Test design Specification v1.0

Revision History

Date	Version	Description	Author
Mon 15 Apr 2019	v1.0	Release version	Andrej Skeledzija

Table of Contents

1. Introduction

This document determines the product items or aspects that should be checked. Defines techniques and methods that will be applied during testing. Also shows identifiers and short descriptions of each Test Case, Test Flaw Diagram and Pass / Fail criteria.

2. Test Objective

2.1. E-commerce site - UI Testing

Features to be tested:

- User Registration
- User Login
- Items Searching
- Shopping cart
- Checkout process

Approach: Test automation integrated with CI (Selenium/JAVA/Jenkins)
Methods and techniques: Black Box

2.2. Maps service - REST API Testing

Features to be tested:

- CRUD
- Proper, Invalid and incorrect informations / requests
- Error handling (200, 401, 304)

Approach: Test automation integrated with CI (RestAssured/JAVA/Jenkins)
Methods and techniques: Black Box

2.3. Random web app (1000 Visit Users simulation) - Load Testing

Aspect to be tested:

- Load test needs to simulate 1000 users who will visit the homepage in a period of 15s

Approach: Test automation integrated with CI (JMeter / Jenkins)
Methods and techniques: Load performance test

3. Testing Strategy

3.1. Black Box Testing

also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.
In our case we will performed two functional and one non-functional.

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would.
We have decided to perform **Decision table** test design technique for our web application UI testing.

Regarding REST API we decided to use **test analysis** process looking to **test basis** (REST API documentation) to derive test information.

Load test is designed by requirements.

3.1.1. Test automation framework integrated with CI (Selenium/JAVA/Jenkins)

First test object is tested using test framework build by:

- Java - programming language
- Maven - automation tool
- TestNG - testing framework
- Selenium Web Driver - collection of open source APIs
- Jenkins - open source CI server

3.1.2. Test automation framework integrated with CI (RestAssured/JAVA/Jenkins)

Second test object is tested using test framework build by:

- Java - programming language
- Maven - automation tool
- TestNG - testing framework
- RestAssured - Java library that provides a domain-specific language (DSL)
- Jenkins - open source CI server

3.1.3. Test automation framework integrated with CI (JMeter / Jenkins)

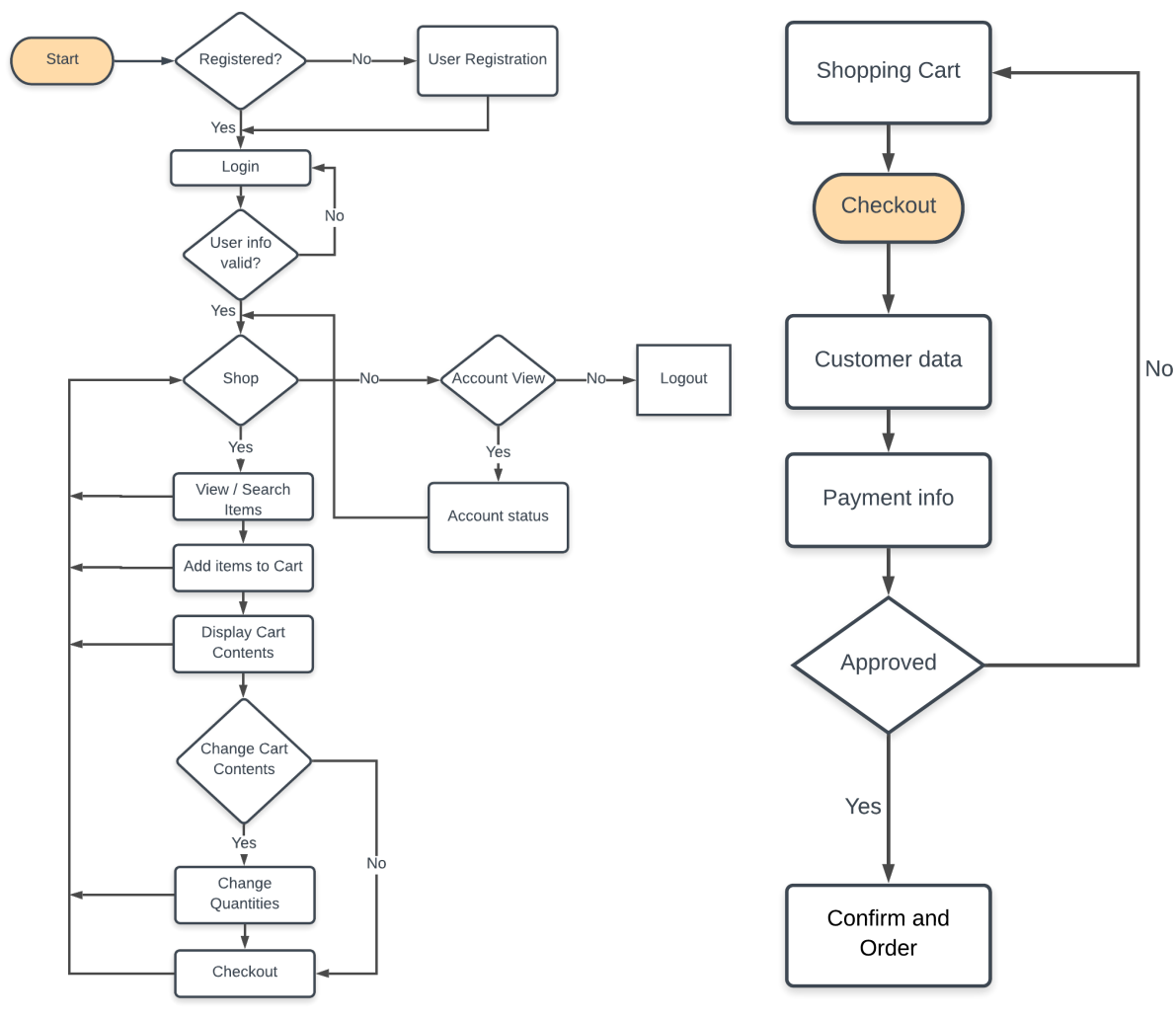
Third test object is tested using test framework build by:

