**Automatic tests report for TC001, TC002, TC003 and TC004**

To automate the Test Cases TC001, TC002, TC003 and TC004 Cucumber and python 3.6.5 were used.

Selenium webdriver was used for the frontend tests, and the requests package for the backend tests.

The frontend tests as requested were run on the chrome browser.

Two feature files and two step files were created:

* ProductSubscription.feature
* Users.feature
* Steps\ProductSubscriptionSteps.py
* Steps\UsersFeatureSteps.py

The nomenclature used for the feature files was based on the feature names indicated in the Acceptance Criteria for TC001, TC002, TC003 and TC004.

Additionally, according to the best practices, a python object developed by me in the past for frontend tests was used (Generic\_Web\_Testing\_Methods.py).

All the tests have passed according to the acceptance criteria.

The tests can be run simply by running the “behave” command with no arguments in the directory of the feature files, after installing all the prerequisites.

The report produced by behave can be found in the text file “output\_of\_behave.txt”.

**Frontend tests**

As previously described for the frontend test the following files were created:

* Generic\_Web\_Testing\_Methods.py
* ProductSubscription.feature
* Steps\ProductSubscriptionSteps.py

Generic\_Web\_Testing\_Methods.py implements an object that is essentially a wrapper of the phyton webdriver object allowing to (amongst other things):

* Click on an element
* Fill in a text form
* Select options in a form drop down menu

All the methods in this file incorporate according to the best practices:

* Webdriverwait calls to deal with server delays
* Exception handling procedures
* Reporting procedures (not used in this exercise as cucumber has its own reporting methods)

Without loss of generality all the methods use xpath to locate the elements.

As all the elements that needed accessed in TC001 had IDs, the following steps were implemented for the product subscription feature:

* Open a browser given the browser and the URL
* Select an option in a dropdown menu given the id and the text of the option to select
* Click on an element given the ID
* Fill in a text to an element given the ID
* Check that the text in a tag with a given ID, agrees with acceptance criteria

All the steps were created to ensure maximum reusability for tags with IDs.

**Frontend tests**

For the Users feature (TC002, TC003 and TC004) the following Cucumber steps were developed:

* Add the API URL to the cucumber context object namespace (also initializes the payload attributes dictionary as an empty dictionary)
* Add an attribute to the Payload dictionary
* Send a Post command to the API
* Send a Delete command to the API
* Check that the http status code is according to the acceptance criteria
* Check that a given text is present in the webservice response according to the acceptance criteria

**Prerequisites**

1. Phyton 3.6.5
2. Cucumber for python (behave implementation)
   1. Can be installed running “pip install behave”
3. Selenium webdriver and the chrome driver
   1. Can be installed running “pip install selenium”
4. For the back end tests the requests package was used
   1. Can be installed running “pip install requests”