

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐÔNG Á**

**……\*\*\*.…..**



**Tên đề tài :**

*Đà Nẵng, ngày 26/04/2022*

**XÂY DỰNG SELENIUM FRAMEWORK THEO**

**MÔ HÌNH PAGE OBJECT MODEL (POM)**

GVHD : ThS. Tạ Quốc Ý

SVTH : Lê Văn Linh

Lớp : ST19A1B

IDSV : 49036

**Ng**

**Mục lục**

[**I. Giới thiệu 2**](#_Toc101997657)

[**II. Đặt vấn đề 2**](#_Toc101997658)

[**2.1 Website Lazada 2**](#_Toc101997659)

[**2.2 Website Amazon 2**](#_Toc101997660)

[**III. Giải quyết vấn đề 3**](#_Toc101997661)

[**3.1. Website Lazada 3**](#_Toc101997662)

[**3.1.1. Selenium Page 3**](#_Toc101997663)

[**3.1.2. Selenium Test 4**](#_Toc101997664)

[**3.1.3. Selenium Utils 5**](#_Toc101997665)

[**3.2 Website Amazon 6**](#_Toc101997666)

[**3.2.1. Selenium Page 6**](#_Toc101997667)

[**3.2.1. Selenium Tests 8**](#_Toc101997668)

[**3.2.1. Selenium Utils 11**](#_Toc101997669)

[**IV. Kết luận 13**](#_Toc101997670)

# I. Giới thiệu

Sau khi học môn kiểm thử phần mềm tôi đã quyết định làm về Xây dựng Selenium framework theo mô hình Page Object Model (POM) và tiến hành chọn code 1 vài test module, chạy và report kết quả.

Trang web mà tôi sẽ làm về testcase đó là website lazada và Amazon công ty thanh toán bằng điện toán đám mây.

POM (Page Object Model) là 1 design pattern (mẫu thiết kế) giúp **mô hình hóa các pages**, hoặc **các phần (component: header, footer, menu…) trong page của trang web** thành mỗi đối tượng riêng biệt. Mỗi component / page sẽ gói gọn tất cả các hành động và các thuộc tính của component / page đó.

