# **Test Plan (Restful Booker Webservice)**

[**Test Plan**](#_heading=h.30j0zll) **1**

[Objective](#_heading=h.3znysh7) 2

[Scope](#_heading=h.2et92p0) 2

[Inclusions](#_heading=h.tyjcwt) 4

[Test Environments](#_heading=h.3dy6vkm) 6

[Defect Reporting Procedure](#_heading=h.1t3h5sf) 7

[Test Strategy](#_heading=h.4d34og8) 8

[Test Schedule](#_heading=h.2s8eyo1) 9

[Test Deliverables.](#_heading=h.17dp8vu) 10

[Entry and Exit Criteria](#_heading=h.3rdcrjn) 10

[Entry Criteria:](#_heading=h.26in1rg) 10

[Exit Criteria:](#_heading=h.lnxbz9) 11

[Test Execution](#_heading=h.35nkun2) 11

[Entry Criteria:](#_heading=h.1ksv4uv) 11

[Exit Criteria:](#_heading=h.44sinio) 11

[Test Closure](#_heading=h.2jxsxqh) 11

[Entry Criteria:](#_heading=h.z337ya) 11

[Exit Criteria:](#_heading=h.3j2qqm3) 11

[Tools](#_heading=h.1y810tw) 11

[Risks and Mitigations](#_heading=h.4i7ojhp) 11

[Approvals](#_heading=h.2xcytpi)

Objective

The goal of this test plan is to ensure the quality, functionality, and reliability of the Restful

Booker API hosted at https://restful-booker.herokuapp.com.

The API is designed to handle booking requests for a fictional hotel booking system.

