**Marlow\_Navigation Selenium\_UI Test Implementation**

1. **Requirement**

At ‘https://www.amazon.com’, go to Electronics -> Computers & Accessories -> Computers & Tablets -> Select 4th item and add to cart -> Verify that the item’s price is the same as the price displayed in the cart.

1. **Design Flow**

The requirement is to look for the product with index in the list (e.g: the 4th product).

Because Amazon is a commercial website, its content is changed rapidly. We don’t know if the requested product is existed in the list of Computers & Tablets or if it’s available to add to cart (e.g Out Of Stock)

Without specify quantity of the product to add to cart, by default quantity=1 is chosen. However, we can extend the test by allow user to set quantities of the product. If the ordered quantity exceeds total numbers of the product in the stock, then it raises error.

With that idea, the flow of code implementation will be explained by diagram and text as followed:

*(note: product\_index is the index of requested product from the list)*

Diagram

Description automatically generated

1. Go to Computers & Tablets
2. Check if the requested product is out of the list

For example: Assume there are only 5 products in Computers & Tablets, it raises error if we request 7th product.

1. The product found, click on product link to get detail of the product
2. Check if the product is in stock?
3. If in stock, check if the ordered quantity is ok?

For example, total number of the product in stock is 6, and the quantity we want to add to cart is 7, then the test will be failed.

1. If quantity is ok, add to cart
2. Go to Cart & Checkout
3. Check if the product is found in cart? (by using product\_title)
4. The product found in cart -> check if price of the product is expected

(10) The price is fine -> PASSED. Otherwise, FAILED

1. **Code Structure**

* **/config**: store pre-config of the test: browser type, base URL, and headless (1: headless, 0: GUI)
* **/data**: store data-driven files used for the test (product index, product\_quantity)
* **/lib**

base/base\_page.py: act like selenium wrapper, handle selenium methods with more secured options

locators: define locators for the test

pages: each file presents what we want to interact with each page

*menu\_navigation.py* – access to side menu and its departments/sub-departments

*computers\_tables.py* – handle interactions at page Computers & Tablets

*cart\_checkout.py* – handle interactions at page Cart & Checkout

* **/test**: where the test case defined and executed by using python unittest
* **/utilities**:
* **/output**

reports – for test result report

logs – for test execution log

A picture containing graphical user interface

Description automatically generated

1. **Data driven test**

To demonstrate the flow of code implementation works, I use data-driven method for the test. Data of the test is defined in the data/data.csv file. It has 2 columns: product\_index and product\_quantity.

* Change these to different values to see how the test run with different cases as below picture:

Table

Description automatically generated

1. **Test result: PASS/FAIL**

PASS when the price of the product is the same as the product’s price displayed in cart

Otherwise, FAIL

1. **Test Execution**

Example of the test execution in MacOS:

Create, activate and enter to virtual environment (venv):

➜ marlow\_navigation python3 -m venv venv

➜ marlow\_navigation source venv/bin/activate

(venv) ➜ marlow\_navigation

Install dependences in venv:

(venv) ➜ marlow\_navigation pip3 install -r requirements.txt

Run the script:

(venv) ➜ marlow\_navigation python3 test/test\_product\_price.py

After running, deactivate venv:

(venv) ➜ marlow\_navigation deactivate

➜ marlow\_navigation

1. **Enhancement**

This is just small demo how to interact with WebUI using selenium/python. In a real-world project, Robot Framework will be the good choice for automation.