TEST PLAN
SKYSCANNER API

Author: Amila Sikalo
Date: 06/18/2020

Table of Contents

Introduction

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

1.1 Objectives

Skyscanner is a leading global travel search site, this platform allows users to access millions of travel options at the best prices. Skyscanner API allows developers to easily integrate their application to their platform to get all the available flight data. It employs an API key authentication model, and is a single purpose API, with a RESTful architecture. It supports JSON, XML response format and JSONP, URI Query String/CRUD request format. This API is free to use.

1.2 Team Members

Name	Role
Amila Sikalo	Quality Assurance Engineer

2 Scope

2.1 In Scope

The Skyscanner API *Test Plan* defines the testing approach. The test scope includes the following:

Testing of all functional, application performance, security and use cases requirements;

Quality requirements and fit metrics;

API Testing of GET request.

2.2 Out of Scope

The following are considered out of scope for Skyscanner API Test Plan and testing scope:

POST, PUT and DELETE requests.

3 Assumptions / Risks

3.1 Assumptions

This section lists assumptions that are made specific to this project.

3.2 Risks

The following risks have been identified and the appropriate action identified to mitigate their impact on the project.

#	Risk	Impact	Mitigation Plan
1	The tester will not be able to do Load testing because of rules on Rapidapi.com.	High	The additional subscription plan should be bought to do Load testing.

4 Test Methodology

4.1 Overview

The purpose of the Test Plan is to achieve the following:

- Define testing strategies for each area and sub-area to include all the functional and quality (non-functional) requirements.
- Divide Design Spec into testable areas and sub-areas.
- Define bug-tracking procedures.
- Identify testing risks.
- Identify required resources and related information.
- Provide testing Schedule.

4.2 API Testing

The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces. In API Testing, instead of using standard user inputs(keyboard) and outputs, software is used to send calls to the API, get output, and note down the system's response. API tests are very different from GUI Tests and won't concentrate on the look and feel of an application. It mainly concentrates on the business logic layer of the software architecture. API testing should cover at least following testing methods apart from usual SDLC process:

Discovery testing: The test group should manually execute the set of calls documented in the API like verifying that a specific resource exposed by the API can be listed, created and deleted as appropriate. Example of the test case is given below:

#	Positive Test Case	Expected Result
1	Validate if the status code is 200.	If the user is able to access API, response Code will be 200.

Usability testing: This testing verifies whether the API is functional and user-friendly, and does API integrates well with another platform as well. Example of the test case is given below:

#	Negative Test Case	Expected Result
1	Verify if the user can access API using missing required parameters.	The user shouldn't be able to access API if the request is not functional.

Security testing: This testing includes what type of authentication is required and whether sensitive data is encrypted over HTTP or both. Example of the test case is given below:

#	Negative Test Case	Expected Result
1	Verify if the user can access API using a missing authorization token.	The user shouldn't be able to access API.

Automated testing: API testing should culminate in the creation of a set of scripts or a tool that can be used to execute the API regularly. Example of the test case is given below:

#	Positive Test Case	Expected Result
1	Verify if the response structure is according to data model.	User should be able to check Data Model through scripts.

Documentation: The test team has to make sure that the documentation is adequate and provides enough information to interact with the API. Documentation should be a part of the final deliverable. The documentation regarding Skyscanner API can be found on the following link.

4.3 Smoke Testing

Smoke tests cases verify the major functionality a high level. The objective is to determine if further testing is possible. These test cases should emphasize breadth more than depth. All components should be touched, and every major feature should be tested briefly by the Smoke Test. If any Smoke test case fails, the build is returned to developers untested. The main smoke tests are given in the section above.

4.4 Regression Testing

During the repeated cycles of identifying bugs and taking receipt of new builds (containing bug fix code changes), there are several processes which are common to this phase across all projects. These include the various types of tests: functionality, performance, stress, configuration, etc. There is also the process of communicating results from testing and ensuring that new drops/iterations contain stable fixes (regression). The project should plan for a minimum of 2-3 cycles of testing (drops/iterations of new builds).

