Test Strategy Document

Project Name: Rakuten Inventory API (Get and update)

Prepared By: Krishna Mohan

Version No: 1.0

Date: 27-08-2016

Table of content

Revision History

Name	Date	Reason for change	Version
	27-08-2016	First Draft	V0.1

1 Introduction

1.1 Overview

This is the Test Strategy for inventory API of Rakuten . This document shall be completed and used by the project test team to guide how testing will be managed for this project. The test effort will be prioritized and executed based on the project priorities as defined in the Project Plan and Requirements Specification. This is a living document that may be refined as the project progresses. The QA Manager, Test Team Lead, Product Manager, Project Manager, and Development Manager ETC. shall review and approve the final version of the Test Strategy document.

1.2 Definition and acronyms

- **Project name Rakuten Inventory API**
  Project name and description **Rakuten** Inventory API
- **Ad Hoc Testing**
  Testing contrived for only the specific purpose or problem at hand; testing not carefully planned in advance.
- **Scenario**
  Detailed description (specific instance) of a use case, including rules, exceptions, boundaries, limits, etc.
- **Test Case**
  A specific set of test data along with expected results for a particular test objective.
- **Test Coverage**
  Describes how much of a system has been tested.
- **Test Design**
  Describes how a feature or function shall be tested.
- **Test Plan**
  Test Strategy

2 Scope and limitation

2.1 Scope

Application under test is following two inventory API's

- get
- update

Testing will cover functional testing of above mentioned two api's, Functionality of these api's are detailed in RMS Manual.

Testing of these api's will cover all business use cases, validation, security, integration and performance testing.

This document will be reviewed and verified by manager then testing can be started with following timelines.

Activity	**Timeline**
Test Planning	22 Aug 2016
Test Tool Selection and R&D	23 Aug 2016
Automation Coding	24 Aug 2016- 25 Aug 2016
Code Review and Test Execution	27 Aug 2016

2.2 Limitations and exclusions

These apis will be tested through mocked server so actual behaviour may differ and testing approach will also differ with actual implementation of API server. It is possible that some functionality will be shown to be incorrect; errors of this type will be entered as a defect in the defect tracking system.

3 Testing approach

3.1 Scope

The testing approach for this release shall be done in a fashion that will accommodate the current functionality in get and update API.

Testing will be designed to encompass the following

Use case: These are the most priority scenario and cover highly used positive flows.

Positive case: These scenarios will cover all positive flows. It will check for positive cases by passing all valid values in request and expects response with appropriate response.

Negative Case: These scenarios will cover all negative flows. It will check for negative cases by passing all combination of invalid values of different parameter in request and expects server to handle all scenario gracefully without any breakage in system.

3.2 Test type

3.2.1 Regression

Regression testing will ensure that existing functionality is not affected because of new api, regression

- **Owners :** Krishna Mohan
- **Implementation Approach:** Regression will be performed through already automated test and for some test cases which are not automated will be executed manually.
- **Tools :** Automation Test suite

3.2.2 Functional

System testing is the process of testing an integrated system to verify that it meets specified requirements. This testing will determine if the results generated by information systems and their components are accurate and that the system performs according to specifications.

- **Owner** : Krishna Mohan
- **Implementation Approach**: The objective of functional testing is to verify the correctness of the newly designed items, and their interaction with the existing functions. Testing will focus on functionality of the get and update API.

Testing will be accomplished through an organized testing process that will have repeatable tests. This process will be accomplished by use of the scripts created and designed to match the requirements being developed for business use cases of get and update inventory API

Test scripts will be structured to give a full range of coverage to the converted functions in both a Positive and Negative fashion, simulating what a potentially unfamiliar user might do during use. Positive test cases will reflect that the application functions as expected and described in the Requirements Specification and the Project Plan. Negative test cases are tests that exercise the limits and boundaries outside the expected designs. The results of this testing will give us some idea as to the stability for the application and its components. Additional testing beyond the scripted test may be done where feasible to exercise the application to verify error handling and system recovery due to incorrect data or entry into fields.

- **Tools** : Automated test cases requires java, maven and netty mock server

3.2.3 Security

Both API's expects valid authorization token in request header and without valid token return error message.

- **Owner** : Krishna Mohan
- **Implementation Approach** : test both api with valid and invalid authorization token in request header and for valid token request should be processed but for invalid token system should throw error message in response
- **Tools** : N/A

3.3 Test Coverage

3.3.1 Outline

The coverage for the testing of specific areas of the inventory get and update api is detailed in the matrix below. The test coverage will include known functions that currently exist in current version. The focus of the testing will be on the new features and functionality.

3.3.2 Test Mapping

Test Mapping	**Requirement**	**Test**	**Test Type**
Get Inventory API	Return list of products for shop Url, base Sku or sku for valid request parameters	Test api for different positive combination of parameters and expects list of products	Automated
Get Inventory API	Return appropriate error message for invalid request for invalid request parameters	Test api for different negative combination of parameters and expects appropriate error messages	Automated
Get Inventory API	Api should return response only for the calls with valid authorization token in request headder	Test API with valid authorization token and expected list of products	Automated
Update Inventory API	Update inventory of shop, marketplace, sku and return success messages	Test api by passing different positive combination of parameters and expects success messages and operation Id	Automated
Update Inventory API	Api should return response only for the calls with valid authorization token in request header	Test API with valid authorization token and expected list of products	Automated
Update Inventory API	Update inventory of shop, marketplace, sku and return success messages	Test api by passing different negative combination of parameters and expects failure error messages and operation Id	Automated

3.3.3 Backlog defects

N/A

4 Resources

4.1 Team

Name	Skills
Krishna Mohan	Java, Automation, testing, maven, Mock Server, testNG, web services

4.2 Software

Following software's are required in order to test these api's with mocked server

- JDK 1.7
- Mock Server (mockserver -netty)
- maven
- testNG
- test Server, preferably linux machine
- reportNG

4.3. SCM

- All test scripts and test codes and all document will be committed in Github in the following project path -
  Automation Code: rakutenAPI/inventoryapi-test
  Documentation: rakutenAPI/documentation

- All defects found during testing will be logged in the defect tracking tool
- Any errors found in a test script will be logged in the defect tracking tool

5 Success factor

5.1 Objective

Objective is to test all the business flows are working and for negative scenarios system is able to handle it gracefully and find defects if any and log them and get them fixed so that quality code can be released on production.

5.2 Critical Success Factor

Critical success for this project is based on delivery on time with all script passes completed and all defects closed and regressed.

Internal success will be measured by:

- Completion of application test scripts (written, reviewed, and approved) when scheduled
- Completion of the scheduled test cycles in a timely fashion (as scheduled in the project timeline in the project schedule)

Metrics for this are:

- The completed test scripts with notations concerning defects logged / repaired.

5.3 Assumptions, dependencies and constraints

- Resources must be assigned full time to the test team in order to carry out the intended test cycles.
- Resources other than the Test team will be the Development Staff, Product manager to fix defects if found any.
- Quick and accurate repairs of defects logged into the defect tracking tool.
- Available resources to provide the above in the time line defined in the project plan.