Test Plan

(Version 1.0)

**Version info**

|  |  |  |  |
| --- | --- | --- | --- |
| Version number | Date | Edited by | Comments and changes |
| 1.0 | 27.01.2017 | V.Shevchenko | First version |

**Table of Contents**

1. INTRODUCTION
2. SCOPE
3. ASSUMPTIONS AND RISKS
   1. Assumptions
   2. Risks
4. TESTING STRATEGY
5. ENVIRONMENT
6. TEST DESIGN AND TESTING METHODS
7. TESTS EXIT CRITERIA
8. EXTERNAL LINKS
9. 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.

The object of testing is a webpage - <https://icecat.us/us/notebooks/s> - it’s an online catalog of hardware connected with notebooks. We can do a search for specific products by type of hardware, manufacturer, brand name or any other search criteria. This tool has a frontend based on a webpage (url above) and a backend based on database.

The testing is performed on frontend part only. Main aim of testing is to ensure that product is working properly, performs correct search and inner functions of page are working as it is expected by users.

2.SCOPE

This part of test plan describes functionality that needs to be checked during testing period. At this point we are testing main functionality of website, functionality of links and accordance to designed layout on different devices or screen resolutions.

We must ensure that:

- Search is performed correctly

- Results are shown in appropriate way (including brand info, product image and info, specs, offers, etc)

- Results are sorted according to different sorting options

- Results are automatically updated on scrolling down (AJAX)

- Sponsored products are shown with brand logo and link to partners page

- Non-sponsored products are shown without brand logo

- It’s possible to add to compare sponsored products

- Searching for similar products is performed correctly

- Switching type of result view between grid and line works

- We can expand product info with specs and showing hints on mouseovers

- Performing filtering according to requirements menu affects search results

- We can change page language and it meets translation requirements

- We can change our location and page meets localization requirements

- Menu is working correctly

- Links to service, login and information pages are working correctly

All tests must be performed on different web browsers (both desktop and mobile). Current list for desktop browsers is based on browser market share data that can be checked here - <https://www.w3counter.com/globalstats.php> . A list of mobile devices is based on this link - <https://en.wikipedia.org/wiki/Usage_share_of_web_browsers>. It can be changed according to market changes or some regional demands (e.g. adding UC browser for asian region). Opera browser is not included as it uses same engine as Chrome so differences may be insignificant.

List of devices for testing on mobile must be created within the testing team and also may differ depending on market situation and regional preferences.

Any changes to test scopes may be considered by team and added to this document. Please - update version info section in the header of this document.

3. ASSUMPTIONS AND RISKS

a. Assumptions

* Product is delivered on a working environment
* Test team works only with already set up project
* Team is working with premade SQL database

b. Risks

* Server may not be available as it is located not in testing laboratory
* Localization and internationalization may take time or not be performed correctly
* We are unable to make any performance tests
* Changing of versions on environment is not in control of testing team
* No possibility to check if SQL requests are handled correctly by SQL server

4. TESTING STRATEGY

Product is already set on a working environment. We are skipping Unit and Integration tests as we don’t have any access to inner part of system.   
  
We need to start with Smoke tests to determine if system is available for deeper testing, if it has blockers or other critical bugs that can make any further testing impossible.

Functional testing should be performed according to test cases or user stories that were created before. We can perform ad-hoc testing to have an idea of functionality and to have base for creating test cases. We also can make some exploratory testing based on requirements mentioned in SCOPE section.

For non-functional methods of testing we are going to focus on Internationalization and Localization testing, UI testing (according to designed templates) and Usability testing. All input fields must be checked for SQL, HTML or script injections. We will not perform any tests connected with server performance and data loading. Also we will not perform installation testing.

All fixed bugs must be re-tested after fixing them. It will be good if we will have any possibility of Sanity testing of functions that were touched by a fixed bug. This is vital to make sure all bugs are fixed and fix is not affecting any connected features. Smoke and Regression testing must be performed for each build of a product that is installed on environment to make sure product is working properly, build is using all latest changes and no new bugs appeared.

User acceptance tests must be performed after successful passing of all previously mentioned tests. Main purpose is to be sure that system is ready for operation use. It must be performed by customers to make sure that it meets all their requirements. Need of Alpha and Beta testing stages must be considered with Product owner.

5. ENVIRONMENT

Server-side environment configuration must be as close as a production one.