Track both leading and lagging indicators for your experiment&#39;s impact

Keep an eye on guardrail metrics to stop bad experiments early on

● Sprint boot

● My SQL

● Documentation via flagger

## Scope

## Scope of Test Plan for Restful Booker API:

## **1. Functional Testing:**

## - Verify the correctness and functionality of all API endpoints as per the API

## documentation.

## - Test various scenarios for booking creation, modification, and cancellation.

## - Validate user authentication and authorization mechanisms for protected endpoints.

## 2. **Data Validation Testing:**

## - Ensure that the API correctly validates input data, rejecting invalid requests.

## - Test boundary values for input fields to check for any unexpected behavior.

## - Validate the accuracy of data returned in responses.

## 3. **Error Handling Testing:**

## - Verify that appropriate error codes and messages are returned for invalid requests.

## - Check error responses for sensitive information disclosure.

## - Validate the API&#39;s ability to handle unexpected errors gracefully.

## 4. **Performance Testing:**

## - Assess the API&#39;s response time under normal and peak loads to identify potential

## bottlenecks.

## - Measure the API&#39;s throughput and scalability to handle concurrent requests.

## **5. Security Testing:**

## - Conduct security assessments to identify vulnerabilities such as SQL injection, XSS, etc.

## - Validate the API&#39;s compliance with secure data transmission practices (e.g., HTTPS).

## - Check for proper access controls to prevent unauthorized access to sensitive resources.

## **6. Integration Testing:**

## - Verify interactions between different API endpoints and services.

## - Test data consistency across related endpoints.

## **7. Compatibility Testing:**

## - Test the API on different platforms, browsers, and devices to ensure cross-compatibility.

## **8. Documentation Review:**

## - Assess the clarity, completeness, and accuracy of the API documentation.

## - Verify that the API documentation is in sync with the actual API behavior.

## **9. Load Testing:**

## - Evaluate the API&#39;s behavior under high concurrent user loads to ensure stability.

## **10. Regression Testing:**

## - Conduct regression testing after bug fixes or updates to ensure existing functionality

## remains intact.

## **11. Edge Case Testing:**

## - Test extreme and boundary scenarios to identify potential issues.

## **12. Concurrency Testing:**

## - Assess the API&#39;s behavior when multiple users attempt to access and modify bookings

## simultaneously.

## **13. Ad Hoc Testing:**

## - Perform exploratory testing to identify any hidden defects or usability issues.

## **14. Usability Testing:**

## - Evaluate the API&#39;s user-friendliness and ease of use from a developer&#39;s perspective.

## **15. Continuous Integration and Deployment (CI/CD) Testing:**

## - Validate the API&#39;s behavior within the CI/CD pipeline to ensure smooth deployments.

## **16. Performance Monitoring:**

## - Implement monitoring to track API performance in real-time.

## **17. Backup and Recovery Testing:**

## - Validate data backup and recovery procedures to ensure data integrity.

## **18. Internationalization Testing:**

## - Test the API&#39;s behavior with different language settings.

## **19. Rate Limiting Testing:**

## - Check the API&#39;s adherence to rate-limiting rules to prevent abuse.

## **20. Third-Party Integration Testing:**

## - Validate any third-party integrations for smooth functioning.

## It&#39;s important to note that the scope of the test plan may evolve during the testing process

## based on feedback, changing requirements, or discoveries during testing. The scope should

## be reviewed and adjusted accordingly throughout the testing phase to ensure

## comprehensive coverage of the Restful Booker API.

## **Inclusions**

## **Create (POST) Operations:**

## Test the API&#39;s ability to create new bookings using valid input data.

## Verify that appropriate error responses are returned for invalid or missing data.

## Validate that newly created bookings are stored correctly in the system.

## **Read (GET) Operations:**

## Test the API&#39;s ability to retrieve booking information by various criteria (e.g., booking ID, date

## range, guest name).