# II. Đặt vấn đề

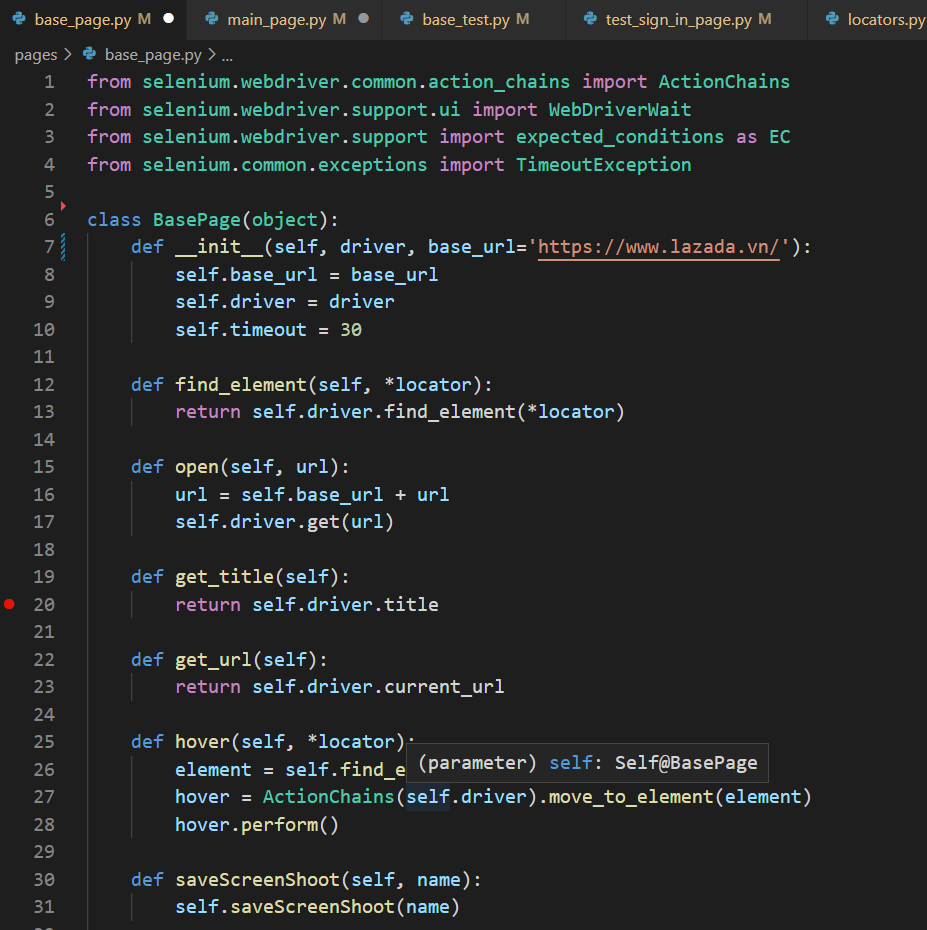
* 1. **Website Lazada**
* Khi người dùng chuyển đến trang chính, trang sẽ được tải
* Trong trang Chính, khi người dùng tìm kiếm nút "Nexus 5", họ sẽ thấy kết quả cho Nexus 5'
  1. **Website Amazon**
* Khi người dùng chuyển đến trang chính, trang sẽ được tải
* Trong trang Chính, khi người dùng tìm kiếm nút "Nexus 5", họ sẽ thấy kết quả cho Nexus 5'
* Trong Trang chính, khi người dùng nhấp vào nút "Hát lên", anh ta sẽ thấy Trang đăng ký
* Trong Trang chính, khi người dùng nhấp vào nút "Đăng nhập", anh ta sẽ thấy Trang đăng nhập
* Trong Trang đăng nhập, khi người dùng đăng nhập với người dùng hợp lệ, anh ta sẽ thấy Trang chủ
* Trong Trang Đăng nhập, khi người dùng đăng nhập bằng người dùng hợp lệ, anh ta sẽ thấy Thông báo Lỗi

