Test Plan

Project: **Restful-Booker**

Version: 1.0
Date: 01/04/2024

Table of Contents

1. Introduction

1.1. Overview

The Restful-Booker project is a simulated hotel booking system featuring a RESTful API. Users interact with the API to perform tasks such as checking room availability, making bookings, updating reservations, and managing room inventory. The API also includes mechanisms for user authentication and authorization to ensure secure access to the system. The project serves as an educational and practical example for understanding web development concepts and API design principles.

1.2. Objective

The Test Plan documents and tracks the necessary information required to effectively define the approach to be used in the testing of the Restful-Booker project. This Test Plan document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and testing team. Some portions of this document may on occasion be shared with the client/user and other stakeholders whose input/approval into the testing process is needed. This document has clearly identified what the test deliverables will be, and what is deemed in and out of scope. By clearly defining test deliverables, scope, and approach, this document aims to facilitate effective testing execution and mitigate risks associated with system failures or security breaches.

2. Scope

Testing scope defines a list of overall features, functionalities, or aspects of the application that will be tested and that will be excluded from the testing.

2.1. In Scope

All the Restful-Booker API features defined in the Restful Booker Documentation need to be tested.

- Testing of all endpoints provided by the RESTful Booker API, including
    - Authentication
    - Booking management (create, retrieve, update, delete)
- Functional testing to validate the correct behavior of API endpoints.
- Performance testing to evaluate API responsiveness under varying loads.

2.2. Out of Scope

The features listed below will be out of our Testing scope and will not be tested because these features were not included in the requirement spec.

- Health check (PING) API endpoint
- Testing of third-party integrations with the API.
- End-to-end testing involving front-end applications.

3. Test Strategy

3.1. Test Levels

Testing can be categorized into different levels based on the scope and objectives of the testing. The following levels of testing are going to be used to carry out our testing task:

- **Unit Testing**: Testing of individual API endpoints in isolation.
- **Integration Testing**: Verifying interactions between different endpoints.
- **System Testing**: Conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements

3.2. Test Types

Depending on the category there are multiple types of testing. Among all of the testing types, we will be using the below testing types:

- **Functional Testing**: The characteristics of functional testing are to provide correctness, reliability, testability, and accuracy of the report output/data.
- **Non-Functional Testing**: Non-Functional testing is done to evaluate the non-functional aspects of an application, such as performance, scalability, usability, reliability, compatibility, etc.

3.3. Test Techniques

In software testing, various techniques are employed to ensure thorough and effective testing of the system. The following test techniques will be used in API testing:

- **Black-box Testing**: Testing without knowledge of internal implementation.
- **Boundary Value Analysis**: Testing at boundaries of input domains.
- **Equivalence Partitioning**: Dividing input domain into equivalence classes.

3.4. Test Process

The testing process that is going to be followed is mentioned below:

- **Understanding Requirements**:
    - Requirement specifications will be sent by the client.
    - QA will under the requirements and clear out all the confusion about the requirements
- **Preparing Test Cases**: QA will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements.
- **Preparing Test Matrix**: QA will be preparing a test matrix that maps test cases to respective requirements. This will ensure the coverage for requirements.
- **Reviewing test cases and matrix**:
    - Peer review will be conducted for test cases and test matrix by the QA Lead.
    - Any comments or suggestions on test cases and test coverage will be provided by the reviewer's respective Author of the Test Case and the Test

Matrix or improvements will be reworked by the author and will be sent for approval.
    - Re-worked improvements will be reviewed and approved by the reviewer
- **Creating Test Data**: Test data will be created by respective QA on the client's developments/test site based on scenarios and Test cases.
- **Executing Test Cases**: Test cases will be executed by respective QA on the client's development/test site based on designed scenarios, test cases, and Test data.
    - Test results (Actual Result, Pass/Fail) will updated in the test case document Defect Logging and Reporting.
    - QA will be logging the defect/bugs in a Word document, found during execution of test cases. After this, QA will inform the respective developer about the defects/bugs.
- **Retesting and Regression Testing**: Retesting for fixed bugs will be done by respective QA once it is resolved by the respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.
- **Deployment/Delivery**: Once all bugs/defects reported after complete testing are fixed and no other bugs are found, the report will be deployed to the client's test site by PM.

3.5. Severity and Priority Definitions

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. The Test Lead will be responsible for seeing that a correct severity level is assigned to each bug. Test Plan 10 The QA Lead, Development Lead, and Project Manager will participate in bug review meetings to assign priority to all currently active bugs. This meeting will be known as "Bug Triage Meetings". The QA Lead is responsible for setting up these meetings on a routine basis to address the current set of new and existing but unresolved bugs.

