1. What are the components in creating this Selenium Hybrid Framework?
2. Python, Selenium, PyTest, POM, HTML reports.
3. What is a framework?
4. Framework is an organized way of maintaining project related files together. In a framework, all the files will communicate with each other to perform a certain task.
5. What is a Hybrid Framework?
6. This is a combination of built-in and user-defined framework. We build and customize on top of built-in frameworks.
7. Phases in Automation Testing.
8. i. Requirement Analysis: Analyze the requirements and come up test scenarios and testcases.

ii. Test Planning/Tool Choosing: What are the tools available right now, which would be the best tool suitable for the requirement, and the skillset of the team in using them.

iii. Choosing and designing Testcases: What are the test cases we need to design and choose to automate. Priority given to re-testing/confirmation tests/regression test cases/other automatable.

iv. Test Execution: Execution of Testcases.

v. Maintenance: Using SCM tools like Git and GitHub, for maintenance and collaboration purposes, and integration with CI/CD tools like Jenkins.

1. What are we automating in this project?
2. We chose an e-Commerce Application called “nopcommerce” for automation.

Frontend: https://demo.nopcommerce.com/

Backend: <https://admin-demo.nopcommerce.com/>

1. What the steps required for this project?
2. Step-1: Create a new Project and install required packages/plugins.

* Selenium: Selenium libraries.
* PyTest: Python Unit test framework.
* Pytest-html: PyTest HTML Reports.
* Pytest-xdist: Run parallel tests.
* Openpyxl: MS Excel Support.
* Allure-pytest: To generate reports.

Step-2: Create a Folder Structure.

Project

|

PageObjects(package)

|

Test Cases(package)

|

Utilities(package)

|

Test Data(folder)

|

Configurations(folder)

|

Logs(folder)

|

Reports(folder)

|

Run.bat

Step-3: Automating Login TC.

* Create LoginPage Object class under “pageObjects” package.
* Create LoginTest under “testCases” package.
* Create conftest.py under “testCases" package.
* To run the tests from terminal: “pytest -v -s testCases/test\_login.py”

Step-4: Capture screenshot on failures.

* Update Login Test with screenshot under “testCases”.

Step-5: Read common values from ini file.

* Add “config.ini” file in “Configurations” folder.
* Create “readProperties.py” utility file under “utilities” package to read common data from the ini file.
* Replace hard-coded values in Login TestCase.

Step-6: Add Logs to test case.

* Add customLogger.py under “utilities” package.
* Add logs to Login test case.

Step-7: Run tests on desired browsers/cross-browser/parallel testing.

* Update conftest.py with required Fixtures which will accept cmd line argument(browser).
* Pass the browser name as argument in cmd line using the ‘--browser’ option.

pytest -v -s testCases/test\_login.py --browser chrome

* Pass the number of workers or threads to be run in parallel using the ‘-n’ option.

Pytest -v -s -n=5 testCases/test\_login.py --browser firefox

Step-8: Generate pytest HTML reports.

* Update conftest.py with pytest hooks.
* To generate HTML report use the ‘--html' option:

Pytest -s -v -n=3 –html=Reports\report.html testCases/test\_login.py –browser chrome

Step-9: Automating Data-driven TC.

* Prepare test data in excel sheet, place the excel file inside the TestData folder.
* Create “ExcelUtils.py” utility class under utilities package.
* Create LoginDataDrivenTest under testCases.
* Run the testcases.

Step-10: Adding new TCs

* Add new customer
* Search customer by email
* Search customer by name

Step-11: Grouping Tests

* Grouping markers (Add markers to every test method)

@pytest.mark.sanity

@pytest.mark.regression

@pytest.mark.smoke etc.

* Add marker entries in pytest.ini file

pytest.ini

------------

[pytest]

markers =

sanity

regression

smoke

* Select groups at run time using ‘-m’ option

-m “sanity”

-m “regression”

-m “sanity and regression”

-m “sanity or regression”

Step-12: Create a local git repo and connect local repo to remote repo:

* git init
* git remote add origin <remote repo URL>

Step-13: Before doing commit for the first time, we need to execute:

* git config –global user.name “<GitHub username>”
* git config –global user.email “<email>”

Step-14: Add the code to the staging/index area and then commit it to git local repo:

* git status => Check the file status before adding and committing
* git add -A
* git commit -m “<commit message>”

Step-15: Push the files to github remote repo.:

* git push -u origin master

Step-16: Pull all changed file(s) from the remote repo to local working directory:

* git pull
* Make changes to the file and repeat step-14 and 15