# III. Giải quyết vấn đề

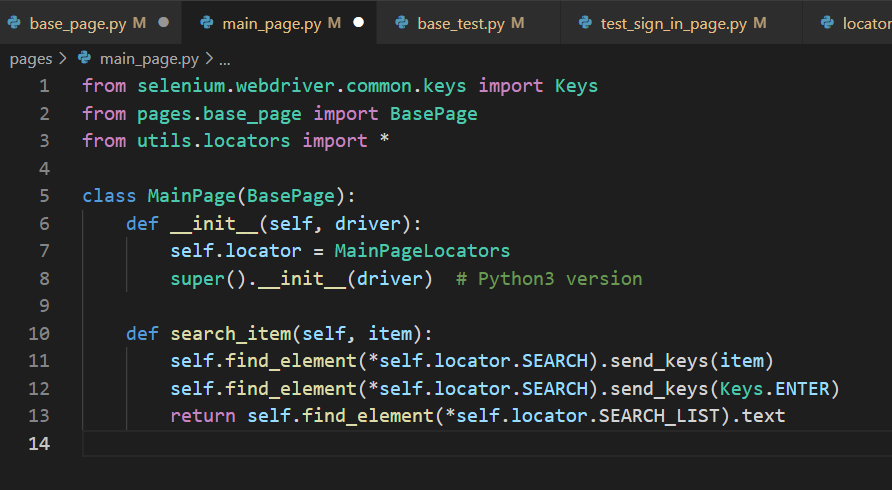
## **3.1. Website Lazada**

### 3.1.1. Selenium Page

* **Base\_page.py**

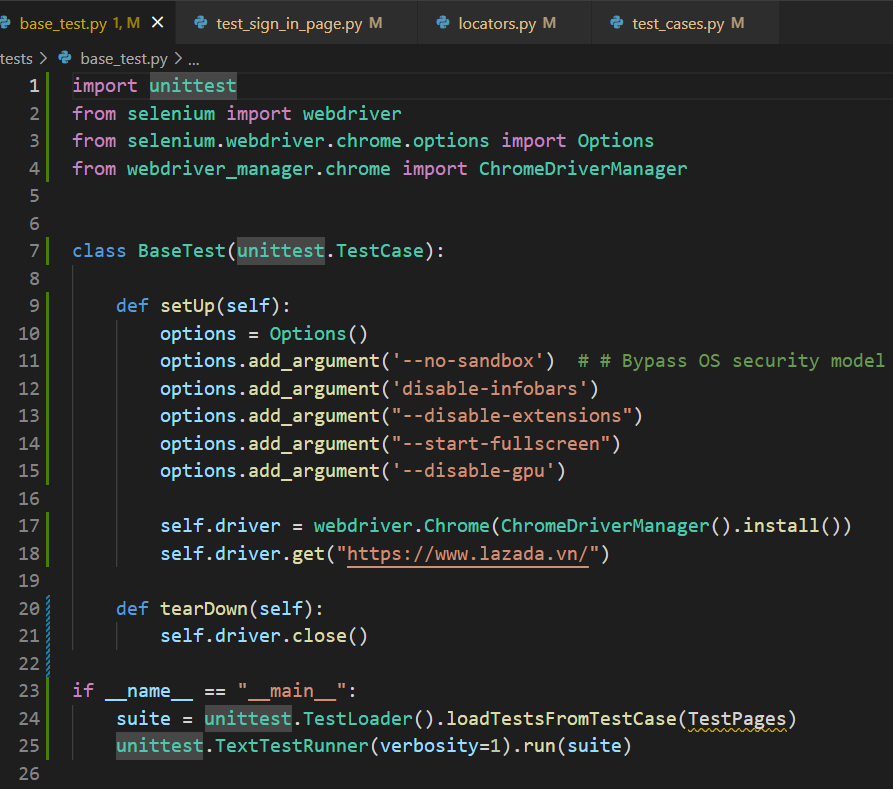
****

* **Main\_page.py**

