pylinkvalidator
Test Report : newAGEtech, Group H

Genevieve Okon (Okong), Abraham Omorogbe(Omorogoa),
Eric Le Forti(Leforte)

November 27, 2015

Contents

List of Tables

List of Figures

Revision History

Revision	Revision Date	Description of Change	Author
1	20-10-15	Initiate Test Plan Document and Introduction	Eric Le Fort
2	26-11-2015	Finalize Outline	Abraham Omorogbe
3	26-11-2015	Functional system tests	Abraham Omorogbe
4	26-11-2015	Functional system tests	Eric Le Fort
5	26-11-2015	Non-Functional system tests	Eric Le Fort
6	26-11-2015	Usability Testing	Eric Le Fort
7	26-11-2015	Requirements Traceability	Eric Le Fort
8	27-11-2015	Testing Summary	Genevieve Okon
9	27-11-2015	Code Coverage	Genevieve Okon

Table 1: Revision History

1 Introduction

1.1 Objective

The purpose of this report is to specify the methodology of testing to be used for Pylinkvalidator in detail. Every test case will be accompanied by a short description to convey the reason each test was written as well as a breakdown of expected results as well as whether those results were achieved or not. Following that section the document will trace the tests back to the requirements and then provide a more general summary of the results of testing.

1.2 Approach

The methodology to be used for testing will involve a succinct set of tests to prove each requirement is fully functional and performing at an acceptable level. These tests will cover white-boxed boundary cases, cases dealing with extremes as well as standard cases.

Certain tests, such as those concerning usability or involving acquiring user input, are tested much more straightforwardly using manual methods. Therefore, these sorts of tests will be performed in a manual manner. All other tests will be conducted using automated testing utilizing a testing suite known as PyUnit.

1.3 Tables of Acronyms, Abbreviations & Definitions

Term	Meaning
html	Hypertext Markup Language

Table 2: Acronyms & Abbreviations

Term	Definition
PyUnit	A widely accepted testing suite to be used with the Python programming language.
$\{\emptyset\}$	Denotes an empty set, not to be confused with $\{0\}$ which is a set containing only 0.
Beautiful Soup	An existing framework that breaks a webpage down into its components.
HTML Error Code	A three digit number that corresponds to various states of a website.

Table 3: Definitions

2 Functional System Tests

2.1 F1: Exact String Searching

Process:

1. Receive String to be parsed through and a query to search for.
2. Search through that String for the query.
3. Check that the results of the search match those that should be returned.

The queries used for each input String will be as follows:

- hello
- LASDGLKGSVLIUGAEOUGSVLUHwe;ofrw.k?bwri;hqf.IBA LIU GqleiugwKUGwrliugwrgOUGFW ;OURW;U
- Eric
- my name is Eric!
- my name is Eric

The input Strings used will be as follows:

1. "Hello world!"
2. ""
3. "Hello. My name is Eric. I am writing this simple test to check to see how well my parsing algorithm is performing. hello again, don't forget my name: Eric. That is all."

Input	Expected Result	Actual Result
1	{0}	{0}
	{∅}	{∅}
	{∅}	{∅}
	{∅}	{∅}
	{∅}	{∅}
2	{∅}	{∅}
	{∅}	{∅}
	{∅}	{∅}
	{∅}	{∅}
	{∅}	{∅}
3	{0, 114}	{0, 114}
	{∅}	{∅}
	{18, 149}	{18, 149}
	{∅}	{∅}
	{7}	{7}

Table 4: F1 Tests

2.2 F2: Similar String Searching

The tests that were run during F1 will be ran again in the same fashion using a proximity of 0. This forces the same functionality as performing an exact String search and so the results shall be the same as above.

Process:

1. Receive String to be parsed through and a query to search for.
2. Search through that String for the query.
3. Check that the results of the search match those that should be returned.

