Selenium Hybrid Framework

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

Step 1 : Create new Project and install required packages/libraries

- selenium : selenium libraries
- pytest : python unit test framework
- pytest-html : pytest HTML report
- pytest-xdist : to run tests parallel
- openpyxl : MS excel support
- webdriver-manager : to get the latest web drivers for each browser
- allure-pytest : to generate allure reports

```
pip3 install selenium pytest pytest-html pytest-xdist openpyxl webdriver-manager allure-pytest
```

Step 2 : Create Folder Structure

Project Name

|

pageObjects (package)

|

tests (package)

|

utilities (package)

|

TestData (folder)

|

Configurations (folder)

|

Logs (folder)

|

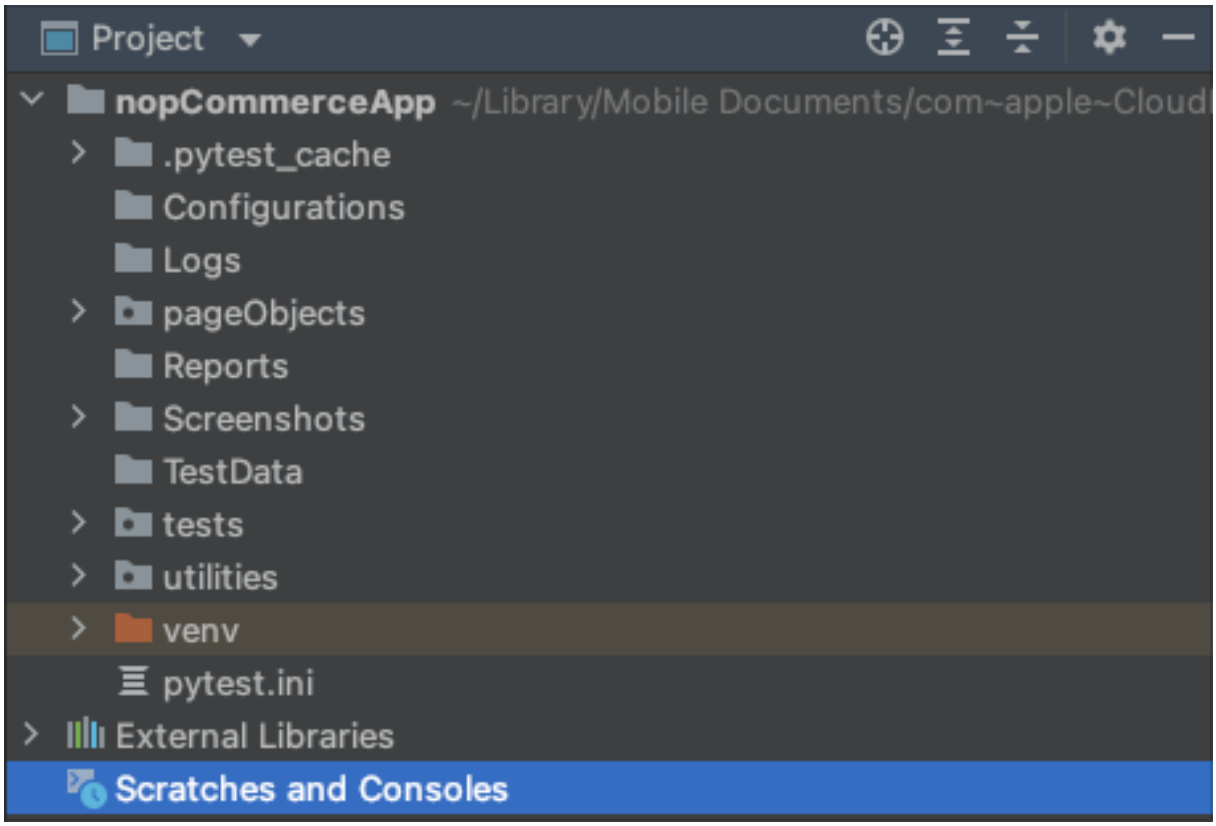
Screenshots (folder)

|

Reports (folder)

|

run batch



Step 3 : Automating Login Test Case

- Create LoginPage object class under “pageObjects”
- Create LoginTests class under “tests” and write relevant code
- Create conftest.py under “tests” and fill the relevant code
-

Step 4 : Capture screenshot on failure

- Update LoginTests with screenshot under “tests”

```
self.driver.save_screenshot("Screenshots/test_name.png")
```

Step 5 : Read common values from ini file:

- Add “config.ini” file in “Configurations” folder to keep all common info
- Create “readProperties.py” utility file under utilities package to read common
- Replace hard-coded values from LoginTests class in “tests”

Step 6 : Adding logs to tests

- Add customLogger.py under utilities package
- Add logs to LoginTests in “tests”

Step 7 : Run tests on desired browser / cross browser / parallel

- update conftest.py with required fixtures which will accept command line argument (browser)
- pass browser name as argument in command line

Run test on desired browser

- `pytest -v --browser chrome`
- `pytest -v --browser firefox`

Run tests parallel

- `pytest -v -n3 --browser chrome`
- `pytest -v -n2 --browser firefox`

Step 8 : Generate pytest html report

- Update conftest.py with pytest hooks
- To generate html report run below command

```
pytest -v -n2 --browser chrome --html=Reports/reports.html
```

Step 9 : Automating Data Driven test case

- Prepare test data in excel sheet and place the excel sheet inside TestData folder
- Create ‘ExcelUtils.py’ utility class under utility package
- Create LoginDataDrivenTests under “tests”
- Run the test case

Step 10 : Adding test cases

Step 11 : Grouping Tests

- Grouping markers (adding markers to every test method)
  - @pytest.mark.sanity
  - @pytest.mark.regression
- Add markers entries in pytest.ini file

```
[pytest]
markers =
  sanity
  regression
```

- Select groups at run time

```
-m "sanity"
-m "regression"
-m "sanity and regression"
-m "sanity or regression"
```

Step 12 : Run the test using .bat or .sh which is already done in the below link

- py_selenium/pySelenium_Framework_2/execute at main · bunnycodec/py_selenium · GitHub

---

Step 13 : Push the code to Git and Github

Step 14 : Run Jenkins Tests

#python/selenium