pylinkvalidator
Test Report : newAGEtech, Group H

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

November 26, 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

//TODO fill in test cases //TODO explain the structure of the website from the starting page to be tested.

Input	Expected Results	Actual Results

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.

Results:
//TODO fill in the results

2.5 F5: Grab Links

//TODO fill in test cases

Input	Expected Results	Actual Results

Table 7: F5 Test Cases

2.6 F6: HTML Correction

//TODO fill in test cases

Input	Expected Results	Actual Results

Table 8: F6 Test Cases

2.7 F7: Error Notifications

//TODO fill in test cases

Input	Expected Results	Actual Results

Table 9: 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 10: Traceability to Requirements

//TODO ensure above table is fully completed after completion of test case specifications.

5 Testing Summary

5.1 Code Coverage

5.2 Testing Results

5.3 Changes Due to Testing