Automation Test Plan

Created by: Shavaiz Safdar & Yohan Desanti

Date: 11/25/2015

The purpose of this document is to outline the approach that will be used to create the automation framework for mobile web. The idea behind automating the mobile web test cases is to create a robust automation framework which would help in effective utilization of the time spent on mobile web regression testing.

Scope:

The primary focus of automating the test case is to automate such test cases which are reusable over the time. All the important test cases with very high to high priorities are going to be given a priority for automation over the less important test cases with medium and low priorities. In addition, we will be using a module based testing framework in the beginning in which, each set of test case is going to independent from the other set so the change in one set will not affect the other.

Test Strategy:

Selecting the right automation framework is very important to create the automation test scripts efficiently. There are number of automation frameworks available with each having its own pros and cons, however, we need to create a framework that is best suitable for our project. In the beginning, we will be creating scripts for some very high and high priority test cases. A test factory approach will be used to create individual objects pages for each mobile web page. It is going to be one time effort which will significantly reduce the script creation time for future automation. We will use Github (https://github.com/isqacostarica/MobileWebAutomation/) as a central repository to store the test assets so others can have access to the test scripts with ease.

Based on the test cases list that has been vetted by the team, it will take us around 25-30 working days to complete the automation (Approximately on December 30th). This time includes creating the actual test scripts, object pages, reporting, identifying date used for the testing, data reading and writing functions, and exception handling. This time frame has been calculated based number of test cases, test case complexity and the resources available.

Tools:

We are planning to use Eclipse IDE to create the test scripts. The test scripts are going to be developed with the help of Selenium Webdriver and TestNG using Java. Selenium is known to be very reliable open source tool for automation therefore we can take advantage of it and its various plugins to develop robust test scripts. It supports multiple operating systems as well as variety of browser. It will allow us to run multiple test scripts at the same time. Chrome emulator will be used to run the test scrips which will allow us to test on several different device. It provides many devices to emulate with help of UA (user agent) which sends request to the server in the form an actual devices and get the response. Some of the popular devices that chrome emulate offers are as follows:

1. Apple
   iPhone 4, 5, 6, 6 plus
2. Samsung
   Note 3, Note 2, S3, S III and S4
3. Google
   Nexus 4, 5, 6, 7, 10

We can initiate each test case on different devices at the same time which will save a significant amount of time or testing. In addition to that, Maven will be used to handle the dependencies and to make the scripts easily shareable among the team members. TestNG will be also used to set test priorities and label the test in different groups and set priorities.

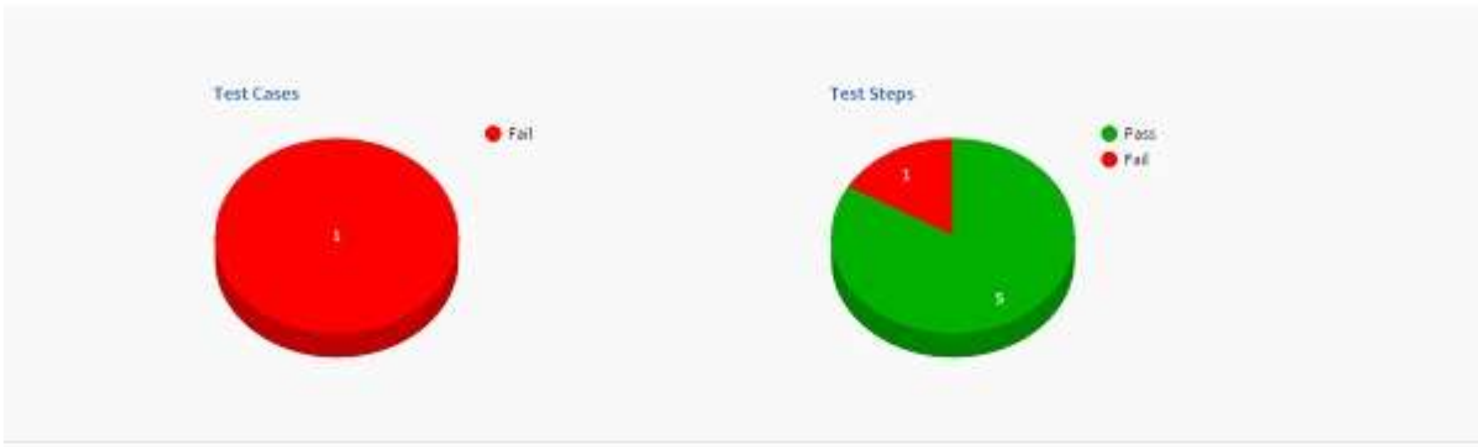
Benefits:

