# **API Test Plan – Restful Booker Playground API**

## **1. Introduction**

The *Restful Booker* API is a RESTful web service for testing booking-related functionality. This test plan outlines the strategy and scope of testing its endpoints, covering authentication, booking management, partial updates, and deletion. The purpose is to ensure that all endpoints behave as expected, meet functional requirements, and handle both valid and invalid scenarios.

## **2. Objectives**

* Verify that each API endpoint returns correct responses for valid inputs.
* Validate authentication and authorization mechanisms for restricted endpoints.
* Ensure proper error handling for invalid/missing parameters, invalid payloads, and unauthorized access.
* Confirm data consistency across create, update, and delete operations.
* Validate compliance with REST API best practices (HTTP methods, status codes, content negotiation).
* Assess performance for basic response time requirements.

## **3. Scope**

### **In Scope**

* All endpoints listed in Restful Booker API documentation:
  1. **Auth**
     + POST /auth – Create Token
  2. **Booking**
     + GET /booking – Get all booking IDs
     + GET /booking/:id – Get specific booking
     + POST /booking – Create booking
     + PUT /booking/:id – Update booking (full)
     + PATCH /booking/:id – Update booking (partial)
     + DELETE /booking/:id – Delete booking
  3. **Ping**
     + GET /ping – Health check

### **Out of Scope**

* UI testing (if any)
* Load testing beyond basic sanity checks
* Security testing beyond token/auth verification

## **4. Test Approach**

Testing will be **black-box functional testing** using **Postman** and **automated scripts** (optional). Each endpoint will be tested for:  
- **Positive scenarios** (valid payloads, valid auth)  
- **Negative scenarios** (invalid payloads, missing fields, expired/invalid tokens, wrong HTTP methods)  
- **Boundary testing** (extreme and edge case values)  
- **Data integrity checks** after Create/Update/Delete

Where applicable, tests will validate: - **HTTP status codes** (200, 201, 400, 401, 404, 500, etc.)  
- **Response time** (basic check: under 2s for normal requests)  
- **Response body structure and field types**  
- **Content-Type compliance**

## **5. Test Environment**

* **Base URL:** https://restful-booker.herokuapp.com
* **Tools:**
  + Postman (manual testing)
  + Newman (CLI automation for Postman collections)
  + JSON schema validators (optional)
* **Data:**
  + Default credentials: username: admin, password: password123
  + Test data sets for bookings (different dates, needs, prices)
* **Test Accounts:**
  + One valid admin account
  + Additional dummy accounts for negative testing

## **6. Test Data Strategy**

* Dynamic booking data generation for Create tests
* Static booking IDs for retrieval and negative testing
* Token re-generation for authorized endpoints before execution
* JSON and XML payload testing to validate content negotiation

## **7. Entry Criteria**

* API endpoints are accessible and stable in test environment
* API documentation is complete and up to date
* Test data is prepared (sample bookings if needed)
* Postman workspace and environment variables configured

## **8. Exit Criteria**

* 100% execution of planned test cases
* No high or critical severity defects remain open
* All mandatory functional scenarios passed
* Test summary report prepared and reviewed

## **9. Test Cases Coverage Summary**

| **Module** | **Endpoint** | **Positive Scenarios** | **Negative Scenarios** |
| --- | --- | --- | --- |
| **Auth** | POST /auth | Valid credentials → 200 + token | Invalid username/password, missing fields, wrong Content-Type |
| **GetBookingIds** | GET /booking | All IDs, filter by firstname, lastname, checkin, checkout | Invalid query params, date format errors |
| **GetBooking** | GET /booking/:id | Valid booking ID → correct data | Invalid ID, non-existent ID, wrong Accept header |
| **CreateBooking** | POST /booking | Valid payload (JSON/XML), all fields correct | Missing fields, invalid data types, invalid dates |
| **UpdateBooking (Full)** | PUT /booking/:id | Valid token + full payload update | Missing token, expired token, invalid ID, invalid payload |
| **PartialUpdateBooking** | PATCH /booking/:id | Valid token + partial fields | Invalid token, invalid payload, no fields sent |
| **DeleteBooking** | DELETE /booking/:id | Valid token + valid ID → 201 | Missing/invalid token, invalid ID, deleting already deleted booking |
| **Ping** | GET /ping | API returns 201 → OK | Wrong HTTP method, wrong endpoint |

## **10. Risks & Mitigation**

* **Risk:** API downtime during testing → **Mitigation:** Schedule testing during stable periods, use /ping health checks before execution.
* **Risk:** Token expiration mid-test → **Mitigation:** Automate token generation step before restricted endpoint calls.
* **Risk:** Data collisions in shared environment → **Mitigation:** Use unique test data and cleanup scripts.

## **11. Deliverables**

* Test plan (this document)
* Detailed test cases in tabular format
* Postman collection with environment variables
* Test execution report with pass/fail status and defect list

## **12. Roles & Responsibilities**

* **QA Engineer:** Create and execute test cases, raise defects
* **API Developer:** Fix issues, provide clarifications
* **QA Lead:** Approve test coverage, track progress

## **13. Schedule**

| **Task** | **Duration** | **Owner** |
| --- | --- | --- |
| Test case design | 2 days | QA |
| Test execution (manual) | 2–3 days | QA |
| Bug verification | 1–2 days | QA |
| Final report | 1 day | QA Lead |