**Test Documentation**

**Brief Description about the project:**

The project ***‘Home-opoly’*** is a web based application that will be written in PHP with a MySQL database at the backend. There are two type of users who would interact with the application- a buyer- who initiates a request from the house, the seller - who authorizes the purchase of the house, he/she or the agent are also manager that can review the status of their houses.

**Definitions:**

As the team planned to start testing, team members had a discussion to use the single Test Case Template. Below is the Test case template with definitions for each term which we will be using in our testing.

|  |  |
| --- | --- |
| Test case ID: |  |
| Test items: |  |
| Test priority : |  |
| Dependencies : |  |
| Preconditions: |  |
| input data: |  |
| Test steps: |  |
| Post conditions: |  |
| Expected output: |  |
| Actual output: |  |
| Pass or Fail: |  |
| Bug id/link: |  |
| Additional notes: |  |
| Is Regression |  |
| Regression Result |  |
| Regression Bug id/link. |  |

**Test case ID:** Describes the ID number for each test case. Ex 1, 2,…

**Test items:** Describes what item needs to be tested.

**Test priority:** Defines the Test Priority. In our case we use High, Medium and Low.

**Dependencies:** Any Dependencies for the present test case should be defined. For ex. Login depends on Signup or House creation depends on Signup and Login.

**Preconditions:** Defines the Preconditions which need to be fulfilled for this test case to pass/complete.

**Post conditions:** Defines Post conditions which should be satisfied once this test case is complete.

**Expected output:** Defines the correct output format.

**Actual output:** Results from the current test case.

**Pass or Fail:** Defines whether the test case is pass or Fail.

**Bug id/link:** Any bug should be notified here with the Bug id or link.

**Is Regression:** Describes whether the test case needs Regression testing or not. If yes the Yes else no.

**Organization:**

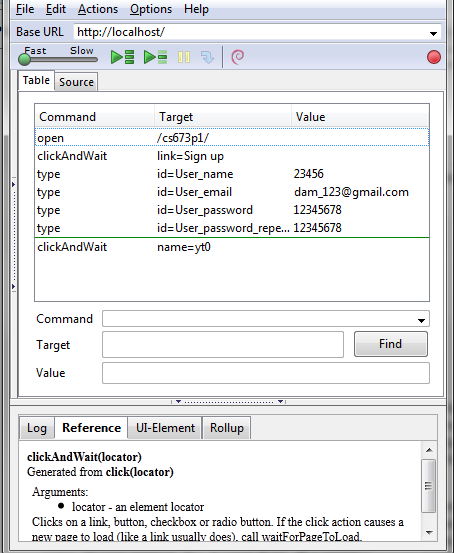
Some of the tools were required for testing the website Home-o-poly. Implementers had to use PHP unit as a testing tool for unit testing since we use PHP. In the later iterations the system testing was done using a tool called Selenium. Also we have used a tool called YSlow which helps us with the performance testing. Following describes about these tools.

**PHP Unit:** The PHP Unit testing procedure is similar to the JavaUnit testing. Each function is tested by checking the required output based on an input. This provides micro-level testing of the project to see that each component is working well, before we can proceed on to integration testing as was required by the projects.

**Selenium:** Selenium is a portable software testing [framework](http://en.wikipedia.org/wiki/Software_framework) for [web applications](http://en.wikipedia.org/wiki/Web_application). Selenium provides a record or playback tool for authoring tests without learning a test [scripting language](http://en.wikipedia.org/wiki/Scripting_language). Tests can then be run against most modern [web browsers](http://en.wikipedia.org/wiki/Web_browser).

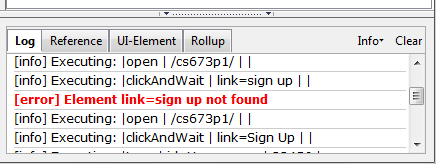
Selenium IDE is a complete [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for Selenium tests. It is implemented as a [Firefox extension](http://en.wikipedia.org/wiki/Add-on_%28Mozilla%29), and allows recording, editing, and debugging tests. Scripts may be automatically recorded and edited manually providing [auto completion](http://en.wikipedia.org/wiki/Autocomplete) support and the ability to move commands around quickly. Scripts are recorded in *Selenese*, a special test scripting language for Selenium.