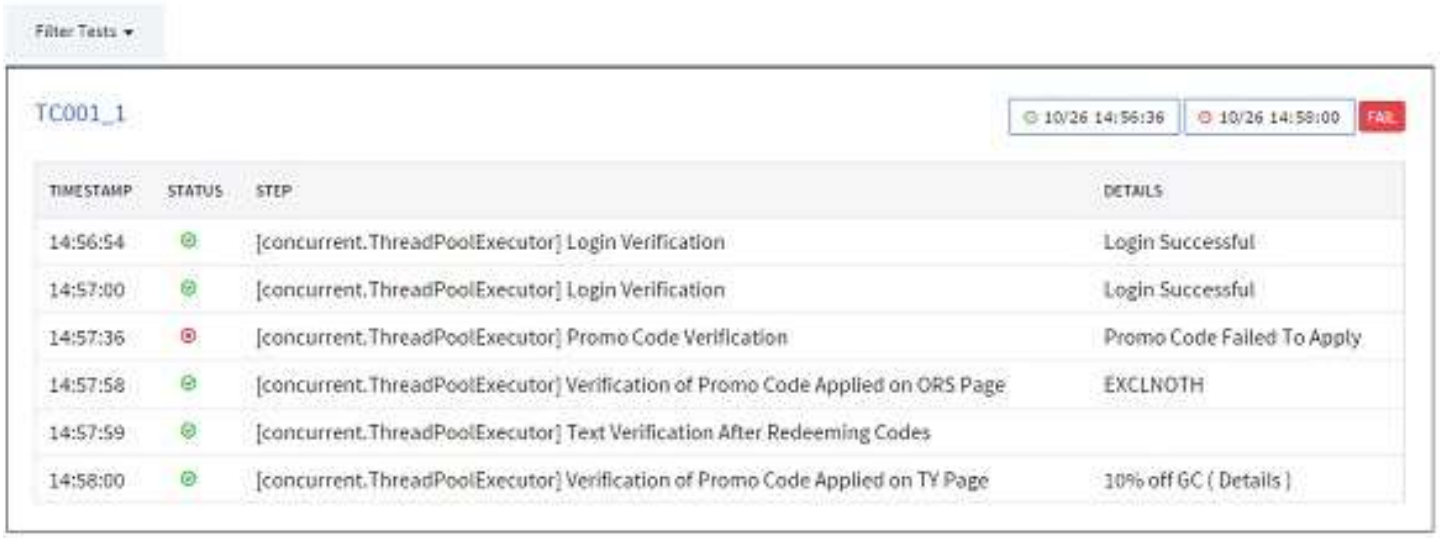
Some of the benefits that the automation will be providing are:

- **Time saving.** Automated tests can be running faster than manual tests, so it will provide reduction on the time needed to run a specific set of tests.
- **Better Resources Management**. Since the automation requires a minimal human intervention, during the execution resources can be assigned to other important manual tasks (new features testing, defects creation/validation, non-automated tests execution, etc.)
- **Multi-Device/OS Testing**. Using the Chrome emulator allows the automation to run against several combinations of device and operating system version.
- **Cross-Platform execution.** Since the automation is based on Java technologies, this can be executed on any system which allows a Java Virtual Machine and the Chrome browser.
- **Adaptable Execution.** Thanks to TestNG, the automation is able to run selected groups of tests making it more adaptable to what is really needed in the moment. (i.e: SmokeTest, FullRegression, CheckoutOnly, etc.)
- **Cristal Clear Reports.** The automation provides a very easy to read HTML Report with the execution results. This allows both technical and non-technical people to understand how the last run was.

Reports/results:

We are planning to use TestNG to display test execution results. TestNG allows to create custom HTML reports that can be easily understand by a non-technical person. These customs reports are very simplified, user friendly and easily manageable. As shown in the attachment below, the custom test result report will show the detailed information of the executed test steps, the number of pass and fail test cases and the time it took to complete the test.




Filter Tests ▾

TC001_1 — 10/26 14:56:36 — 10/26 14:58:00 — FAIL

TIMESTAMP	STATUS	STEP	DETAILS
14:56:54	✓	[concurrent.ThreadPoolExecutor] Login Verification	Login Successful
14:57:00	✓	[concurrent.ThreadPoolExecutor] Login Verification	Login Successful
14:57:36	✗	[concurrent.ThreadPoolExecutor] Promo Code Verification	Promo Code Failed To Apply
14:57:58	✓	[concurrent.ThreadPoolExecutor] Verification of Promo Code Applied on ORS Page	EXCLNOTH
14:57:59	✓	[concurrent.ThreadPoolExecutor] Text Verification After Redeeming Codes	
14:58:00	✓	[concurrent.ThreadPoolExecutor] Verification of Promo Code Applied on TY Page	10% off GC (Details)

Deliverables:

1. **Automation test plan**
2. **HomePage script**
3. **Product page script**
4. **Product list page script**
5. **Search script**
6. **Account page script**
7. **Login script**
8. **AU-checkout script**
9. **PAU-checkout script**
10. **Daily Mark down script**
11. **PayPal checkout script**
12. **PDP ADQ script**
13. **PPD Monograming**
14. **OAP script**
15. **ODS script**
16. **Shopping bag script**
17. **Object Pages**
18. **Confluence Documentation**