Here is an example of a list of configurations for user-side environments. It can be changed according to regional or market demands. Test team need to keep in mind that we need to check different screen resolutions and different browsers. We may use PICT or similar tool for creating device configurations

Device: MacBook Pro late 2012 - MacOS 10.11.2 - Safari 10.1.2

Device: PC (Intel i5, 4GB RAM, integrated video) - Windows 10 - Edge 20.10240

Device: PC (Intel i7, 8GB RAM, integrated video) - Windows 10 - Firefox 58.0

Device: iPhone 6 - iOS 10.3.3 - Safari mobile (version changes with OS update)

Device: Samsung S6 - Android 7.0 - Chrome 29.0.1547

6. TEST DESIGN AND TEST METHODS

We are starting with ad-hoc testing to get acquainted with current page and it’s basic functions. In EXTERNAL LINKS section of this document you can find a checklist with functions to be tested. We may write test cases for all of these functions and it will definitely ease our work. But lack of time may prevent us from this.

We are using Grey Box testing as we have partial access to HTML source of page and scripts used on it. We are performing dynamic tests as we are running tests on working environment. At this point we are performing manual testing. Automation may be added on later phases of testing.

As we don’t have any “expected result” yet and if we have no time for preparing user stories or test cases we need to create a checklist. For an “expected result” we need to consider what we are waiting as a standard behaviour from this system. We may check other similar pages or elements to know how they should behave.

Let’s split page into sections:

* Search results with product
* Search field
* Requirements filter block
* Site header with links
* “Hamburger” menu
* Copyright and other information (block under requirements filter)

We will check each section in a separate way. Also we will pay attention on working with page using REST architecture.

Search results with product

Mainly we are going to use cause/effect technique.. We need to check all links, buttons and switches and check if they behave in the way we expect. Sorting must be changed according to the type we choose. We need to check if only sponsored products are shown with a brand logo and it can be added into comparison.

Search field

As search field is an input ve must perform all test techniques that are suitable for it. Mainly we are going to use equivalence partitioning with small adding of boundary values analysis. Here is the list of value types we need to use:

* One word
* Two words
* Three words
* Using “+” symbol for searching exact combination
* Reaction on capital letters
* Numbers
* Words and numbers combination
* Empty enter
* Reaction on spaces
* Reaction on ascii code symbols
* Copy/pasting data and links into field
* Special characters
* Query length limit
* Logical operators
* Security check for SQL, HTML and script injections

For exact values please - see checklist.

As search field has functionality to offer hints on data entry, we must check if it works correctly and can it understand data entered with mistakes. It could be good if we can perform a search using not only english language.

Also we must check behaviour of “search” button and possibility of performing search with simple keypress on enter while search field is in focus.

Unfortunately we don’t have any information about search mechanism or algorythm so we are unable to check if search works according to them.

Requirements filter block

We need to check different configurations and how they affect search result. Number of configurations is really big and we may not be able to check all of them. Especially when we are checking combinations of options. Number of tests should depend on time. We need to create paired tests to cover as much combinations as possible. Link to example of configurations created by PICT is located in EXTERNAL LINKS section of this document. This list is shortened not to take more space in this document. It can be extended by adding more categories and brands.

Price range slider can be checked inside of combinations tests but we need to pay separate attention on it’s behaviour.

Site header with links

Again we are using cause/effect technique. We need if all links are leading to correct sections. Change of language and region must lead to correct interface and unit changes. Logo should be positioned correctly and lead to home page. Share with social networks link should work correctly, same as social network icons near it.

“Hamburger” menu

Button should toggle on expand/collapse of menu. All links must lead to correct pages. Social media icons should lead to Icecat pages on social networks.

Copyright and other information (block under requirements filter)

All links must lead to correct pages. Social media icons should lead to Icecat pages on social networks.

REST testing

We can perform changes to filters and search result by working with url and HTTP methods to send data for queries directly to server. Using instruments like SoapUI or similar can help us to work directly with query and make manual queries without even interacting with user interface.

7. TESTS EXIT CRITERIA

As we may never be sure that software is bug-free, our tests must continue until we have time for them. By the end of testing period we need to be sure that:

* all planned tests have been run (according to checklist)
* there are NO critical or high severity defects that are left
* all found bugs are fixed and retested

On lack of time team must determine list of functions priority and perform testing according to this list.

8. EXTERNAL LINKS

Check list: <https://docs.google.com/spreadsheets/d/e/2PACX-1vRyJAgRWBD7scPAGyV4oDyesYMg7sDUnWeLTEY6Z-mj9ZsCKSDUD3LC2-dcjhC3zYWjkpJG2P2f2mDE/pubhtml>

Pairwise testing list:

<https://docs.google.com/spreadsheets/d/e/2PACX-1vSFuvIkEaFZ_nMTBzFydGberUHOsQdnES6hUzubG59s8tJFNCR9j7H-1yvrR1Kup1gjf_7JshSoKWZI/pubhtml>

Bug report examples:

<https://docs.google.com/spreadsheets/d/e/2PACX-1vSQkKi9Yvy4E4f65OttuGGTXoXsGc46sN6QoPsdo0PKrWhvouesLi0Usy1Yf7JOcv5ooxOusLWseZX6/pubhtml>