- JMeter - load testing tool
- Java - programming language
- Jenkins - open source CI server

4. Test Cases

4.1. Module 1 - E-commerce site - UI

4.1.1. Test Flaws



Registration, Login, Shop and Checkout scenarios

4.1.2. Decision table - Web Shop App

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Registered?	F	T	T	T	T	T
User info valid?		F	T	T	T	T
Shop?			T	T	F	F
Account View?					T	F
Change Cart contents?			F	T		
Payment approved?			T	T		
Actions						
User registration	T					
Login						
Account Status					T	
Logout						T
View / Search items			T	T		
Add items to Cart			T	T		
Display Cart Contents			T	T		
Change quantities				T		
Checkout			T	T		
Customer info			T	T		
Confirm and Order			T	T		

4.1.2.1. [TD01] Registration scenario

[ID] Name	[TD01] Registration scenario
Summary	New User registration process
Features to be tested	User registration
Approach	Black Box (Manual and Automation)
Test cases	[TC01] Verify registration with already used email [TC02] Verify registration with valid mail
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.1.2.2. [TD02] Login scenario

[ID] Name	[TD02] Login scenario
Summary	User login process
Features to be tested	User login
Approach	Black Box (Manual and Automation)
Test cases	[TC03] Verify login with invalid email and password [TC04] Verify login with valid email or password
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.1.2.3. [TD03] Searching products

[ID] Name	[TD03] Searching products
Summary	Searching products by different categories
Features to be tested	Searching products
Approach	Black Box (Manual and Automation)
Test cases	[TC05] Searching products by Product Name [TC06] Searching products by Model No. [TC07] Searching products by KeyWord
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.1.2.4. [TD04] Shopping Cart

[ID] Name	[TD04] Shopping Cart
Summary	Add / Remove products to/from Shopping Cart
Features to be tested	Shopping Cart
Approach	Black Box (Manual and Automation)
Test cases	[TC08] Add products to Shopping Cart [TC09] Remove products from Shopping Cart
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.1.2.5. [TD05] Checkout scenarios

