# **Test Plan for Restful Booker API**

**Created by:** Prasad Valiv  
 **Date:** Mar 5, 2025

## **1. Objective**

This document outlines the test plan for the Restful Booker API hosted at<https://restful-booker.herokuapp.com>. The objective is to ensure that all features and functionalities work as expected for API consumers, including creating, reading, updating, and deleting bookings, as well as performing health checks and token-based authentication.

## **2. Scope**

### **2.1 Features to be Tested**

* **Create Booking:** Validate that new bookings can be created with valid payloads.
* **Update Booking:** Verify full update functionality for an existing booking using PUT.
* **PartialUpdateBooking:** Confirm that partial updates work as expected (PATCH, if supported).
* **Delete Booking:** Ensure bookings can be removed using DELETE.
* **Ping - HealthCheck:** Validate the API’s availability and response through a simple health check endpoint.
* **Auth - CreateToken:** Validate the creation of authentication tokens.
* **Booking - GetBookingIds:** Verify that the API returns correct booking IDs and supports query filters.

### **2.2 Types of Testing**

* **Functional Testing:** Confirm correct responses for valid requests and error messages for invalid requests.
* **Integration Testing:** Check the interaction between different API endpoints (e.g., token generation for updating/deleting bookings).
* **Performance Testing:** Measure API response times and load handling (using tools like JMeter).
* **Security Testing:** Assess vulnerabilities such as authentication flaws and improper data exposure.
* **Regression Testing:** Ensure that modifications do not break existing functionality.

### **2.3 Environments**

* **Operating Systems:** Windows 10, macOS, Linux
* **Browsers:** Google Chrome, Mozilla Firefox, Microsoft Edge (for API tools)
* **Devices:** Desktop, Laptop
* **Network:** Wi-Fi, wired connections
* **Tools:** Postman, SOAP UI, Curl, JMeter
* **Security Protocols:** Token-based authentication via Cookie headers

## **3. Inclusions**

* **Introduction:** Overview of the API, including its purpose, endpoints, and testing objectives.
* **Test Objectives:** Detailed objectives such as ensuring correct booking operations, validating error handling, and achieving performance benchmarks.
* **Detailed Test Cases:** Both positive (valid data) and negative (invalid data) test cases, boundary analysis, and concurrency tests.
* **Reporting and Documentation:** Daily defect logs, test execution reports, and final test summary reports.

## **4. Exclusions**

* UI testing (as the API is backend-only).
* Backend database integrity (unless it directly affects API response).
* Testing features not directly related to booking operations (e.g., third-party integrations).

## **5. Test Environments**

| **Component** | **Details** |
| --- | --- |
| Operating Systems | Windows 10, macOS, Linux |
| Browsers | Chrome, Firefox, Edge |
| Devices | Desktops, Laptops |
| Network Connectivity | Wi-Fi, Cellular, Wired |
| Tools | Postman, SOAP UI, Curl, JMeter |
| Security | Token-based authentication (Cookie header) |
| Access Permissions | Testers, Developers, Test Leads |

## **6. Defect Reporting Procedure**

* **Defect Identification:** Any deviation from expected output, performance degradation, or security vulnerability.
* **Reporting Steps:**
  + Log defects using a tool (e.g., JIRA or Bugzilla).
  + Include detailed reproduction steps, screenshots/logs, and affected endpoints.
  + Assign severity:
    - **Critical:** API downtime or incorrect data (e.g., wrong booking details).
    - **High:** Major functional failures (e.g., booking not created/updated).
    - **Medium:** Performance issues (e.g., response times exceeding acceptable limits).
    - **Low:** Minor UI/formatting issues in responses.
* **Communication:** Daily status updates via emails/Slack; weekly defect triage meetings.

## **7. Test Strategy**

### **7.1 Test Scenarios & Test Cases Creation**

* **Techniques:**
  + Equivalence Partitioning
  + Boundary Value Analysis
  + Decision Table Testing
  + Exploratory Testing
  + Error Guessing

### **7.2 Testing Procedure**

1. **Smoke Testing:** Verify the health check endpoint (/ping) to ensure the API is up and running.
2. **Functional Testing:**
   * **Create Booking:** Use the provided cURL command to post valid booking data and validate the response.
   * **Update Booking:** Test full updates using PUT requests with valid token authentication.
   * **Partial Update (if applicable):** Validate if partial data changes are correctly applied.
   * **Delete Booking:** Ensure DELETE operations remove the correct booking.
   * **Get Booking:** Retrieve bookings using GET and verify response data.
   * **Auth - CreateToken:** Confirm that token creation returns a valid token.
   * **Booking - GetBookingIds:** Verify that booking IDs are correctly returned and optionally filtered.
3. **Negative Testing:**
   * Send invalid JSON payloads, missing required fields, or incorrect tokens and validate error handling.
4. **Performance Testing:**
   * Use JMeter to simulate concurrent requests (e.g., 10, 100, 500 users) and measure response times (target: under 2 seconds for most calls).
5. **Security Testing:**
   * Test for unauthorized access by omitting the token in protected endpoints.
   * Validate that error messages do not expose sensitive information.
6. **Regression Testing:**
   * Re-run critical test cases after any API updates or fixes.