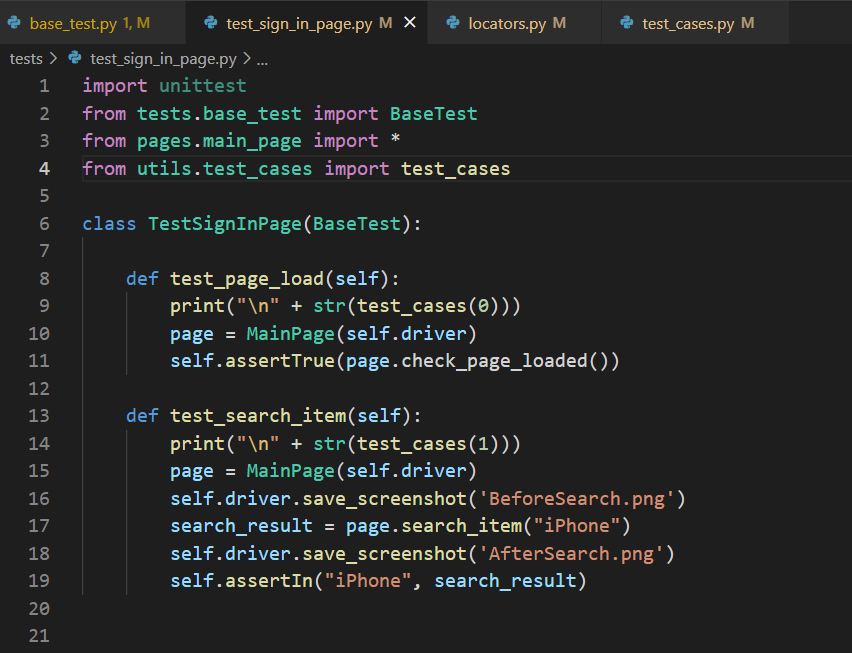
****

### 3.1.2. Selenium Test

* **Base\_test.py**

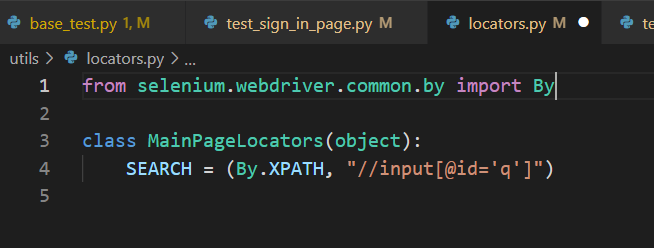
****

* **test\_sign\_in\_page.py**

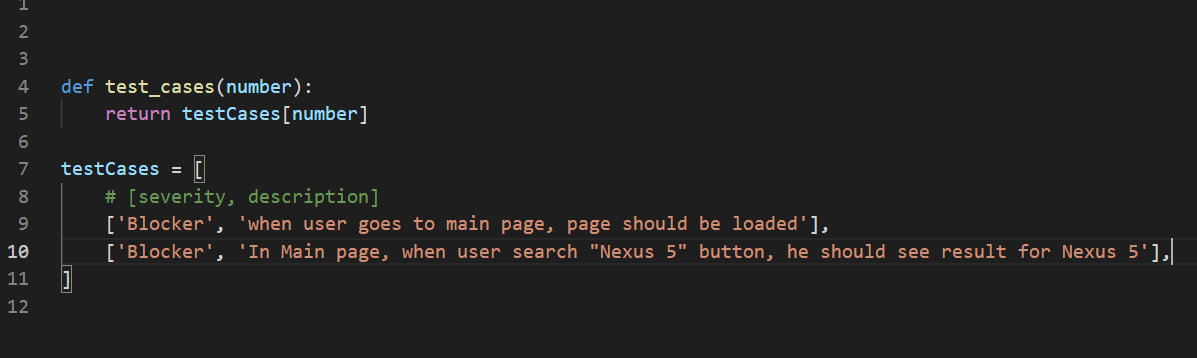