Sample Scripts from our tests.

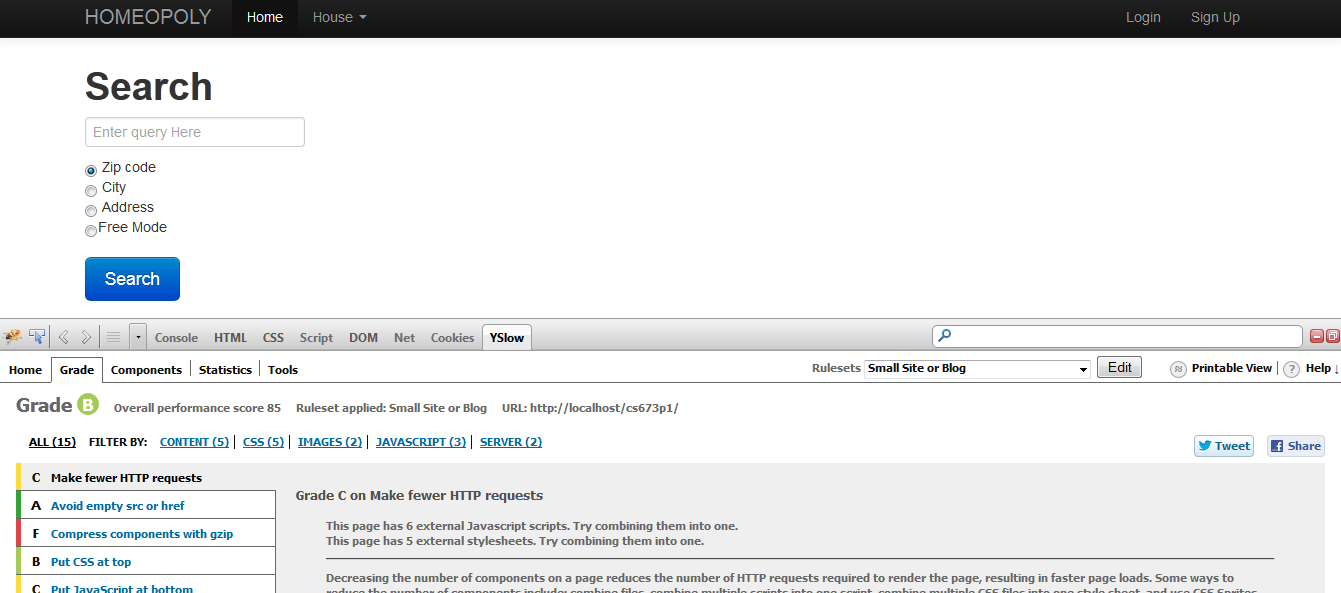


Above is the script produced by Selenium? The Command, Target, and Value entry fields display the currently selected command along with its parameters.

Selenium also provides us with the Log pane which displays error messages and information messages showing the progress. This is shown below.



**YSlow:** YSlow is a tool which helps in testing the performance of the website. YSlow is a freely available tool. If the performance of a web page is slow, YSlow also analyzes why it is slow based on the Yahoo’s rules for the high performance websites. YSlow runs on Firefox, Chrome, Opera and many other platforms. Another plugin called Firebug needs to be installed in order for YSlow to work. YSlow also generates grades for the Website depending on its performance and the components we have used to build a website. YSlow also provides the user few advices on how to improve the performance of the website analyzing our code. Below is a screenshot displaying the performance of our website Home-o-poly?



**The Testing process:**

As was mentioned in the SPMP previously, this project is being undertaken as a part of a semester long group project with 6 members. The purpose of the testing phase is to provide a high quality end product to the user as well as understand to a decent detail the entire testing process. The following gives details regarding the process:

**Organization:**

The team had a testing leader who predominantly decide the testing tools and provided support to various members of the group. It was assured that everyone participated in the process and therefore the entire code database was divided into six parts and each member assigned a specific part. The entire process lasted for around two weeks and had the following phases:

1. *Testing Phase*

Everyone tested and reported issues using Github as well as documented using the Template provided by the instructor. No issues were solved in this phase.