## Verify that the API returns the correct data in response to read requests.

## Test for correct handling of non-existent or invalid booking IDs.

## **Update (PUT) Operations:**

## Test the API&#39;s ability to update existing bookings with valid data.

## Verify that the API rejects invalid update requests with appropriate error responses.

## Validate that the booking data is correctly modified in the system after updates.

## **Delete (DELETE) Operations:**

## Test the API&#39;s ability to delete bookings by providing valid booking IDs.

## Verify that the API returns appropriate responses after successful deletion.

## Validate that the deleted bookings are removed from the system.

## **Boundary Testing**:

## Test the API with minimum and maximum allowed values for input fields.

## Validate the behavior of the API with values close to the boundaries.

## **Concurrency Testing:**

## Test the API&#39;s behavior when multiple users try to perform CRUD operations simultaneously.

## Verify data consistency and handling of concurrent modifications.

## **Data Validation:**

## Test the API&#39;s response to various data validation scenarios (e.g., invalid characters, data

## types, mandatory fields).

## Verify that the API handles validation errors appropriately.

## **Authentication and Authorization:**

## Test CRUD operations for both authenticated and unauthenticated users.

## Verify that only authorized users can perform certain CRUD operations.

## **Error Handling:**

## Test the API&#39;s response when invalid or malformed requests are made for CRUD operations.

## Validate that appropriate error codes and messages are returned.

## **Security Testing:**

## Test for security vulnerabilities during CRUD operations (e.g., SQL injection, XSS).

## Verify that sensitive data is not exposed in responses.

## **Performance Testing:**

## Evaluate the API&#39;s response time for CRUD operations under normal and peak loads.

## Measure the throughput and scalability of the API.

## **Integration Testing:**

## Verify the interaction and data consistency between CRUD operations and other API

## components.

## **Regression Testing:**

## Perform regression tests after bug fixes or updates to ensure existing CRUD functionalities

## remain intact.

## **Documentation Review:**

## Assess the accuracy of API documentation related to CRUD operations.

## **Load Testing:**

## Evaluate the API&#39;s behavior and performance during CRUD operations under high

## concurrent user loads.

## **Compatibility Testing:**

## Test the API&#39;s CRUD operations on different platforms, browsers, and devices.

## **Usability Testing:**

## Evaluate the ease of using CRUD functionalities from a developer&#39;s perspective.

## Continuous Integration and Deployment (CI/CD) Testing:

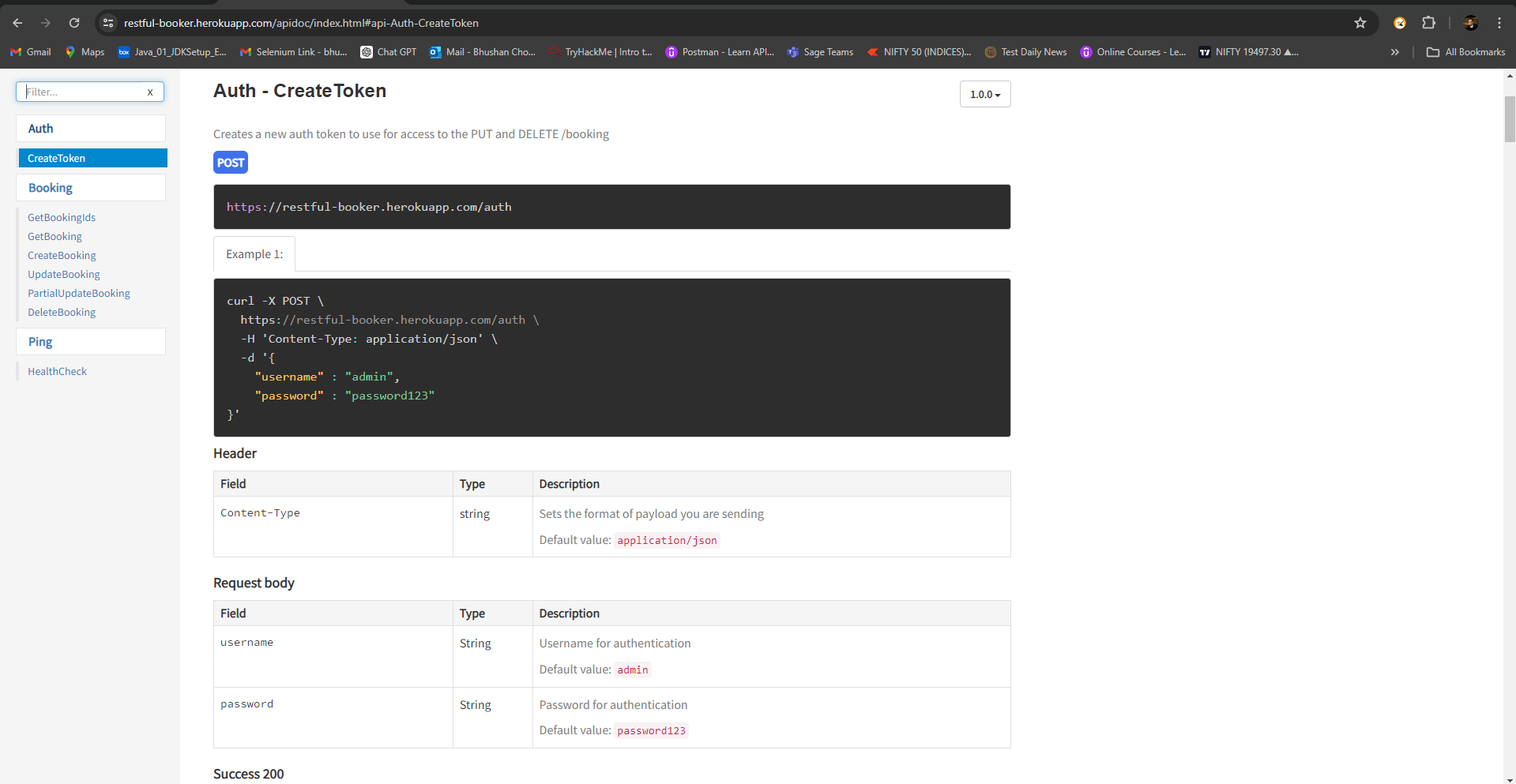
## Validate the CRUD operations within the CI/CD pipeline to ensure smooth deployments.

## **Rate Limiting Testing:**

## Check the API&#39;s adherence to rate-limiting rules for CRUD operations to prevent abuse.

## Backup and Recovery Testing:

## Validate data backup and recovery procedures for CRUD-related data.



## **Test Environments**

## The operating systems and versions that will be used for testing, such as Windows 10,

## macOS, or Linux.

## The browsers and versions that will be tested, such as Google Chrome, Mozilla Firefox, or Microsoft Edge.

## The device types and screen sizes that will be used for testing, such as desktop computers, laptops, tablets, and smartphones.

## The network connectivity and bandwidth that will be available for testing, such as Wi-Fi, cellular, or wired connections.

## The hardware and software requirements for running the test cases, such as a specific

## processor, memory, or storage capacity.

## The security protocols and authentication methods that will be used to access the test

## environment, such as passwords, tokens, or certificates.

## The access permissions and roles of the team members who will be using the test

## environment, such as testers, developers, or stakeholders.

|  |  |
| --- | --- |
| Name | Env. URL |
| QA | https://restful-booker.herokuapp.com/apidoc/index.html |
| Pre PROD | https://restful-booker.herokuapp.com/apidoc/index.html |