****

### 3.1.3. Selenium Utils

* **locators.py**

****

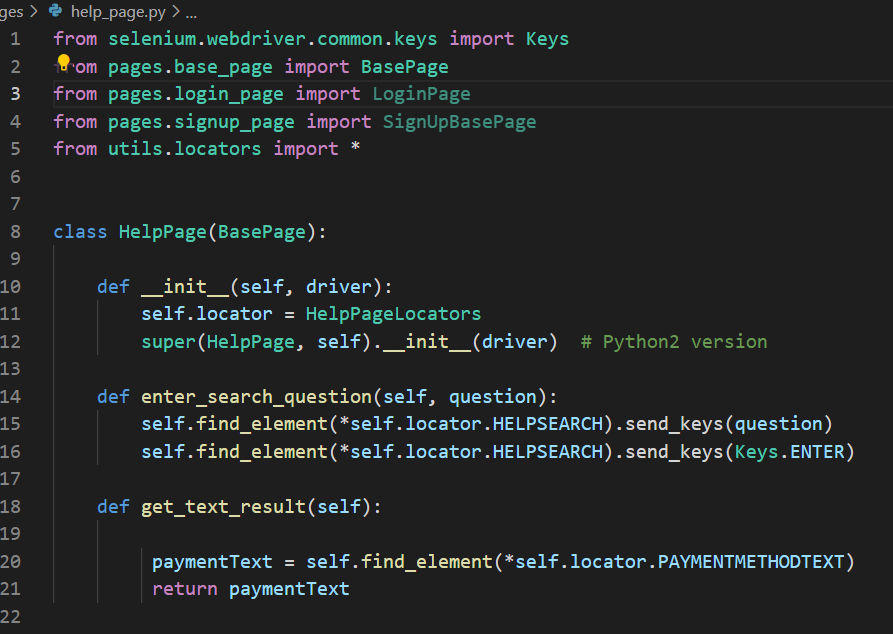
* **test\_cases.py**

****

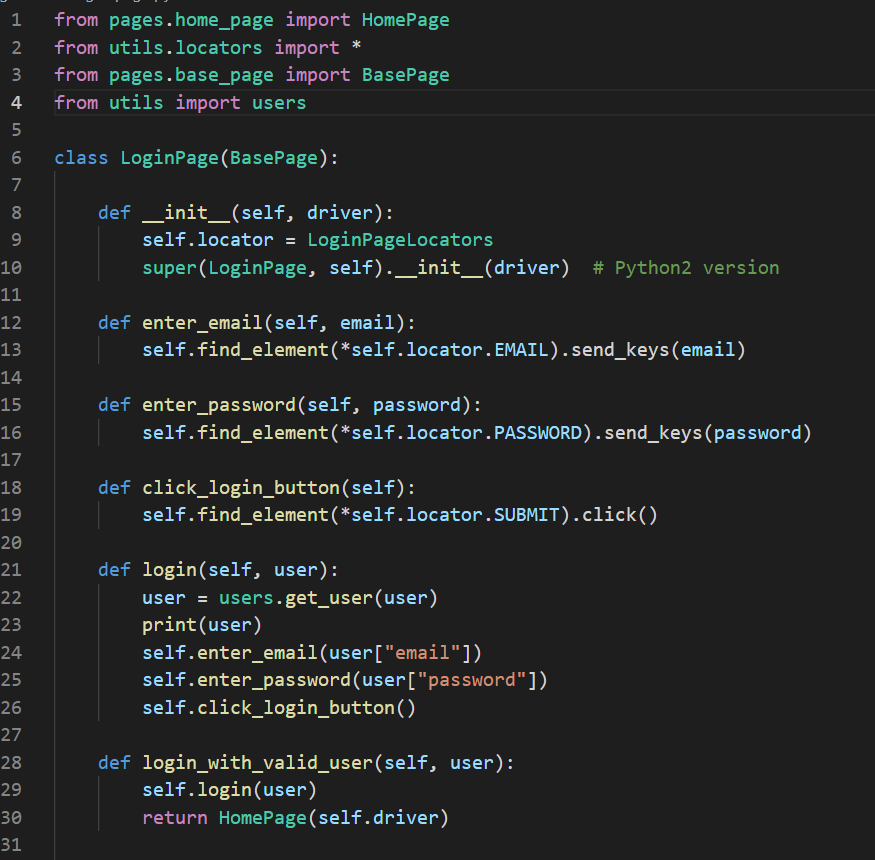