1. *Solution phase*

This phase was used to solve issues discovered during the testing. Each implementer was supposed to solve issues regarding their functionality. This made the process quick and efficient. Regression testing was done to assure quality.

**Software Testing:**

Software testing is a process of validating and verifying that a product meet the requirements, works as expected and satisfies the needs of the stakeholders. Testing can be divided into Unit, Integration, System, and Regression testing. Our team did follow the same phases of testing.

**Types of Testing:**

**Unit Testing:**

Unit testing is a method where the individual units of source code are tested to make sure each of them works fine. The main advantage of Unit testing is it helps us finding bugs early and fixing it early as well. Unit testing allows the programmer to [refactor](http://en.wikipedia.org/wiki/Refactoring) code at a later date, and make sure the module still works correctly. We have done the Unit testing using a tool called PHP Unit.

**Component Testing:**

Each implemented and working component of the website was tested. For example, during the first iteration the Login page and the database were ready. These components were tested individually. Likewise tested all the components individually which were ready.

**Integration testing:**

Integration testing is the phase in [software testing](http://en.wikipedia.org/wiki/Software_testing) in which individual software modules are combined and tested as a group. For example, during the first iteration we could test the login page. During the second iteration we could see the houses which user has on his account by giving the user id on the URL (using query string). Both the test cases were passed but we did not have any logic to display the houses as soon as user logs in. Once we had the code for displaying the details from the DB we tested the entire process to check whether it has integrated properly or not. Test cases can be found on the below link. Test cases 7 and 12 in the document *‘Test case document\_Iteration1\_Sowmya’* is for testing login and checking the user details. Test case TID11 in the document *‘TestCase\_Master\_Sowmya’* gives a detailed report for the integration testing.

<https://drive.google.com/a/bu.edu/folderview?id=0B2Pst2Ik8HkvYVpaMWFrT19YdUk&usp=sharing>

**System Testing:**

**System testing** of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified [requirements](http://en.wikipedia.org/wiki/Requirements). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. There are many testing techniques that falls under system testing. We have used Black box testing which does not bother about the inner code and expects the functionalities required by the customer. We have also used Domain testing technique for testing the User Signup page. A report regarding the Domain testing is documented and uploaded in the Google drive. This can be found here at *‘Domain testing\_Sowmya’* in the below link. Also test cases have been documented and uploaded on the Google drive which can be accessed through the below link.

<https://drive.google.com/a/bu.edu/folderview?id=0B2Pst2Ik8HkvYVpaMWFrT19YdUk&usp=sharing>

**Regression Testing:**

**Regression testing** is any type of [software testing](http://en.wikipedia.org/wiki/Software_testing) that seeks to uncover new [software bugs](http://en.wikipedia.org/wiki/Software_bug), or [*regressions*](http://en.wikipedia.org/wiki/Software_regression), in existing [functional](http://en.wikipedia.org/wiki/Functional_testing) and [non-functional](http://en.wikipedia.org/wiki/Non-functional_testing) areas of a system after changes, such as enhancements, [patches](http://en.wikipedia.org/wiki/Patch_%28computing%29) or [configuration](http://en.wikipedia.org/wiki/Configuration_file) changes, have been made to them. In our project the Regression testing is under progress and should be done with it in couple of days. Regression testing details can also be found on the link given above.

**Statistics:**

Given below are some of the statistics regarding the testing process.

**Logistical stats:**

|  |  |
| --- | --- |
| **Statistic** | **Details** |
| Testing phase duration | 7 days |
| Resolution phase duration | 7 days |
| People involved | 6 |
| Time spent in both phases | 50 hours |

**Technical stats:**

|  |  |
| --- | --- |
| **Statistics** | **Details** |
| Approximate number of functionality tested | 6 |
| Lines of codes reviewed (approx.) | 1000 |
| Issues reported | 20 |
| Issues Resolved | 20 |
| Regression testing issues | 2 |