H Input	Expected Result	Actual Result
String: "Here comes my hero" Query: "HERE" proximity: 1	{0, 14}	{0, 14}
String: "Some text, hello" Query: "hello" proximity: 2	{11}	{11}
String: "hehhhehellp" Query: "hello" proximity: 1	{6}	{6}
String: ". \n .. \n q1/.SQL \n q2.SQL \n fileResult.txt" Query: "fileResults" proximity: 1	{28}	{28}
String: ". \n .. \n q1/.SQL \n q2.SQL \n fileResult.txt" Query: "fileResults" proximity:	{32}	{∅}

Table 5: F2 Test Cases

2.3 F3: BFS Execution

This test will be created in an environment to effectively test depth of a BFS. The depth folder is structure in such a way the root.html, link to the 1.html and 1.html links to 2.html and 2,html links to 3.html and so on. With this structure you can test the depth 0 -5

H Input	Expected Results	Actual Results
url: "/depth/root.html" depth: "0"	[u'root.html']	[u'root.html']
url: "/depth/root.html" depth: "1"	[u'/root.html', u'/0.html']	[u'/root.html', u'/0.html']
url: "/depth/root.html" depth: "2"	[u'/root.html', u'/0.html', u'/1.html']	[u'/root.html', u'/0.html', u'/1.html']
url: "/depth/root.html" depth: "3"	[u'/root.html', u'/0.html', u'/1.html', u'/2.html']	[u'/root.html', u'/0.html', u'/1.html', u'/2.html']
url: "/depth/root.html" depth: "4"	[u'/root.html', u'/0.html', u'/1.html', u'/2.html', u'/3.html']	[u'/root.html', u'/0.html', u'/1.html', u'/2.html', u'/3.html']
url: "/depth/root.html" depth: "5"	[u'/root.html', u'/0.html', u'/1.html', u'/2.html', u'/3.html', u'/4.html']	[u'/root.html', u'/0.html', u'/1.html', u'/2.html', u'/3.html', u'/4.html']

Table 6: F3 Test Cases

2.4 F4: Program Navigation

Process:
This test will be conducted by manually entering each possible value the program prompts for. Once entering a number, the program shall use that option as appropriate. Values that are not within the acceptable set of options shall return a result specifying that the value entered is invalid.

H Input	Expected Results	Actual Results
value: 1	Download Resources	Download Resources
value: 2	Check for Errors	Check for Errors
value: 3	Search for Query	Search for Query
url: "/depth/root.html" value: 4	Just Crawl	Just Crawl
url: "/depth/root.html" value: 100	Incorrect input.	Incorrect input.

Table 7: F4: Program Navigation

2.5 F5: Grab Links

Process:
The program is going to take all the links on any page

H Input	Expected Results	Actual Results
input HTML: $< html >$ $< body >$ $< a href = "\#" > Testhash < /a >$ $< a name = "hello" > Testname < /a >$ $< a href = "a.html" > TestA < /a >$ $< a href = "sub/b.html" > TestB < /a >$ $< a href = "/c.html" > TestC < /a >$ $< a href = "d.html" > TestD < /a >$ $< a href = "//www.perdu.com" > TestExternal < /a >$ $< /body > < /html >$	[None, u'a.html', u'sub/b.html', u'/c.html', u'd.html', u'//www.perdu.com']	[None, u'a.html', u'sub/b.html', u'/c.html', u'd.html', u'//www.perdu.com']

Table 8: F5 Test Cases

2.6 F6: HTML Correction

Process:
The program is going to correct any url that is missing a HTTP SCHEME such as HTTP,www or the base url. So for example, if a url is just "/about", the program will append "/about" with the base url (e.g. http://www.google.ca/about"). And if a url is missing a www, or HTTP:// ,this should fix that issue.

