YOU TUBE LINK: <https://youtu.be/57pjD89IFXA>

<https://www.youtube.com/watch?v=aVr0P2EE3zo&list=PLUDwpEzHYYLt2RzOb-_eafLAP0VSoyJhf&index=4>

GIT BRANCH NAME: <https://github.com/sainiashu/sel_pytest_python_pom_html_v2>

Install selenium : <https://pypi.org/project/selenium/>

pip install selenium

install python:

<https://pypi.org/project/pytest/>

pip install pytest

install html report plugin: <https://pypi.org/project/pytest-html/>

pip install pytest-html

Parallel execution: <https://pypi.org/project/pytest-xdist/>

pip install pytest-xdist

Create a tests folder for executable tests

**test\_google\_search.py**

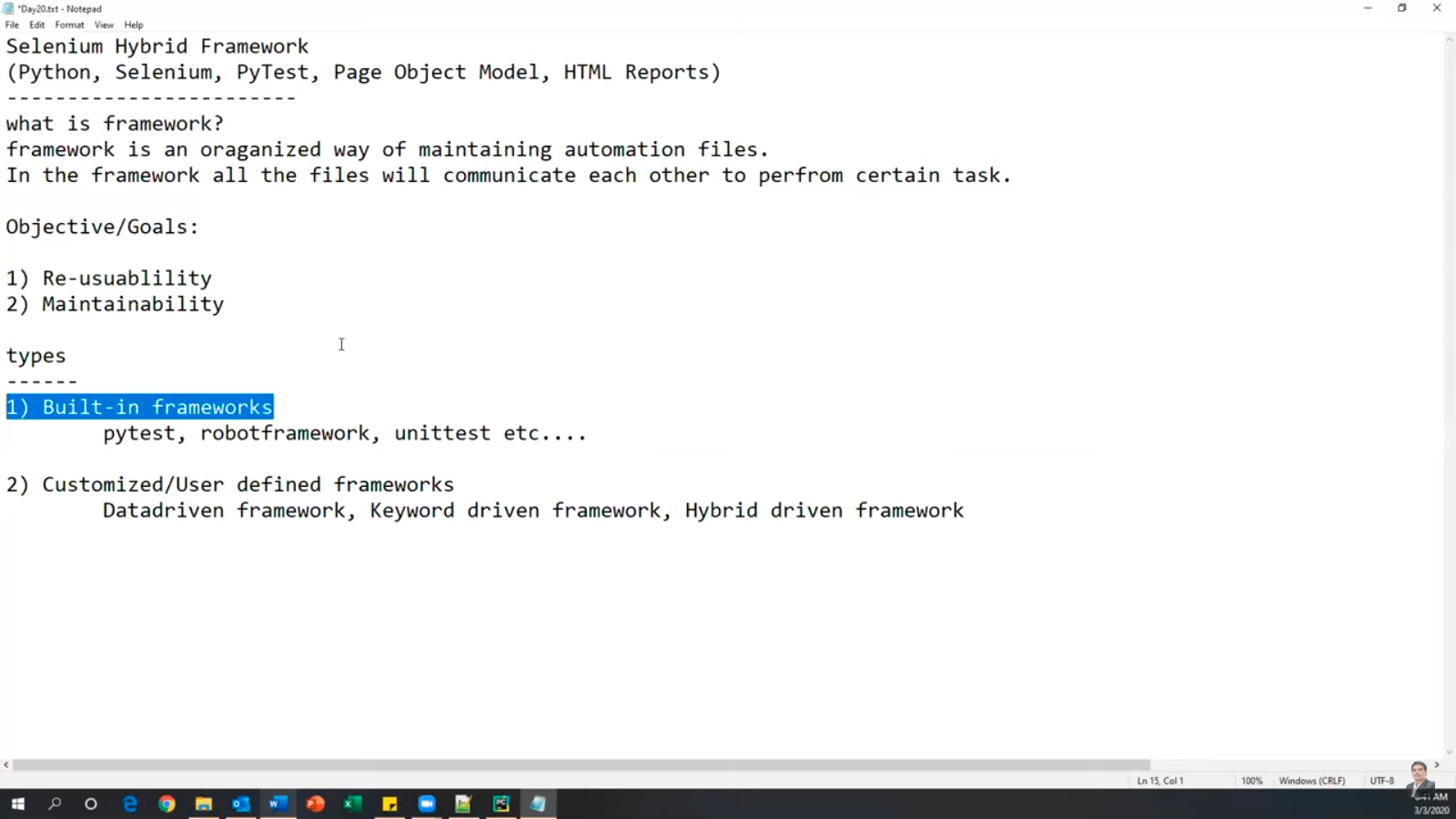
import time  
  
import pytest  
  
from selenium import webdriver  
from selenium.webdriver import Keys  
from selenium.webdriver.common.by import By  
  
  
search\_keywords = [  
 "test1",  
 "test2",  
 "test"  
]  
  