Severity:

Severity	Severity Description
Critical	The module/product crashes or the bug causes nonrecoverable conditions
Major	Major system components unusable due to failure or incorrect functionality
Medium	Incorrect functionality of component or process. There is a simple workaround for the bug if it is a Medium
Minor	It won't cause any major break-down of the system

Priority:

Priority	Priority Description
High	The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed
Medium	During the normal course of the development activities, defects should be resolved. It can wait until a new version is created
Low	The Defect is an irritant but repair can be done once the more serious Defect has been fixed

3.6. Bug Reporting

Bugs will be reported using a standardized bug report template in the issue-tracking software: Jira. Each bug will be assigned a unique identifier for tracking purposes.

3.7. Entry-Exit Criteria

Entry Criteria:

- All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system and judge the correct behavior.
- All the standard software tools including the testing tools must have been successfully installed and functioning properly.
- Proper test data is available.
- The test environment such as lab, hardware, software, and system administration support should be ready.

Exit Criteria:

- All test cases are executed, and there are no functional bugs.
- All unresolved bugs are of low severity.
- The schedule has been achieved

3.8. Assumptions

Before conducting testing activities, it's essential to establish certain assumptions to ensure the validity and reliability of the testing process. The following assumptions are made for testing the RESTful Booker API:

- The API documentation accurately reflects the behavior of endpoints.
- The test environment closely mirrors the production environment.

4. Test Deliverables

Test deliverables are provided below

4.1. Before Testing

- **Test plan document:** The scope, objectives, and approach of the testing endeavor are all outlined in the test plan.
- **Test suite:** Test cases illustrate how to run a test, including input data, expected output, and pass/fail criteria.
- **Test design and environment specifications:** The test environment outlines the hardware and software configurations used for testing.

4.2. During Testing

- **Test log**: The test log records each test case's results, including issues and resolutions.
- **Defect report**: A defect report lists testing issues by severity, priority, and reproducibility.
- **Test data**: According to the International Software Testing Qualifications Board (ISTQB), test data is data created or selected to satisfy the execution preconditions and input content required to execute one or more test cases.
- **Test summary report**: The test summary report lists the number of tests run, passed, and failed, as well as open defects.

4.3. After Testing

- **Test completion report**: Covers the testing scope, product quality, and lessons discovered.
- **User acceptance test (UAT) report**: Points to any issues found and fixed.
- **Release notes**: List information about what the release includes. Examples include any new features for development, advancements, or fixes.

5. Test Resources

5.1. Test Team and Responsibilities

Test Manager	Manage the whole project Acquire appropriate resources
Test Administrator	Builds up and ensures test environment and assets are managed and maintained
Tester	Execute the tests, Log results, and Report the defects
SQA members	Take charge of quality assurance and confirm whether the testing process is meeting specified requirements
Developer in Test	Implement the test cases, test program, test suite, etc.

5.2. Test Environment

- Operating System: Windows/Linux with 4GB of RAM
- Server: Nginx
- Browser: Chrome
- Database: MongoDB

5.3. Tools

- Bug Tracking: Jira
- API testing Tool: Postman, Cypress
- Test case creation: Google Sheet

5.4. Resources and Training

- Training sessions on API testing techniques
- Access to relevant documentation and resources

6. Test Schedule

An approximate schedule is shared below. This schedule can be altered depending on the situations:

Task	Time
Test Planning, preparation, and creation	Week 1
Test execution	Week 2-3
Bug fix and retest	Week 4-6
Test summary report preparation	Week 7

7. Risk And Contingencies

This section defines the pre-identified risks and associated mitigation plans. It describes the actions that can be taken when a hazard occurs.

Risk	Mitigation
QA environment down	Keep an alternative environment
Sudden sickness of a team member	Distribute tasks with other team members
Lack of skills	Arrange proper training

8. Suspension and Resumption Requirements

The following criteria can be considered:

- Suspend testing when
    - Critical (show-stopper) bugs are open – (e.g. login issues).

    - An open bug blocks many of the test cases in the test suite.
- Testing can be resumed when
    - Show stopper bug(s) are resolved and test cases can be executed.

9. Approvals

	Project Manager	Test Manager
Name		
Signature		

10. Glossary

API	Application Programming Interface
QA	Quality Assurance
SQA	Software Quality Assurance