This UROP project is a web application test automation project. GenomeSpace is the name of the website the program is testing on. It is a cloud-based storage which acts like a bridge connectings the Nectar Research Cloud with the Genomics Virtual Laboratories. A GenomeSpace user can mount the swift container from his/her Nectar cloud and then send the files in the container to the virtual laboratories for processing. GenomeSpace also provide basic file manipulation functionalities like move, copy, rename, delete, etc.

Any faulty functionality in an application could cause customer loss and potential financial loss, therefore making sure that the application is working correctly is a crucial part in the development of any software application. GenomeSpace is a web application and has a very high dependency on the network. Any change in any part of the applications related to GenomeSpace could cause failures in GenomeSpace. A manual testing of the application requires a developer to spend huge amount of time to test through every functionality of the application and keep an eye closely on any response of the application, which is a huge cost of human and time resources and this is where test automation becomes important.

The aim of this project is to complete and provide an automatic testing program for GenomeSpace to ensure that all the functionalities are working as intended and that the broken ones could be spotted easily, immediately and with minimum cost of resources. Ultimately, the program would be put to use and aid the development and maintenance of the application.

The program is written in Python and the two libraries used in this program are Selenium and Unittest. Selenium provides functionalities of simulating the website browsing process, for examples open up a browser, go to a page, type something in the text fields and click a link to go to the linked page, and also provides the functionalities like halt the program until an alert is present or wait for the page to be loaded. Unittest is for putting the test into test cases, one for each functionality. The test cases are run one after another and the result of each one is shown. A report based on the test results will be auto generated after each run of the program.

The functionalities that are tested at this stage include:

* user registration
* user login
* mount a swift container
* disconnect a swift container
* import a file using the expired public URL
* generate public URL of a file and access the file with the URL
* file rename
* copy file between directories
* copy file between containers
* move file between directories
* move file between containers
* delete a file
* lunch a GVL instance with files
* generate a DOI of a file for publishing

Process:

Problems along the way with solution:

Initially, the user interface was tested with Selenium by conducting a series of events, waiting for the response and asserting the outcome was as expected. In order to do so, elements of the web page needed to be located for the events to be conducted on. However, most of the elements in the page were dynamically generated and injected by Ajax. They didn’t have any id, any name or anything that can be uniquely identified. Some of them are even merely with tag names. It was almost impossible to locate the element without hardcoding some numbers and the hardcoded code are always fragile and easy to break. That was why most of the test cases are not about the interface now.

As the functionalities of GenomeSpace are service oriented. The communications between clients and servers are the most crucial part. In order the solve the problem described above another approach is used for the test cases apart from registration and log-in, which is sending requests and checking if the response is correct.

Conclusion:

The test automation program is basically to test if the functionalities are working as intended and generate report of the test outcome.

At the setting up stage of the program, Chrome driver, which was installed in advance, or Firefox driver, which came along with Selenium package are started and the a window shows up, then Genomespace website (<https://genomespace.genome.edu.au/jsui>) is opened. The program looks for the cookie stored under the same directory as itself is in and restore it, if there is any, for skipping the log in test and speed up the debugging process of the program itself, as the web drivers would always start a new window with no cookies, no cache and the login test is relatively slow. The program is then ready to test.

The biggest problem encountered when setting up was that the web drivers would always start a new window with no cookies and no cache. Logging in was needed every time running the program to pass the authentication and this takes a relatively long time, which slowed down process of debugging the program itself.