4.5 Bug Regression Testing

Every bug that was "Open" during the previous build, but marked as "Fixed, Needs Re-Testing" for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed.

4.6 Critical Path Tests

Critical Path test cases must pass by the end of every 2-3 Build Test Cycles. They do not need to be tested every drop, but must be tested at least once per milestone. Thus, the Critical Path test cases must all be executed at least once during the Iteration cycle, and once during the Final Release cycle.

4.7 Suspension Criteria and Resumption Requirements

- Testing will be suspended on the affected software module when Smoke Test or Critical Path test case bugs are discovered after the 3rd iteration.
- Testing will be suspended if there is critical scope change that impacts the Critical Path.

A bug report should be filed by Development team. After fixing the bug, Development team will follow the drop criteria to provide its latest drop for additional Testing. At that time, adopters will regress the bug, and if passes, continue testing the module.

4.8 Test Completeness

Testing will be considered complete when the following conditions have been met:

Standard Conditions:

- When Adopters and Developers, agree that testing is complete, the app is stable, and agree that the application meets functional requirements.
- Script execution of all test cases in all areas have passed.
- Automated test cases have in all areas have passed.
- All priority 1 and 2 bugs have been resolved and closed.
- NCI approves the test completion
- Each test area has been signed off as completed by the Test Lead.
- 50% of all resolved severity 1 and 2 bugs have been successfully re-regressed as final validation.
- Ad hoc testing in all areas has been completed.

Bug Reporting & Triage Conditions:

- Please add Bug reporting and triage conditions that will be submitted and evaluated to measure the current status.
- Bug find rate indicates a decreasing trend prior to Zero Bug Rate (no new Sev. 1/2/3 bugs found).
- Bug find rate remains at 0 new bugs found (Severity 1/2/3) despite a constant test effort across 3 or more days.
- Bug severity distribution has changed to a steady decrease in Sev. 1 and 2 bugs discovered.
- No 'Must Fix' bugs remaining prior despite sustained testing.

5 Resource & Environment Needs

5.1 Testing Tools

XX bug tracker is used to enter and track all bugs and project issues. The Test Lead is responsible for maintaining the XX database.

5.2 Test Environment

API Testing is different than other software testing types as GUI is not available, and yet you are required to setup initial environment that invokes API with a required set of parameters and then finally examines the test result. Hence, Setting up a testing environment for API testing seems a little complex. Database and server should be configured as per the application requirements. Once the installation is done, the API Function should be called to check whether that API is working.

5.2.1 Hardware

Include the minimum hardware requirements that will be used to test the Application. Testing will have access control to one or more application/database servers separate from any used by non-test members of the project team. Testing will also have access control to an adequate number of variously configured PC workstations to assure testing a range from the minimum to

the recommended client hardware configurations listed in the project's Requirements, Functional Specification and Design Specification documents.

5.2.2 Software

In addition to the application and any other customer specified software, the following list of software should be considered a minimum:

JMeter - version 5.3
Visual Studio Code
Java - version 11.0.7
Node.js

5.3 Bug Severity and Priority Definition

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 to see that a correct severity level is assigned to each bug.

5.3.1 Severity List

The tester entering a bug into XX is also responsible for entering the bug Severity.

Severity ID	Severity	Severity Description
1	Critical	The module/product crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

5.3.2 Priority List

Priority	Priority Level	Priority Description
5	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.
4	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
2	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed.
1	Trivial	Enhancements/ Good to have features incorporated- just are out of the current scope.

5.4 Bug Reporting

The Test Lead will be responsible for managing the bug reporting process. Testing's standard bug reporting tools and processes will be used. XX is the company-wide standard Bug Logging / Tracking tool. Testing and development will enter their data into the XX database.

6 Milestones / Deliverables

6.1 Test Schedule

6.2 Deliverables

Deliverable	For	Date / Milestone
Test Plan	Project Manager; QA Director; Test Team	
Traceability Matrix	Project Manager; QA Director	
Test Results	Project Manager	
Test Status report	QA Manager, QA Director	
Metrics	All team members	