Selenium Automation Testing using Python

Requirements:

1. download Python

   https://www.python.org/downloads/

2. download Pycharm community

   https://www.jetbrains.com/pycharm/download/#section=windows

3. edit system environmental variables -> system variables -> path -> edit

   C:\Users\ARAVIND\AppData\Local\Programs\Python\Python38

   C:\Users\ARAVIND\AppData\Local\Programs\Python\Python38\Scripts

4. create python project in pycharm

5. run these commands in terminal

   pip install selenium

   pip install webdriver-manager //for managing web driver

   pip install openpyxl //for managing excel data

   pip install schedule // for scheduling tasks to run at a particular date and time

   pip install smtplib // for sending email

   pip install allure-pytest

   pip install pytest

   pip install unittest

   pip install python-dotenv // for setting up .env file.

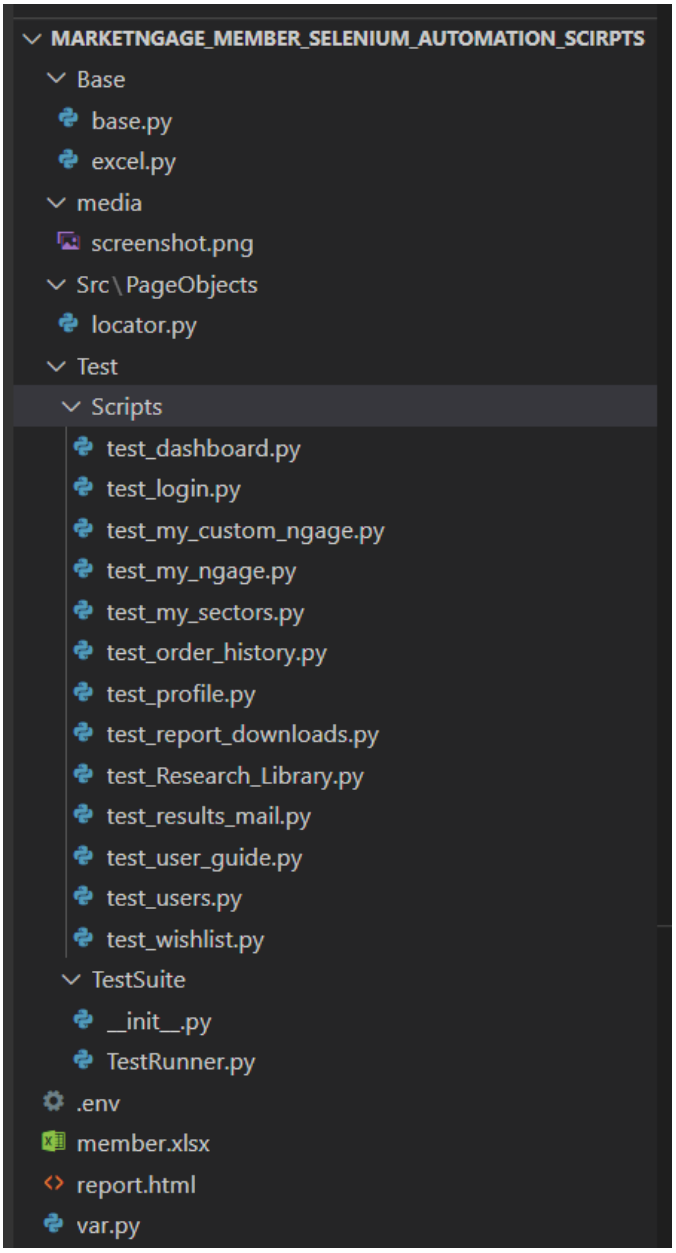
   pip install prettytable // for generating html tables

If you choose visual studio as your go to editor, then download python in visual studio code and run the above commands after creating a project.

What is Page Object Model(POM)?

Page Object Model is a design pattern that is used mostly in Web UI testing for enhancing maintenance of the test code and reducing duplication of it. Each page in the web application has to be separate class different than the others. These are the classes that contain Web Elements(Locators) of the web page, the classes include the methods which perform operations on those Web Elements and the classes are exist so that to execute corresponding page tests.

Folder structure: POM



The **Base folder** consists of base.py and excel.py files .

Base.py consists of the parent class Base whose features and functionalities are inherited in is inherited in the Test cases in files located in Test/Scripts.It consists of code for handling browsers(opening, closing) and loading excel workbooks. We can also add functions of code that we intent to use in child classes. I added a function for generating email which is called with exception details as arguments in test script classes when an exception occurred during running of scripts.

Excel.py consists of code to loading excel workbooks and code to make sure the results for the test cases we ran are recorded in the excel.

For more info on how to read or write data from excel visit https://openpyxl.readthedocs.io/en/stable/tutorial.html

The **media** folder consists of screenshots we took using webdriver .

The **Test/Scripts** folder consists of files for test cases for each page of our application. Each of those files consists of a class which inherits features from parent class in Base/Base.py file. In side those classes we write functions for test cases. It is important to start the function name with test_ . The code for test cases is located in try block. This is done to prevent breakage in flow of execution of test cases. We store the result of test case in the respective sheet of excel file and dictionary variable. In case of an exception, the program control is passed to Except block where we record exception details, take a screenshot where the exception has occurred and share those details by calling mail function present in base.py file.

For more info on how to locating html elements visit https://selenium-python.readthedocs.io/locating-elements.html

The locators required for test scripts are located in **Src/PageObjects/locator.py** file. We store locators as variables and we import this file in test scripts files and use variables to locate html elements on web page.

We can run multiple test cases at once by using Test suites. These test suites are located in **TestSuite/TestRunner.py** file. In this file we import all the test case classes from Test/Scripts and load those test cases using unittest.TestLoader(). Then we create a test suite and we run that test suite. We can also schedule our tasks at a particular date and time in this file.

I included the **.env** file in the project folder. I t consists of variables that I consistently use throughout the program like base_url.

We can use the contents of .env file by using the following sniplet.

```
from dotenv import load_dotenv
load_dotenv()
os.getenv('base_url')
```

the var.py file consists of array and dictionary datatype variables require to generate test report that is sent via mail to the users. The Test Report mail functionality is present in **Test/Scripts/test_results_mail.py** file.

The excel sheet that contains input data for forms and used to store Test results is present in project folder with name **member.xlsx**