Test Plan for Hotel Booking API

Created by: Deepak Kumar

---

1. Objective

This document outlines the test plan for the **Hotel Booking API** application. The objective is to ensure that all API endpoints and functionalities work as expected for the target audience, **developers and QA engineers** integrating or testing booking system APIs.

---

2. Scope

The scope of this test plan includes:

- **Features to be tested**:
    - Authentication (Create Token)
    - Create Booking
    - Get Booking (by ID & all)
    - Update Booking (PUT and PATCH)
    - Delete Booking
    - Health Check
- **Types of testing**:
    - Manual testing
    - Automated testing
    - Performance testing

    - Security testing (Basic Auth validation)

- **Environments**:
    - REST clients (Postman, Swagger)
    - Different browsers and OS for frontend (if applicable)
    - Tools like cURL for CLI testing
- **Evaluation criteria**:
    - Number of defects found
    - Response time (Performance)
    - Accuracy of returned data
    - Pass/fail rate of test cases
- **Team roles and responsibilities**:
    - **Test Lead**: Plan, monitor, review testing process
    - **Testers**: Execute test cases, report defects
    - **Developers**: Fix defects
    - **DevOps**: Manage environment setup and test data

---

3. Inclusions

- **Introduction**: This API testing plan ensures proper validation of REST endpoints provided by the Hotel Booking system.
- **Test Objectives**:
    - Validate correctness of data across API responses
    - Confirm authentication and role-based access

    - Check behavior under load and stress
    - Ensure error messages and status codes are accurate

---

4. Exclusions

- Frontend UI testing (if not in scope of API testing)
- Integration with external booking engines or payment gateways
- Database testing (unless APIs are exposing DB-related issues)

---

5. Test Environments

- **Operating Systems**: Windows 10, macOS Monterey
- **Browsers**: Chrome, Firefox, Edge (for API tools)
- **Devices**: Desktop/Laptop (No mobile UI)
- **Network**: Wi-Fi and 4G
- **Requirements**: Postman, Swagger UI, cURL
- **Security Protocols**: Basic Auth for Create/Update/Delete operations
- **Access Permissions**: Token-based access for authorized operations

---

6. Defect Reporting Procedure

- **Criteria**: Deviations from API schema, wrong HTTP status codes, security issues
- **Steps**:

    - Use JIRA for defect logging
    - Provide endpoint, payload, actual vs expected response
    - Attach cURL/Postman evidence
- **Triage**:
    - Severity & Priority set based on business impact
    - Assigned to API developers
- **Tracking Tools**: JIRA, Confluence
- **Communication**: Email, Slack updates, stand-ups
- **Metrics**:
    - Defects by severity
    - Time to resolve
    - Reopened defect count

---

7. Test Strategy

Step 1: Test Scenario and Case Creation

- Techniques:
    - Equivalence Partitioning (valid/invalid data)
    - Boundary Value Analysis (ID ranges)
    - State Transition (token expiration scenarios)
    - Use Case Testing (Booking, Cancelation flows)

Step 2: Testing Procedure

- **Smoke Testing**: Run health check endpoint
- **In-depth Testing**: Test all endpoints with all possible inputs
- **Defect Reporting**: JIRA with daily syncs

Types of Testing:

- Smoke Testing
- Regression Testing
- Retesting
- Security Testing
- Performance Testing
- Negative Testing (invalid payloads)

Step 3: Best Practices

- Context Driven Testing
- Shift Left Testing (involve testers in API design)
- Exploratory Testing (test undocumented behavior)
- End-to-End Flow Testing (e.g., Create → Retrieve → Update → Delete)

---

8. Test Schedule

Task	Start Date	End Date
Test Plan Preparation	Apr 25	Apr 26
Test Case Design	Apr 27	Apr 29
Environment Setup	Apr 27	Apr 28

Test Execution	Apr 30	May 5
Defect Retesting	May 6	May 7
Final Report Submission	May 8	May 9

---

9. Test Deliverables

- Test Plan Document
- Test Scenarios & Test Cases
- Test Execution Report
- Defect Report
- Final Test Summary Report

---

10. Entry and Exit Criteria

Requirement Analysis:

- **Entry**: API documentation available
- **Exit**: Clarified all endpoint behavior

Test Execution:

- **Entry**: Test cases reviewed, environment ready
- **Exit**: All test cases executed, critical bugs fixed

Test Closure:

- **Entry**: Execution reports complete
- **Exit**: Summary shared, lessons learned documented

---

11. Tools

- **Postman**: API Testing
- **Swagger UI**: API documentation and basic testing
- **JIRA**: Defect tracking
- **Excel/Google Sheets**: Test case documentation
- **JMeter**: Load Testing (optional)
- **Snipping Tool**: Screenshots for defect evidence

---

12. Risks and Mitigations

Risk	Mitigation
Unavailability of API endpoint	Work on offline test cases/documentation
Test data instability	Use mock servers or controlled data
Environment downtime	Use alternate base URLs if available

---

13. Approvals

Documents for Client Approval:

- Test Plan
- Test Cases
- Execution Reports
- Summary Report