## **3.2 Website Amazon**

### 3.2.1. Selenium Page

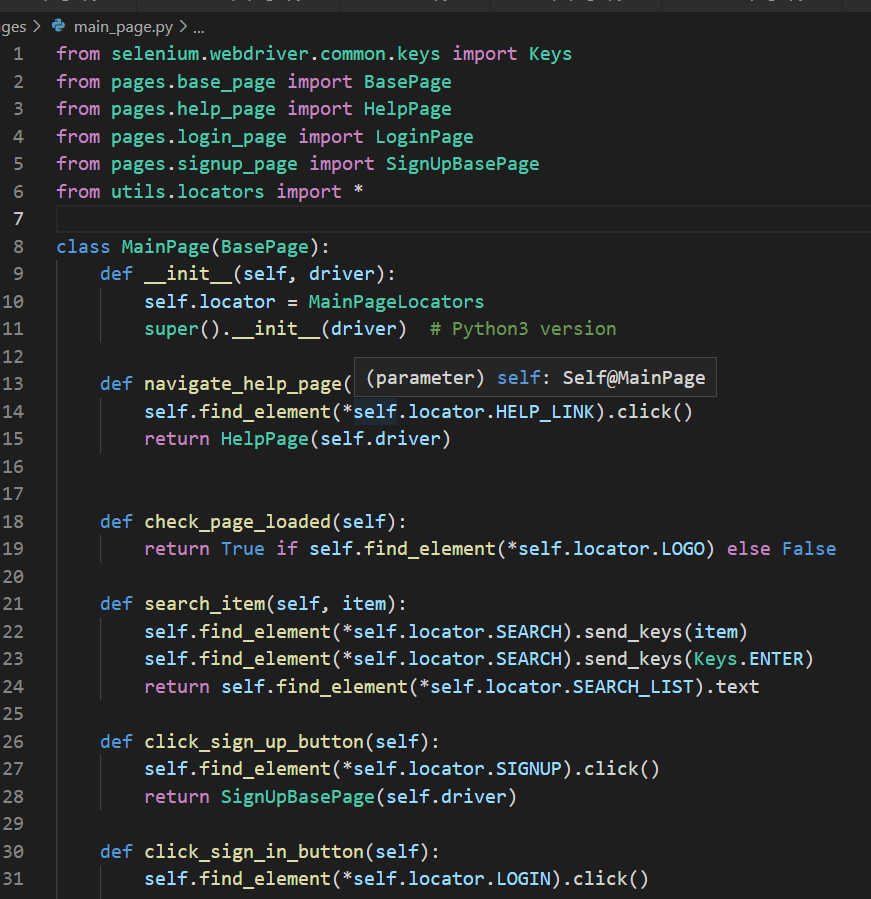
* **Help\_page.py**

****

* **home\_page.py**
* from pages.base\_page import BasePage
* class HomePage(BasePage):
* pass
* **login\_page.py**