[ID] Name	[TD05] Checkout scenarios
Summary	Checkout process
Features to be tested	Checkout
Approach	Black Box (Manual and Automation)
Test cases	[TC10] Checkout using Bank Transfer Payment [TC11] Checkout using Cash Payment [TC12] Checkout Credit Card Payment [TC13] Checkout Negative - Payment not approved
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.2. Module 2 - Maps and Navigation services - REST API

4.2.1. APIs to be tested:

- **Datasets API**

  Base URI = https://api.mapbox.com
  Base path = /datasets/v1/

  Create a dataset
  POST
  /datasets/v1/{username}

  Retrieve a dataset
  GET
  /datasets/v1/{username}/{dataset_id}

  Update a dataset
  PATCH
  /datasets/v1/{username}/{dataset_id}

  Delete a dataset
  DELETE
  /datasets/v1/{username}/{dataset_id}

  List datasets
  GET
  /datasets/v1/{username}

- **Tile-query API**

  Base path = /v4/
  GET
  /v4/{map_id}/tilequery/{lon},{lat}.json

The Navigation services are composed of the following APIs:

- **Directions API**

  Base path = /directions/v5/
  GET
  /directions/v5/{profile}/{coordinates}

4.2.1.1. ***[TD07] Datasets APIs***

[ID] Name	[TD07] Datasets APIs
Summary	Implement tests for creating new, modifying existing or deleting data.
Features to be tested	Maps Service - Dataset REST API
Approach	Black Box (Manual and Automation)
Test cases	[TC01] Create a dataset - POST [TC02] Retrieve a dataset - GET [TC03] Update a dataset - PATCH [TC04] Delete a dataset - DELETE [TC05] Check dataset is deleted - GET
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.2.1.2.***[TD08] Tilequery APIs***

[ID] Name	[TD08] Tilequery APIs
Summary	Test for at least three different HTTP response codes in your tests. For example, 200 OK, 401 Unauthorized or 304 Not Modified
Features to be tested	Maps Service - Tile query REST API
Approach	Black Box (Manual and Automation)
Test cases	[TC06] GET - Status code 200 - OK [TC07] GET - Status code 304 - Not Modified [TC08] GET - Status code 401 - Unauthorized
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.2.1.3.*[TD09] Directions APIs*

[ID] Name	[TD09] Directions APIs
Summary	Test sending proper information, invalid information, incorrect format, and other possible edge cases.
Features to be tested	Maps Service - Dataset REST API
Approach	Black Box (Manual and Automation)
Test cases	[TC09] Retrieve directions - GET - 200 OK [TC10] Retrieve directions - GET - 422 InvalidInput [TC11] Retrieve directions - GET - 404 Not Found [TC12] Retrieve directions - GET - 401 Unauthorized
Acceptance criteria	All tests were performed and passed successfully. No High or Highest priority ticket open.

4.3. Module 3 - Performance test - Web App

4.3.1. Load test

Load testing is a type of non-functional testing. A load test is type of software testing which is conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system's behavior under both normal and at peak conditions.
Load testing one among the different kinds of performance testing that determines the performance of the system in real time load conditions. It is basically used to ensure that the application performs satisfactorily when many users try to access or use it at the same time.

4.3.1.1. ***[TD10] Load test***

[ID] Name	[TD10] Load test
Summary	Load test needs to simulate 1000 users who will visit the homepage in a period of 15s. Measure web application response time before and during the test run.
Features to be tested	Server Performance: Response time - time from when a user enters a request until the first character of the response is received.
Approach	Load test (JMeter)
Test cases	[TC01] Retrieve Home Page By 1000 Users and measure server response time
Acceptance criteria	Application response time is in the range of modern day web applications.

4. Pass/Fail Criteria

A test will be considered a failure if the expected result or output is not achieved. A bug report will be filled out for each failure and will be submitted to the development team for correction. After the bug has been fixed, the test case will be repeated.

4.1. Shipping or Live Release

15/04/2019 CEST

4.1.1. Shipping/Live Release Exit Criteria

The product must fully satisfy its release specifications and the user documentation must adequately describe the product's functionality. Both should be ready for use by the end user.

- QA tests the final product version to verify that the product to be released to the general public is of the utmost quality and satisfies original design specifications.
- The product must receive approval from the product team.
- QA and Development must prepare Release Notes.

The product is now ready to ship or published to production environment.