Apple

One Apple Park Way
Cupertino, CA 95014
(408) 996–1010

iTunes Search API Test Plan

Overview

This test plan is designed to conduct comprehensive testing, which includes analyzing requirements, writing a test plan, creating automated tests, and documenting all discovered issues.

The main goal of this test plan is to ensure high-quality software by thorough testing in accordance with requirements and client expectations.

The key stages of this testing include:

Requirements Analysis: studying and analyzing software requirements to understand functional and non-functional requirements.

Test Plan Writing: developing a detailed test plan that defines the strategy, scope, resources, and schedule for test execution.

Creating Automated Tests: developing and executing automated tests for efficient and repeatable testing.

Documenting Discovered Issues: registering and documenting all discovered issues in the appropriate issue tracking system for further resolution.

This test plan is created to ensure software compliance with requirements, improve its reliability, and ensure user satisfaction.

Date: April 15, 2024

Author: Dariia Medvedieva

Introduction

Purpose

The purpose of this test plan is to outline the strategy, scope, resources, and schedule for conducting functional testing of the iTunes Search API using a Java-based testing framework.

Scope

The scope of this test plan is limited to functional testing of the iTunes Search API module. It includes validating the functionality and behavior of the Search API under various scenarios.

Objectives

- Validate the accuracy and correctness of search results returned by the API;
- Verify the functionality of search parameters;
- Identify and report any functional defects or inconsistencies in the iTunes Search API behavior;
- Ensure that the iTunes Search API meets specified functional requirements and user expectations.

Test Strategy

Approach

The testing approach for the iTunes Search API will primarily involve functional testing, focusing on validating the functionality and behavior of the API endpoints. Test scenarios will be designed to cover various search queries.

Automation

Functional tests will be automated using a Java-based testing framework Rest Assured to improve efficiency and repeatability. Automation scripts will cover critical functionalities.

Test Data

A variety of test data will be used to cover different scenarios, including valid and invalid search queries.

Reporting

Test execution results, including passed tests, failed tests, and any defects identified, will be documented.

Test Schedule

Phases

Requirements Analysis

- Quickly review available requirements or specifications.
- Identify key functionalities, user stories, and acceptance criteria.

Test Planning

- Define high-level test objectives and scope based on requirements.
- Create a basic test plan outlining test approach, strategy, and resources required.
- Allocate time for test case development, execution, and issue documentation.

Test Case Development

- Rapidly draft test cases focusing on critical functionalities and high-priority scenarios.
- Prioritize test cases based on criticality and risk.

Test Execution:

- Execute prioritized test cases, focusing on critical functionalities and high-risk areas.
- Conduct functional testing, emphasizing API endpoints, input validation, and error handling.
- Document test results, including passed tests, failed tests, and any defects found.

Issue Documentation:

- Log all discovered issues or defects in a structured format, including steps to reproduce and severity.
- Prioritize issues based on severity and impact on the system.

GitHub Publishing:

- Organize test plan, test cases, and issue documentation into a repository structure.
- Publish the test plan and test cases to GitHub, ensuring clear documentation and README instructions.

Timeline

- Requirements Analysis: 1 hour
- Test Planning: 2 hours
- Test Case Development: 4 hours
- Test Execution: 8 hours
- Issue Documentation: 3 hours
- GitHub Publishing: 2 hours
- Buffer Time: 4 hours (for unexpected delays or additional tasks)

Milestones

- Requirements Analysis and Test Planning: Complete by 1:00 pm, 04/15/2024
- Test Case Development: Complete by 5:00 pm, 04/15/2024
- Test Execution: Complete by 1:00 am, 04/16/2024
- Issue Documentation: 8:00 am, 04/16/2024
- GitHub Publishing: Complete by 10:00 am, 04/16/2024

Resources

Team Members:

- Dariia Medvedieva, as the QA candidate, will be responsible for all testing activities.

Equipment

Hardware

- Personal laptop with internet access for test planning, development and execution.

Software

- Integrated Development Environment for writing test scripts: IntelliJ IDEA.
- API testing tool for manual testing: Postman.
- GitHub account for version control and project management.
- Web browser for accessing API documentation and testing endpoints.

Budget

No additional budget required as all resources are assumed to be personal or freely available.

Risks and Contingencies

Risks

Limited Timeframe

- Risk: The tight schedule may lead to incomplete testing or rushed execution.
- Impact: Critical functionalities or edge cases may not be adequately tested.
- Likelihood: High, given the short duration of the testing window.
- Mitigation: Prioritize testing of critical functionalities and high-risk areas. Communicate potential risks and limitations to stakeholders.

Technical Dependencies

- Risk: Dependencies on external systems or services may impact test execution.
- Impact: Test environment setup or test execution may be delayed or disrupted.
- Likelihood: Moderate, depending on the stability of external systems.
- Mitigation: Identify and document dependencies early. Have backup plans or alternatives in case of system failures or issues.

Incomplete Requirements

- Risk: Incomplete or unclear requirements may lead to ambiguity in test coverage.
- Impact: Test cases may not accurately reflect the expected behavior of the system.
- Likelihood: Moderate, especially in agile or rapidly changing environments.
- Mitigation: Seek clarification from stakeholders or project managers regarding unclear requirements. Document assumptions and uncertainties in test cases.

Contingencies

Time Management

- Contingency: If testing falls behind schedule, prioritize critical functionalities and high-risk areas for testing.
- Action Plan: Allocate additional time from buffer periods or adjust the scope of testing to focus on essential functionalities.

Technical Failures

- Contingency: In case of technical issues with the test environment or tools, have alternative tools or environments available.
- Action Plan: Quickly troubleshoot technical issues or switch to alternative tools or environments to minimize downtime.

Requirement Clarification

- Contingency: If requirements are unclear or incomplete, seek clarification from stakeholders or project managers.
- Action Plan: Document assumptions and uncertainties in test cases and escalate issues to relevant stakeholders for clarification.

Test Cases and Automated Tests

Test Cases

Search Functionality

- TC001: Verify that the search API returns relevant results for a valid query.
- TC002: Verify that the search API handles requests without required parameters gracefully.
- TC003: Verify that the search API handles invalid queries gracefully.

Search Parameters

- TC004: Verify that the search API supports various parameters.
- TC005: Verify that each search parameter works as expected and influences the search results appropriately.

Error Handling

- TC006: Verify that the search API returns proper error responses for invalid requests.
- TC007: Verify that error responses include meaningful error messages and appropriate HTTP status codes.

Automated Tests Development

Search API Test Suite

- Automate the test cases related to the search API functionality using a Java-based testing framework like Rest Assured.
- Validate that API requests return the expected search results and handle edge cases appropriately.