**Python, selenium, pytest, page object model, html report**

**Step1. Install-->**

Selenium

Pytest

Pytest-html

Pytest-xdist(running tcs parallely)

Openpyxl

**Step2. Folder structure --->**

Projectname

* Pageobjects(package)
* Testcases(package)
* Utilities(package)
* Testdata(folder)
* Configurations(folder)
* Logs(folders)
* Screenshots(folders)
* reports(folder)
* Run.bat

**Step3. Automating Login test case**

* Create loginpage object class under 'pageobjects'
  + Create a class, get all locator assign it to class variable
  + Create constructor Pass driver as instance variable
  + Create method for each action pass appropriate arguments wherever needed

* Create logintest under 'testCases'
  + Import the pageobject class, import selenium webdriver
  + Create a class with test id and name
  + Create all common used variable
  + Create launch browser statements
  + Call all the pageobj class by creating the class
  + Verify the title and add assertion

* Create conftest.py under 'testCases'
  + Create a fixture method
  + And put all the common setup methods like launch browser, close in fixture method

[remove webdriver.chorme() in testcase.py and add it in conftest.py ]

**Step4. Capture screenshot on failure**

* Update login test with screenshots under "textCases"
  + Add the save screenshot methods before assert failure
  + And then close the browser. [put driver.close() before all the assert statement]

**Step5. Read common values from .ini files**

* Add 'config.ini' file in 'Configurations' folder

To avoid the hardcode value keep all variable in .ini files[this is not test data]

* Create 'readProperty.py' utility file under utilities package to read common data
  + Read data from .ini -> provide data to test cases
* Replace hardcoded values in login test cases

**Step6. Adding logs to test case**

* Add customerLogger.py under utilities package
* Add logs to Login test cases

**Step7. Run Tests on Desired Browser/Cross Browser/Parallel**

* Update 'contest.py' with required fixture which will accept command line arguments (browser)
* Pass browser name as arguments in command line
  + Modify the conftest file add below lines
  + <https://www.ontestautomation.com/pytest-and-custom-command-line-arguments/>

def pytest\_addoption(parser): # this will get the values from CLI/hooks

parser.addoption("--browser")

@pyest.fixture()

def browser(request): # this will return browser value to setup method

print(request.config.getoption("--browser"))

return request.config.getoption("--browser")

**To run test on desired browser**

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

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

**To run tests parallely**

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

Pytest -s -v -n=3 testCases/test\_login.py --browser firefox

\*\*\*It uses pytest-xdist module

**Step8. Generate pytest HTML Reports[Completion percentage-70%]**

* Update conftest.py with pytest hooks
* To generate HTML report run below command:

Pytest -s -v -n=3 --html=Reports\reports.html testCases/test\_login.py --browser firefox

\*\*\*it uses pytest-html

def pytest\_configure(config):

config.\_metadata={

"Tester":"shirly",

"ModuleName":"customers",

"ProjectName":"nopcommerce",

}

Add in contest.py

* pytest -v -s --capture=tee-sys --html=.\reports\report.html testCases\Test\_loginPage.py
* To captue logger info

**Step9. Automating Data driven test cases[Completion percentage-90%]**

* 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 test cases

**Step10. Adding new test cases(if framework is already ready this is your first step)**

* Add new customer using random password generator
* Search customer using name
* Search customer using mail id

**Step11. Grouping Tests**

@pytest.mark.sanity🡪 Functionality basic

@pytest.mark.regression-🡪 Functionality end-to end scenarios

pytest -s -v -m "sanity" --html=Reports\reports.html testCases --browser chrome

pytest -s -v -m "sanity and regression" --html=Reports\reports.html testCases --browser chrome

pytest -s -v -m "sanity or regression" --html=Reports\reports.html testCases --browser chrome

pytest -s -v -m "regression" --html=Reports\reports.html testCases --browser chrome

**Step12. create Run.cmd for one click run**

**.bat or .cmd windows**

**.sh file for linux**

**Step13: push the code to github repo**