**Test Plan for RESTful Booker API**

**Introduction**

The purpose of this test plan is to ensure that the RESTful Booker API functions correctly and meets the specified requirements. This document outlines the test strategy, test objectives, test environment, test scope, test cases, and acceptance criteria.

**Test Strategy**

**Testing Types**

1. **Functional Testing**: Verify that each endpoint works as expected according to the API documentation.
2. **Integration Testing**: Ensure that all components of the API work together correctly.
3. **Security Testing**: Test for vulnerabilities and ensure secure access using authentication tokens.
4. **Performance Testing**: Measure the performance of the API under various loads.
5. **Error Handling Testing**: Verify that the API returns appropriate error messages and status codes.

**Test Environment**

* **Base URL**: [https://rahulshettyacademy.com](https://rahulshettyacademy.com/)
* **Tools**: Postman, JMeter, OWASP ZAP, and custom scripts for automation.

**Test Scope**

The test scope includes the following endpoints:

1. /ping - Health Check
2. /location - Create Location
3. /location/{id} - Get Location, Update Location, Delete Location, Partial Update Location
4. /auth - Create Token

**Test Objectives**

1. Verify that the API endpoints work as specified.
2. Ensure that the API handles different types of requests and returns appropriate responses.
3. Validate the security mechanisms, including authentication.
4. Test the API's performance and reliability under load.
5. Ensure proper error handling and messaging.

**Test Cases**

**1. Health Check**

**Test Case ID**: TC-001  
**Endpoint**: /ping  
**Method**: GET  
**Description**: Verify that the API is healthy.  
**Expected Result**: Status code 200 OK.  
**Steps**:

1. Send a GET request to /ping.
2. Verify the response status code is 200 OK.

**2. Create Location**

**Test Case ID**: TC-002  
**Endpoint**: /maps/api/place/add/json   
**Method**: POST  
**Description**: Verify that a new location is created can be created.  
**Expected Result**: Status code 200 OK and a valid location object in the response.  
**Steps**:

1. Send a POST request to /location with a valid JSON body.
2. Verify the response status code is 200 OK.
3. Verify the response body contains the location details.

**3. Get Location**

**Test Case ID**: TC-003  
**Endpoint**: /maps/api/place/get/json  
**Method**: GET  
**Description**: Verify that a Location can be retrieved by PLACE\_ID.  
**Expected Result**: Status code 200 OK and the correct location details in the response.  
**Steps**:

1. Create a new location and note the location ID.
2. Send a GET request to /location/{id} using the noted ID.
3. Verify the response status code is 200 OK.
4. Verify the response body contains the correct location details.

**4. Update Location**

**Test Case ID**: TC-004  
**Endpoint**: maps/api/place/update/json  
**Method**: PUT  
**Description**: Verify that a location can be updated by ID.  
**Expected Result**: Status code 200 OK and the updated location details in the response.  
**Steps**:

1. Create a new location and note the location ID.
2. Send a PUT request to /location/{id} with a valid JSON body to update the location.
3. Verify the response status code is 200 OK.
4. Verify the response body contains the updated location details.

**5. Delete Location**

**Test Case ID**: TC-005  
**Endpoint**: /maps/api/place/delete/json  
**Method**: DELETE  
**Description**: Verify that a location can be deleted by ID.  
**Expected Result**: Status code 201 Created.  
**Steps**:

1. Create a new location and note the location ID.
2. Send a DELETE request to /location/{id}.
3. Verify the response status code is 201 Created.

**6. Partial Update Location**

**Test Case ID**: TC-006  
**Endpoint**: maps/api/place/update/json  
**Method**: PATCH  
**Description**: Verify that a location can be partially updated by ID.  
**Expected Result**: Status code 200 OK and the partially updated location details in the response.  
**Steps**:

1. Create a new location and note the location ID.
2. Send a PATCH request to /location/{id} with a valid JSON body to partially update the location.
3. Verify the response status code is 200 OK.
4. Verify the response body contains the partially updated location details.

**7. Create Token**

**Test Case ID**: TC-007  
**Endpoint**: /auth  
**Method**: POST  
**Description**: Verify that an authentication token can be created.  
**Expected Result**: Status code 200 OK and a valid token in the response.  
**Steps**:

1. Send a POST request to /auth with valid credentials.
2. Verify the response status code is 200 OK.
3. Verify the response body contains a valid token.

**Acceptance Criteria**

1. All test cases must pass with the expected results.
2. The API must handle all valid and invalid requests gracefully.
3. Security mechanisms must be validated and secure.
4. Performance testing should show that the API can handle a reasonable load.
5. Proper error messages and status codes must be returned for invalid requests.

**Roles and Responsibilities**

* **Test Lead**: Responsible for planning and overseeing the testing process.
* **QA Engineers**: Responsible for writing and executing test cases.
* **Developers**: Responsible for fixing any defects found during testing.

**Schedule**

* **Test Planning**: 2 days
* **Test Case Development**: 3 days
* **Test Execution**: 5 days
* **Bug Fixing and Re-testing**: 3 days
* **Final Review and Sign-off**: 1 day