**Selenium Hybrid Framework**

(Python, Selenium, Pytest, Page Object Model, HTML Reports)

**Step 1: Create new Project & Install Required Packages/plugins**

* Selenium: Selenium Libraries
* Pytest: Python Unit Test framework
* Pytest-html : Pytest HTML Reports
* pytest-xdist : Run Tests Parallel
* Openpyxl : MS Excel Support
* Allure-Pytest: to generate allure reports

**Step 2: Create Folder Structure**

**Project Name**

pageObjects (Package)

testCases (Package)

utilities (Package)

TestData (Folder)

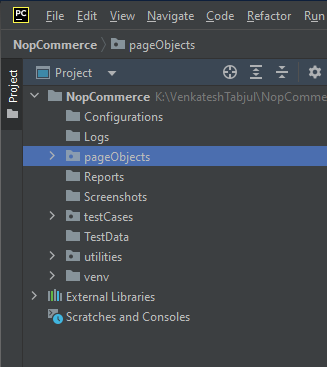
Configurations (Folder)

Logs (Folder)

Screenshots (Folder)

Reports (Folder)

Run.bat



**Step 3: Automating Login Test Case**

3.1: Create LoginPage Object Class under "pageObjects"

3.2: Create LoginTest under "testCases"

3.3: Create conftest.py under "testCases 🡪 to avoid duplication

**Step 4: capture screenshot on failures**

4.1. Update Login Test with Screenshot under "testCases"

**Step 5: Read common values from ini file.**

5.1: Add "config.ini" file in "Configurations" folder.

5.2: Create "readProperties.py" utility file under utilities package to read common data

5.3: Replace hard coded values in Login test case.

**Step 6: Adding logs to test case**

6.1: Add customLogger.py under utilities package.

6.2: Add logs to Login test case.

**Step 7: Run Tests on Desired Browser/Cross Browser/Parallel**

7.1: update contest.py with required Fixtures which will accept command line argument (browser).

7.2: Pass browser name as argument in command line.

**To Run tests on desired Browser**

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

pytest -v -s testCases/test\_Login.py --browser firefox

**To Run tests parallely**

pytest -v -s –n=3 testCases/test\_Login.py --browser chrome

pytest -v -s –n=3 testCases/test\_Login.py --browser firefox

**Step 8: Generate Pytest HTML reports**

8.1: Update conftest.py with Pytest hooks

8.2: To generate HTML report run below command

pytest -v -s –n=3 **--html=Reports\report.html** testCases/test\_Login.py --browser chrome

**Step 9: Automating Data Driven Test Case**

9.1: Prepare test data in Excel sheet, place the excel file inside the TestData folder.

9.2: Create "ExcelUtils.py” utility class under utilities package.

9.3: Create LoginDataDrivenTest under testCases

9.4: Run the test case

**Step 10: Adding new testCases**

10.1: Add new customer

10.2: Search customer by email

10.3: Search customer by name

**Step 11: Grouping Tests**

11.1: Grouping markers (Add markers to every test method)

@pytest.mark.sanity

@pytest.mark.regression

11.2: Add Marker entries in pytest.ini file

pytest.ini

[pytest]

markers=

sanity

regression

11.3: Select groups at run time

-m “sanity”

-m “regression”

-m “sanity and regression”

-m “sanity or regression”

Run Command:

Pytest -v -s -m “sanity or regression” **--html=Reports\report.html** testCases/test\_Login.py --browser chrome

**Step 12: Run Tests in cmd prompt and run.bat file**

12.1: Create run.bat file

Pytest -v -s -m “sanity” **--html=./Reports/report.html** testCases/test\_Login.py --browser chrome

12.2: Open cmd prompt as **Administrator** and then run **run.bat** file

**Step 13: Push the code to Git & GitHub Repository**

Git 🡪Download Git

GitHub 🡪 Create account in GitHub

Jenkins 🡪Download Generic Java Package (.war) file and keep it in some folder

Go to that location in cmd: java –jar Jenkins.war