

Test Plan (Spotify Web APIs)

Test Plan	1
Objective	2
Scope	2
Inclusions	3
Test Environments	4
Defect Reporting Procedure	5
Test Strategy	6
Test Schedule	7
Test Deliverables.	Error! Bookmark not defined.
Entry and Exit Criteria	7
Entry Criteria:	7
Exit Criteria:	8
Test Execution	8
Entry Criteria:	8
Exit Criteria:	8
Test Closure	8
Entry Criteria:	8
Exit Criteria:	8
Tools	8
Risks and Mitigations	8
Approvals	9

Objective

The goal of this test plan is to ensure the quality, functionality, and reliability of the some of the APIs of Spotify Web

The API is designed to perform various operations which are currently provided by Spotify.

Scope

Scope of Test Plan for Spotify Web APIs:

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's ability to handle unexpected errors gracefully.

4. Performance Testing:

- Assess the API's response time under normal and peak loads to identify potential bottlenecks.
- Measure the API's throughput and scalability to handle concurrent requests.

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. Regression Testing:

- Conduct regression testing after bug fixes or updates to ensure existing functionality remains intact.

10. Edge Case Testing:

- Test extreme and boundary scenarios to identify potential issues.

11. Ad Hoc Testing:

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

12. Usability Testing:

- Evaluate the API's user-friendliness and ease of use from a developer's perspective.

13. Performance Monitoring:

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

It'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

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.

Data Validation:

Test the API'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's response when invalid or malformed requests are made for CRUD operations.

Validate that appropriate error codes and messages are returned.

Performance Testing:

Evaluate the API'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.

Compatibility Testing:

Test the API's CRUD operations on different platforms, browsers, and devices.

Usability Testing:

Evaluate the ease of using CRUD functionalities from a developer's perspective.

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.

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	Devesh
Backend	Sonal
Dev Ops	Prajeeth

Test Strategy

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

While developing test cases, we'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'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 & 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.
- **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	10/04/2024
▪ Test Case Creation	13/04/2024
▪ Test Case Execution	14/04/2024
▪ Summary Reports Submission Date	15/04/2024

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