****

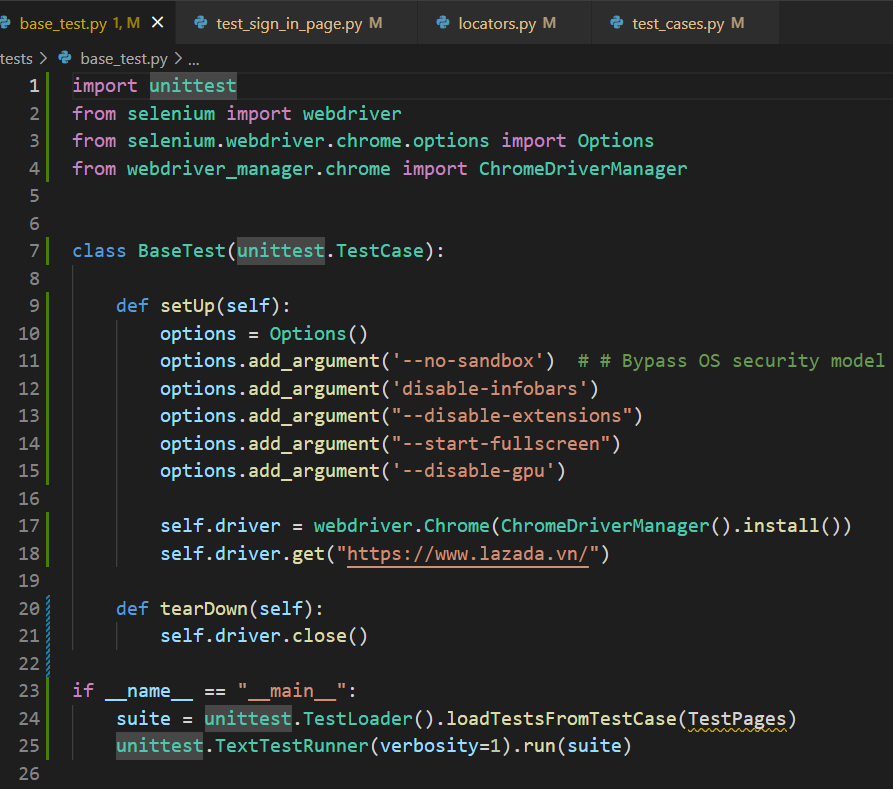
* **main\_page.py**

****

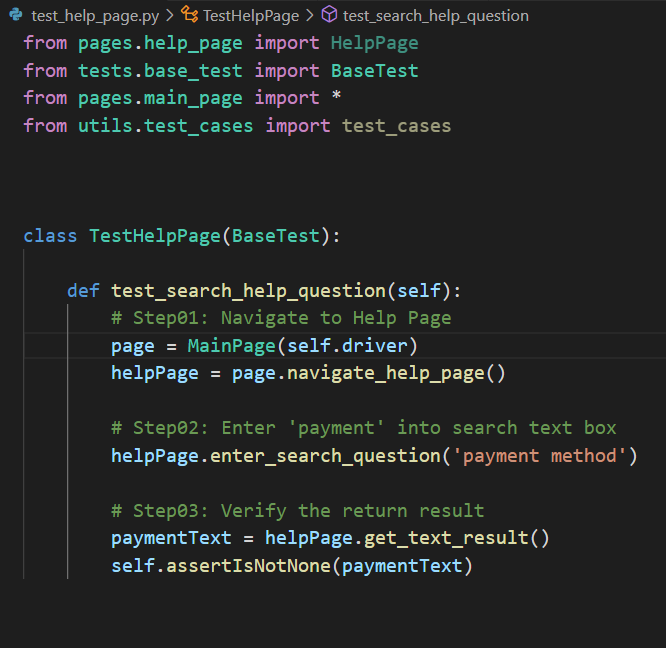
* **signup\_page.py**
* from pages.base\_page import BasePage
* class SignUpBasePage(BasePage):
* pass

### 3.2.1. Selenium Tests

* **base\_test.py**

****

* **test\_help\_page.py**

****

* **test\_sign\_in\_page.py**

import unittest

from tests.base\_test import BaseTest

from pages.main\_page import \*

from utils.test\_cases import test\_cases

class TestSignInPage(BaseTest):

    def test\_page\_load(self):

        print("\n" + str(test\_cases(0)))

        page = MainPage(self.driver)

        self.assertTrue(page.check\_page\_loaded())

    def test\_search\_item(self):

        print("\n" + str(test\_cases(1)))

        page = MainPage(self.driver)

        self.driver.save\_screenshot('BeforeSearch.png')

        search\_result = page.search\_item("iPhone")

        self.driver.save\_screenshot('AfterSearch.png')

        self.assertIn("iPhone", search\_result)

    def test\_sign\_up\_button(self):

        print("\n" + str(test\_cases(2)))

        page = MainPage(self.driver)

        sign\_up\_page = page.click\_sign\_up\_button()

        self.assertIn("ap/register", sign\_up\_page.get\_url())

    def test\_sign\_in\_button(self):

        print("\n" + str(test\_cases(3)))

        page = MainPage(self.driver)

        login\_page = page.click\_sign\_in\_button()

        self.assertIn("ap/signin", login\_page.get\_url())

    def test\_sign\_in\_with\_valid\_user(self):

        print("\n" + str(test\_cases(4)))

        main\_page = MainPage(self.driver)

        login\_page = main\_page.click\_sign\_in\_button()

        result = login\_page.login\_with\_valid\_user("valid\_user")

        self.assertIn("yourstore/home", result.get\_url())

    def test\_sign\_in\_with\_in\_valid\_user(self):

        print("\n" + str(test\_cases(5)))

        main\_page = MainPage(self.driver)

        login\_page = main\_page.click\_sign\_in\_button()

        result = login\_page.login\_with\_in\_valid\_user("invalid\_user")

        self.assertIn("There was a problem with your request", result)

### 3.2.1. Selenium Utils

* **locators.py**

from h11 import ERROR

from selenium.webdriver.common.by import By

# for maintainability we can seperate web objects by page name

class MainPageLocators(object):

    LOGO = (By.ID, 'nav-logo')

    ACCOUNT = (By.ID, 'nav-link-accountList')

    SIGNUP = (By.CSS\_SELECTOR, '#nav-signin-tooltip > div > a')

    LOGIN = (By.CSS\_SELECTOR, '#nav-signin-tooltip > a')

    SEARCH = (By.ID, 'twotabsearchtextbox')

    SEARCH\_LIST = (By.CSS\_SELECTOR, 'div[data-component-type="s-search-result"]')

    HELP\_LINK = (By.XPATH, '//a[text()=\'Help\']')

class LoginPageLocators(object):

    EMAIL = (By.ID, 'ap\_email')

