**govini-site-crawler Software Test Plan**

Christopher R. Herman

### **Introduction**

The goal of this Software Test Plan is to devise a set of instructions and test-case scenarios on how to test the use and results of the script *govini-site-crawler*. The script *govini-site-crawler* is a python web crawler utilizing the python library *selenium* to accomplish scraping data from website.

**\*Note - To construct a much stricter Software Test Plan a Functional Requirements document is necessary so that all known functionality can be tested.**

### **Resource**

Interviewee - Christohper R. Herman

### **In** **Scope**

Python library utilized for the script is *selenium*. The website used to search and scrape data from is [*https://apps.dtic.mil/dodinvestment/#/advancedSearch*](https://apps.dtic.mil/dodinvestment/#/advancedSearch), with the search term *0603680F*.

### **Out of Scope**

Any other website or search term that is not listed above in the ***In Scope***section is out of scope for this project.

### **New Functionality**

This is a newly created library/script, therefore, there is no new functionality (i.e. no regression tests).

### **Performance Testing**

This script is very simple and would have very little performance improvements at this time as in most of the leveraging is done by the third-party library, *selenium*.

### **UAT**

|  |  |
| --- | --- |
| User Acceptance Testing | |
| 1. | User is able to run a script named: *run\_crawler* |
| 2. | Chrome WebDriver runs in `headless` mode (i.e. Chrome does not appear) |
| 3. | Results from the search are returned to the terminal as a json/dict object |
| 4. | Unit Tests have been created for the script |
| 5. | Integration test(s) have been created for the script |

### **Assumption + Risks**

|  |  |
| --- | --- |
| Assumptions | |
| 1. | URL: https://apps.dtic.mil/dodinvestment/#/advancedSearch |
| 2. | Search Term: 0603680F |

|  |  |
| --- | --- |
| Risks | |
| 1. | Slow internet connection may lead to a timeout |
| 2. | Slow database server may lead to a timeout |

### **Future Improvements**

Further development could be implemented in this library to make it more useful in the future. One future development would be to expand the scrapers and websites utilized to retrieve data. Abstract classes have already been created so that it is easier to add scrapers and parsers with the same structure (i.e. contracts). The abstract classes can be further improved once more information from other websites has been gathered. Another improvement is that the user could provide the website and search term to the script *run\_crawler* in the terminal. More useful exceptions could be implemented in the library so that the user knows what potentially could be causing the error. Adding the exceptions would be very easy to implement, creating a custom exceptions package for more specific exceptions related to the library which could be raised when selenium specific exceptions arise. Finally, once the library has been developed significantly with many websites to scrape from, a user interface (UI) could be built utilizing React. React can be used to develop UIs for websites and mobile applications. A backend service utilizing a python web server, Flask or Django, to easily interact with the already developed python library utilizing selenium. These frontend and backend services could be wrapped in docker-compse or individual docker services deployed to a kubernetes cluster. This way end-users would have access to data in a visual manner with a great user experience.