## Windows 10 – Chrome, Firefox and Edge

## • Mac OS – Safari Browser

## • Android Mobile OS – Chrome

## • iPhone Mobile OS - Safari

## **Defect Reporting Procedure**

## The criteria for identifying a defect, such as deviation from the requirements, user

## experience issues, or technical errors.

## The steps for reporting a defect, such as using a designated template, providing detailed

## reproduction steps, and attaching screenshots or logs.

## The process for triaging and prioritizing defects, such as assigning severity and priority

## levels, and assigning them to the appropriate team members for investigation and resolution.

## The tools and systems that will be used for tracking and managing defects, such as a

## defect tracking software or a project management tool.

## The roles and responsibilities of the team members involved in the defect reporting

## process, such as testers, developers, and the test lead.

## The communication channels and frequencies for updating stakeholders on the progress and status of defects.

## The metrics and metrics that will be used to measure the effectiveness of the defect

## reporting process, such as the number of defects found, the time taken to resolve them, and the percentage of defects that were successfully fixed.

|  |  |
| --- | --- |
| **Defect Process** | **POC** |
| New Frontend | Dinesh |
| Backend | Srikanth |

## Tools - JIRA

## **Test Strategy**

## The first step is to create test scenarios and test cases for the various features in

## Scope.

## While developing test cases, we&#39;ll use a number of test design techniques.

## o Equivalence Class Partition

## o Boundary Value Analysis

## o Decision Table Testing

## o State Transition Testing

## o Use Case Testing

## We also use our expertise in creating Test Cases by applying the below:

## o Error Guessing

## o Exploratory Testing

## • We prioritize the Test Cases

## Step 2: Our testing procedure when we receive a request for testing:

## • First, we&#39;ll conduct smoke testing to see if the various and

## important functionalities of the application are working.

## • We reject the build, if the Smoke Testing fails and will wait for the stable

## build before performing in depth testing of the application functionalities.

## • Once we receive a stable build, which passes Smoke Testing, we perform

## in depth testing using the Test Cases created.

## • Multiple Test Resources will be testing the same Application on Multiple

## Supported Environments simultaneously.

## We then report the bugs in bug tracking tool and send dev. management

## the defect found on that day in a status end of the day email.

## As part of the Testing, we will perform the below types of Testing:

## o Smoke Testing and Sanity Testing

## o Regression Testing and Retesting

## o Usability Testing, Functionality &amp; UI Testing

## • We repeat Test Cycles until we get the quality product.

## Step3 – We will follow the below best practices to make our Testing better:

## • Context Driven Testing – We will be performing Testing as per the context

## of the given application.

## • Shift Left Testing – We will start testing from the beginning stages of the

## development itself, instead of waiting for the stable build.

## • Exploratory Testing – Using our expertise we will perform Exploratory

## Testing, apart from the normal execution of the Test cases.

## • End to End Flow Testing – We will test the end-to-end scenario which

## involve multiple functionalities to simulate the end user flows.

## **Test Schedule**

## Following is the test schedule planned for the project –

## Task Time Duration

## **Task Dates**

## ▪ Creating Test Plan

## ▪ Test Case Creation

## ▪ Test Case Execution

## ▪ Summary Reports Submission Date

## 2 Sprints to Test the Application

## **Test Deliverables.**



## Entry and Exit Criteria

## The below are the entry and exit criteria for every phase of Software Testing Life

## Cycle:

## Requirement Analysis

## Entry Criteria:

## • Once the testing team receives the Requirements Documents or details

## about the Project

## Exit Criteria:

## • List of Requirements are explored and understood by the Testing team

## • Doubts are cleared

## **Test Execution**

## Entry Criteria:

## • Test Scenarios and Test Cases Documents are signed-off by the Client

## • Application is ready for Testing

## Exit Criteria:

## • Test Case Reports, Defect Reports are ready

## **Test Closure**

## Entry Criteria:

## • Test Case Reports, Defect Reports are ready

## Exit Criteria:

## • Test Summary Reports

## **Tools**

## The following are the list of Tools we will be using in this Project:

## • JIRA Bug Tracking Tool

## • Mind map Tool

## • Snipping Screenshot Tool

## • Word and Excel documents

## **Risks and Mitigations**

## The following are the list of risks possible and the ways to mitigate them:

## Risk: Non-Availability of a Resource

## Mitigation: Backup Resource Planning

## Risk: Build URL is not working

## Mitigation: Resources will work on other tasks

## Risk: Less time for Testing

## Mitigation: Ramp up the resources based on the Client needs dynamically

## **Approvals**

## Team will send different types of documents for Client Approval like below:

## • Test Plan

## • Test Scenarios

## • Test Cases

## • Reports

## Testing will only continue to the next steps once these approvals are done