports for continuous improvement. These measures will deliver a high-quality API testing solution that meets current needs and supports future scalability and maintenance.