pylinkvalidator
Test Report : newAGEtech, Group H

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

November 27, 2015

Contents

Revision History

Revision	Revision Date	Description of Change	Author
1	20-10-15	Initiate Test Plan Document	Eric Le Fort

Table 1: Revision History

List of Tables

List of Figures

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

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.

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} {∅} {18, 149} {∅} {7}	{0, 114} {∅} {18, 149} {∅} {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.

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

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

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

acceptable set of options shall return a result specifying that the value entered is invalid.

2.5 F5: Grab Links

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

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

Input	Expected Results	Actual Results
input HTML: $<html><body>$ $Testhash$ $Testname$ $TestA$ $TestB$ $TestC$ $TestD$ $TestExternal$ $</body></html>$	[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 is a url is just /about, it program will append /about with the base url. And is a url is missing a www, or HTTP:// ,this should fix that issue..

2.7 F7: Error Notifications

Process:
This test will be conducted to check errors and success on my test html pages. 404 error means the page doesn't exist and 200 means it exist and it is healthy and has no errors.

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

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.

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

Table 10: F7 Test Cases

3.2 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 Code Coverage

5.2 Testing Results

5.3 Changes Due to Testing