### **7.3 Best Practices**

* Context-Driven Testing: Adapt tests based on observed behavior and API documentation updates.
* Shift-Left Testing: Begin testing as early as possible in the development cycle.
* End-to-End Flow Testing: Simulate complete user journeys (e.g., create booking, update, then delete).

## **8. Test Schedule**

| **Activity** | **Duration** | **Timeline** |
| --- | --- | --- |
| Test Plan Creation | 2 days | [Start Date] – [End Date] |
| Test Case Development | 3 days | [Start Date] – [End Date] |
| Test Environment Setup | 1 day | [Start Date] |
| Test Execution (Functional) | 5 days | [Start Date] – [End Date] |
| Performance & Security Testing | 3 days | [Start Date] – [End Date] |
| Defect Reporting & Fixing | Ongoing during testing | Daily |
| Test Closure & Reporting | 2 days | [Start Date] – [End Date] |

## **9. Test Deliverables**

* Test Plan Document (this document)
* Detailed Test Cases and Scenarios
* Test Execution Reports (logs, screenshots, performance graphs)
* Defect Reports and Closure Summary
* Final Test Summary Report

## **10. Entry and Exit Criteria**

### **10.1 Entry Criteria**

* API documentation is received and reviewed.
* API endpoints are deployed and accessible in the test environment.
* Test data is prepared and test tools are set up.
* Test cases are reviewed and approved.

### **10.2 Exit Criteria**

* All planned test cases have been executed.
* All critical and high severity defects are resolved or accepted.
* Final test reports and summary documents are submitted and signed off.

## **11. Tools**

* **Test Management:** Microsoft Excel/Word, TestRail (if available)
* **Defect Tracking:** JIRA or Bugzilla
* **API Testing:** Postman, SOAP UI, Curl
* **Performance Testing:** JMeter
* **Automation Scripting:** Newman (for Postman collections), REST Assured (Java), Python Requests
* **Documentation:** Microsoft Word

## **12. Risks and Mitigations**

| **Risk** | **Mitigation** |
| --- | --- |
| API endpoint downtime | Use mock servers; schedule tests during known uptime periods |
| Delays in API deployment or updates | Plan for regression testing; maintain close communication with dev teams |
| Inconsistent test data or environment | Standardize test environments; use automated environment setups |
| Security vulnerabilities not detected | Conduct thorough security testing; use penetration testing tools |

## **13. Approvals**

| **Role** | **Name** | **Signature/Approval** |
| --- | --- | --- |
| Test Lead | [Your Name] | [Signature/Date] |
| QA Manager | [Manager Name] | [Signature/Date] |
| Project Manager | [PM Name] | [Signature/Date] |

## **Appendix**

### **Sample cURL Commands**

**Create Booking:**

bash

CopyEdit

curl -X POST \

https://restful-booker.herokuapp.com/booking \

-H 'Content-Type: application/json' \

-d '{

"firstname" : "Jim",

"lastname" : "Brown",

"totalprice" : 111,

"depositpaid" : true,

"bookingdates" : {

"checkin" : "2018-01-01",

"checkout" : "2019-01-01"

},

"additionalneeds" : "Breakfast"

}'

**Update Booking:**

bash

CopyEdit

curl -X PUT \

https://restful-booker.herokuapp.com/booking/1 \

-H 'Content-Type: application/json' \

-H 'Accept: application/json' \

-H 'Cookie: token=abc123' \

-d '{

"firstname" : "James",

"lastname" : "Brown",

"totalprice" : 111,

"depositpaid" : true,

"bookingdates" : {

"checkin" : "2018-01-01",

"checkout" : "2019-01-01"

},

"additionalneeds" : "Breakfast"

}'

**Delete Booking:**

bash

CopyEdit

curl -X DELETE \

https://restful-booker.herokuapp.com/booking/1 \

-H 'Content-Type: application/json' \

-H 'Cookie: token=abc123'

**Get Booking:**

bash

CopyEdit

curl -i https://restful-booker.herokuapp.com/booking/1

**Ping - HealthCheck:**

bash

CopyEdit

curl -i https://restful-booker.herokuapp.com/ping

**Auth - CreateToken:**

bash

CopyEdit

curl -X POST \

https://restful-booker.herokuapp.com/auth \

-H 'Content-Type: application/json' \

-d '{

"username": "admin",

"password": "password123"

}'

**Booking - GetBookingIds:**

bash

CopyEdit

curl -i https://restful-booker.herokuapp.com/booking

This Test Plan provides a structured approach to ensuring the quality, functionality, and reliability of the Restful Booker API. It covers manual, automated, performance, security, and regression testing with clearly defined objectives, scope, environments, and reporting procedures.

Feel free to modify any sections as needed for your project requirements.