# pylinkvalidator
# Problem Statement : newAGEtech, Group H

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

October 23, 2015

## 1 Tested Properties

### 1.1 Summary

These properties will be a succinct list of the aspects of the software we plan to test. The verification of the properties will be must useful in validating most non-functional requirements. A list of the properties, the explanation of each property and the different techniques we will implement to test the properties are shown below:

The table 1 is a Test Properties List.

| Test Components | Description | Method Used |
|---|---|---|
| Ease of Use | Generally difficulty and average time to input a query | Functional System Testing |
| Accuracy | Verify that specific output from the program is correct and accurate | Dynamic Testing |
| Speed | Verify the average speed of each query | Structural System Testing |
| Fault-Tolerance | Verify the program handle common users errors | Functional System testing |
| Adaptability | Verify program runs on multiple operating systems | Manual Testing |
| Legal Requirements | Verify web crawler cannot be used to crawler website, that discourage web crawls. i.e Kijiji and Footlocker | Functional System Testing |

Table 1: Test Properties List

## 1.2 Properties To Be Tested on Pyweblinkvalidator

Different properties of the program that must be considered and verified through testing.

### 1.2.1 Ease of Use

The main objective of ease of use testing is ensuring the program is easy to set up and can easily be used by the stakeholders mentioned in the SRS document.

#### Rationale

This program setup time and ease of use must be verify. As developers we need to ensure the program is easy enough for any of the stakeholders to use and setup in minutes.

#### Testing Approach

We are going to use one of the Functional System Tests, Usability Testing. To prove our program is easy to use and step up. We will analysis our code and make sure everything is well documented, users are prompt corrected and the program is fairly easy to debug. After we believe the code is sound and easy to use, we will ask 5 non-engineers to use the program to crawl their favourite website and they will rate the ease of use. We aim to have an average score of 8/10

### 1.2.2 Accuracy

The main objective for the accuracy testing is to verify the results and status codes from the web-crawler are accurate.

#### Rationale

This program must have accurate results, so customers and stakeholders trust to products and will continue to use it to increase efficiency.

#### Testing Approach

We are going to use one of the Dynamic Testing. To prove our program is produces the correct results we will have a set of test HTML files that the program shall crawl. The programs output must correspond to what occurs when a user manual clicks all the links on the test HTML and the status codes on each HTML page.

### 1.2.3 Speed

The main objective of speed testing is to make sure the program, returns results in a timely manner.

#### Rationale

This program speed must be verified, we must assure stakeholders that the program will work quickly and increase efficiency, like we mentioned in our SRS.

**Testing Approach**

We are going to use one of the Structural System Tests, Speed Testing. To prove the program will run in a reasonable timeframe, In order to do this we are going to setup a stopwatch function at the begin of each query and it will print the time it took for each query. The time to crawl a page for any content should be between 0-2 seconds.

### 1.2.4 Fault-Tolerance

The main objective of Fault-Tolerance is to make sure this program can handle any kind of user input

**Rationale**

This program should able to take any input and if the input is the wrong format, the program should inform the user or fix the issue. This feedback with be essential in helping user use the software correctly.

**Testing Approach**

We are going to use one of the Functional System Tests, Error Handling. To prove the program can handle invalid entries and it can correct inputs missing HTTP:// or www before the domain name, we with input queries that should break the program. Queries include but not limited to 23wew, example.com, www.hi.me,1/2/2 and much more. These queries should be handled with such response as "invalid entry" or a list crawled website (normal output) if the domain was missing HTTP:// or www.

### 1.2.5 Adaptability

The main objective of Adaptability is to make sure this program will work as expect on any operating system.

**Rationale**

This program should be platform independent, our software should be able to run on any system that the use decides to use.

**Testing Approach**

We are going to use Manual Tests to verify this works. The team with test is the program works on a Linus system, Mac system and Windows system. We will load up the program on each of the mentioned Operating Systems and verify the program acts identical across each platform.

#### 1.2.6 Legal Requirements

The main objective of Legal Requirements, verify our program follows the law and a website's Terms of Use.

**Rationale**

This program should block sites that has strict rules against web crawl, such as Kijiji, Foot Looker and eBay

**Testing Approach**

We are going to use Dynamic Testing. We are going to query website on our list of forbidden website and make sure the program respond with a message along the lines of "Web Crawling is not permitted by this website". We will query sites on our forbidden list such as Kijiji and eBay and verify the program outputs the message above.

# 2 Tested Components

## 2.1 Summary

These components are different parts of our code and design we need to test and review in order verify all functional requirements have been met and the program is functioning correctly. The following section with be a list of all the components we plan on testing and an overview of how we plan to test these components.

1. Software Requirements Specifications
2. Code
3. Features
4. Error Handling

## 2.2 Software Requirements Testing

As we continue working on this project and developing more features, we most be able to run some tests that can verify all the requirements in the SRS have been met. We will implement black box tests to achieve this.

## 2.3 Code Testing

As we continue to build we must vet every team members commits to the repository. After every submission any team member other than the original submitter should analyze the submitted code, and make sure all the is sound logically and functionally. We would be using a form white box testing.

## 2.4 Features Testing

We will be observing the code as we add more features. We must verify every feature does exactly what it is supposed to do and each feature can be traced back to a requirement. More details on individual test cases with be provides sections 5 and 6.

## 2.5 Error Handling Testing

We are building a program that should be able to handle any query from the user and different error that may occur during run-time. We will implement tests that will reproduce these errors and ensure there are handled corrected. More details mentioned in sections 3,5 and 6.