H Input	Results	Status
url: "www.example.com"	"http://www.example.com/"	PASS
url: "//www.example.com"	"http://www.example.com/"	PASS
url: "http://www.example.com"	"http://www.example.com/"	PASS
url: "www.example.com/"	"http://www.example.com/"	PASS
url: "//www.example.com/"	"http://www.example.com/"	PASS
url: "http://www.example.com/"	"http://www.example.com/"	PASS
base: "https://www.example.com /hello/index.html"		
url: "//www.example2.com/test.js"	"http://www.example2.com/test.js"	PASS
base: "https://www.example.com /hello/index.html"		
url: "/hello2/test.html"	"http://www.example.com /hello2/test.html"	PASS
base: "https://www.example.com/hello/index.html"		
url: "test.html"	"http://www.example.com/hello/test.html"	PASS
base: "https://www.example.com/hello/index.html"		
url: "../test.html"	"https://www.example.com/test.html"	PASS

Table 9: F6 Test Cases

2.7 F7: Error Notifications

Process:
This test will be conducted to check errors and successes on the test html pages. Two HTML error

codes that we will be testing for are a 404 error, which means the page doesn't exist, and 200, which means it exist and it is healthy and has no errors.

Input	Results	Status
url: "/does_not_exist.html"	Error code: 404	PASS
url: "/index.html"	Status code: 200	PASS

Table 10: F7 Test Cases

3 Non-Functional Tests

3.1 Usability

U1: A user that knows how to execute a Python application should be able to operate this program without difficulty.

3.2 Performance

P1: All String searching operations shall be completed within a second assuming the input String isn't unreasonably large.

P2: Grabbing links from a page using Beautiful Soup shall be completed within a second assuming the webpage isn't unreasonably large.

3.3 Robustness

Implicitly included within the list of functional test cases since extreme cases as well as boundary cases will be performed.

4 Traceability to Requirements

Requirement Designation	Associated Requirement
F1	Functional requirements 5 and 7.
F2	Functional requirements 5 and 7.
F3	Functional requirements 3 and 5
F4	Functional requirements 1, 2, 3, 4 and 7
F5	Functional requirements 5
F6	Functional requirements 1, 4, 5
F7	Functional requirements 1
U1	Non-Functional requirements:
P1	Non-Functional requirement: Performance
P2	Non-Functional requirement: Performance

Table 11: Traceability to Requirements

5 Testing Summary

5.1 Usability Testing

For our Usability testing we had a few participants execute common actions within the system. We found that while the log-in process was received quite well by all the participants, there was sort of an issue with the command-line interface. For the downloading of the Webcrawler program files, it wasn't hard for the participants. Although they felt the number of files needed to be downloaded for the program was a bit too much and the initialization of the command line was a bit tedious. Inputting the option for the web crawler program wasn't deemed as hard by most users.

5.2 Performance Testing

We will want to test the number of active users that can be on the system at one time even though each user of the program is independent of the other. We will also want to test the number of active users that can be downloading resources at the same time from the same website. These tests will also be run using the same website on multiple browsers to test compatibility. To test the performance of a single user, we manually executed tasks that the user normally would.

5.3 Security

Security concerns have already been validated. Every user of the program has a different command line interface, no additional security is required. In light of this, we focused on getting feedback in other areas.

5.4 Response to feedback

A User Manual will need to be created in order to make learning how to use this program on the command line much easier. One participant expressed during a task where they were to check for errors that it was slightly difficult. This was due to the user not having the right installations present on the user's system. The solution to this will be in the user manual. Participants were generally pleased with the way the program worked.

5.5 Changes Due to Testing

The interface for saving html links needs some refinement. There were some changes implemented, however more refinement will occur during future versions.

6 Code Coverage

Requirement	Positive TC Result	Negative TC Result
Locate Information and related websites	Pass	Pass
Automatically download resources from a web page.	Pass	Pass
View list of status codes from website	Pass	Pass
Specify website to crawl	Pass	Pass
Audit Content on companies website	Pass	Pass
Map website Depth	Pass	Pass
Return website status code	Pass	Pass

Table 12: Code Coverage

We have successfully ran our tests. From the tests we conducted, a proficient coverage rate has been achieved. A few of our methods have never been called however, it is not crucial that these methods be tested. This report shows that code coverage is satisfactory since the amount of testing necessary depends on a number of factors, and all of those factors have in face been achieved. These factors in the case of this program include algorithmic implementations, use of external libraries and interface robustness. Since all of these factors have been tested thoroughly, there is no need for anymore coverage.