    PASSWORD = (By.ID, 'ap\_password')

    SUBMIT = (By.ID, 'signInSubmit-input')

    ERROR\_MESSAGE = (By.ID, 'message\_error')

class HelpPageLocators(object):

    HELPSEARCH = (By.XPATH, '//input[@id=\'helpsearch\']')

    PAYMENTMETHODTEXT = (By.XPATH, '// h2[contains(text(), \'Manage Payment Methods\')]')

* **test\_cases.py**

def test\_cases(number):

    return testCases[number]

testCases = [

    # [severity, description]

    ['Blocker', 'when user goes to main page, page should be loaded'],

    ['Blocker', 'In Main page, when user search "Nexus 5" button, he should see result for Nexus 5'],

    ['Blocker', 'In Main page, when user click "Sing up" button, he should see Sign up Page'],

    ['Blocker', 'In Main page, when user click "Sing in" button, he should see Sign in Page'],

    ['Blocker', 'In Login Page, when user login with a valid user, he should see Home Page'],

    ['Blocker', 'In Login Page, when user login with a in-valid user, he should see Error Message'],

* **users.py**

from operator import itemgetter

# we can store test data in this module like users

users = [

    {"name": "invalid\_user", "email": "invalidUser@test.com", "password": "qwert1235"},

    {"name": "valid\_user", "email": "validUser@yahoo.com", "password": "ValidPassword"},

    {"name": "Staff2", "email": "staff@test.com", "password": "qwert1235"},

    {"name": "Admin0", "email": "admin@test.com", "password": "qwert1234"},

    {"name": "Admin1", "email": "admin@test.com", "password": "qwert1234"},

    {"name": "Admin2", "email": "admin@test.com", "password": "qwert1234"},

]

def get\_user(name):

    try:

        return next(user for user in users if user["name"] == name)

    except:

        print("\n     User %s is not defined, enter a valid user.\n" % name)

# IV. Kết luận

- Giúp bảo trì dễ dàng: POM hữu ích khi có sự thay đổi trong phần tử giao diện người dùng hoặc có sự thay đổi trong một hành động. Vì mọi màn hình sẽ có các tệp java khác nhau, nên việc xác định này là cần thiết để thực hiện các thay đổi bắt buộc đối với các tệp đó. Điều này làm cho các trường hợp kiểm thử dễ bảo trì và giảm lỗi.

- Giúp sử dụng lại mã: Như đã thảo luận, tất cả các màn hình đều độc lập. Bằng cách sử dụng POM, người ta có thể sử dụng mã thử nghiệm cho một màn hình và sử dụng lại nó trong một trường hợp thử nghiệm khác. Không cần phải viết lại mã, do đó tiết kiệm thời gian và công sức.

- Dễ đọc code: Khi tất cả các màn hình có các tệp java độc lập, người ta có thể dễ dàng xác định các hành động sẽ được thực hiện trên một màn hình cụ thể bằng cách điều hướng qua tệp java đó thôi. Nếu một thay đổi ảnh hưởng đến một phần mã code nhất định thì nó có thể được thực hiện một cách hiệu quả mà không ảnh hưởng đến các tệp (class/package) khác.

- Tạo kho lưu trữ: Có thể một kho lưu trữ duy nhất cho các xử lý chung hoặc hoạt động chung cho các trang thay vì có các xử lý này nằm rải rác trong các test case riêng lẻ

**V. Lời Cảm ơn**

Qua thời gian học tập với sự dẫn dắt của thầy Tạ Quốc Ý em đã làm và hoàn thành được cách viết Selenium Python theo mô hình Page Object Model môn Kiểm thử phần mềm. Tuy cách viết Selenium bây giờ còn nhiều thiếu sót chưa hoàn thành những mục tiêu ban đầu đề ra nhưng vẫn hoàn thành được testcase cơ bản. Hiện tại em vẫn còn nhiều thiếu sót về kinh nghiệm và kỉ thuật. Em sẽ cố gắng trao dồi để có thể làm Selenium được tốt hơn đầy đủ hơn.