@pytest.mark.parametrize("search\_keyword", search\_keywords)  
class TestGoogleSearch:  
 def test\_search(self,browser,search\_keyword):  
 driver = browser  
 driver.get("https://www.google.com/")  
 search\_box = driver.find\_element(By.ID,"APjFqb")  
 search\_box.send\_keys(search\_keyword)  
 search\_box.send\_keys(Keys.RETURN)  
 time.sleep(5)  
  
 assert search\_keyword in driver.title  
  
 print(f"Search for : {search\_keyword}")

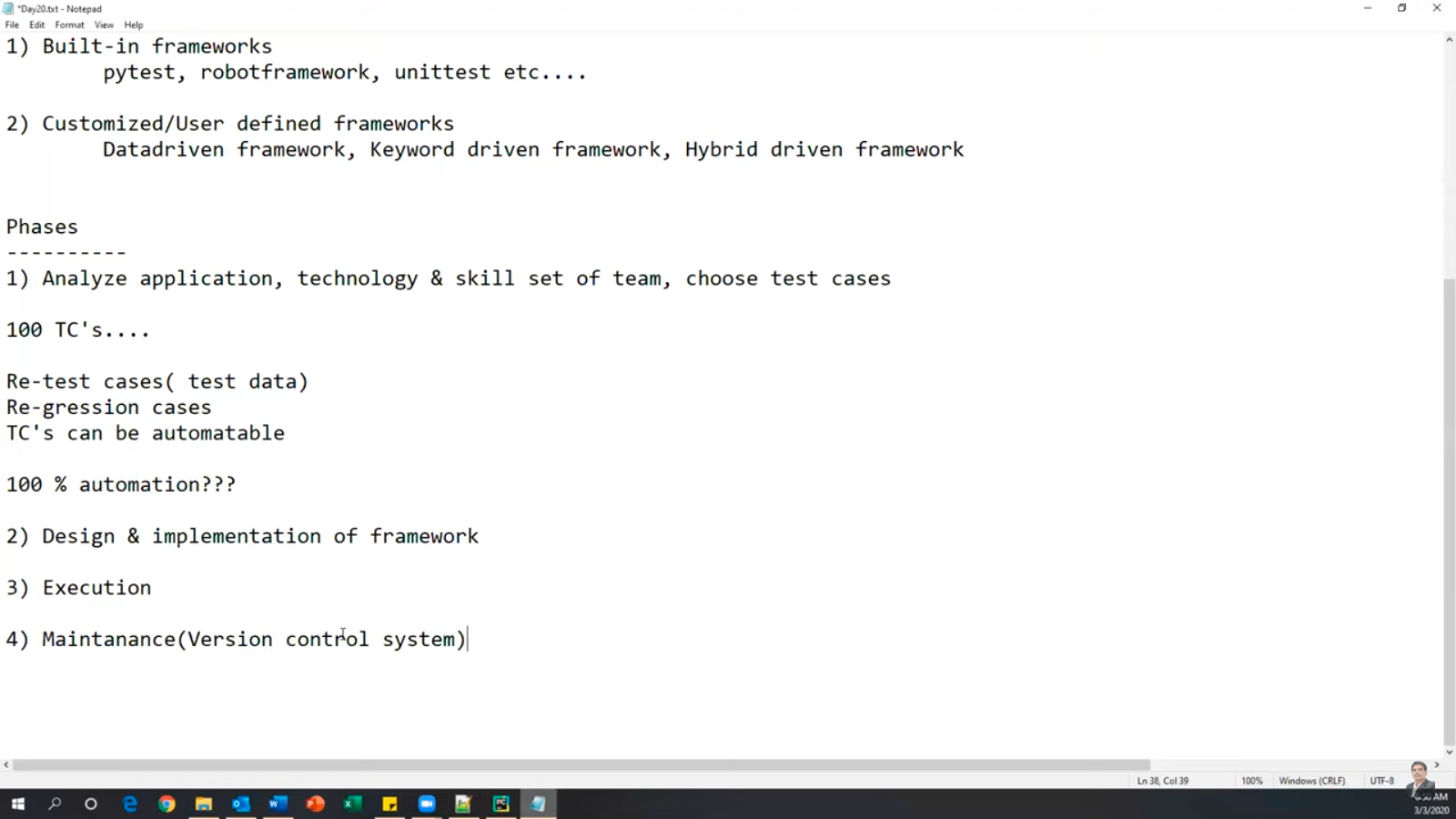
Create **conftest.py** file for configuration

import pytest  
  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
  
@pytest.fixture(scope="class")  
def browser():  
 driver = webdriver.Chrome()  
 driver.maximize\_window()  
  
 yield driver  
  
 driver.quit()

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

YOUTUBE : <https://youtu.be/57pjD89IFXA>





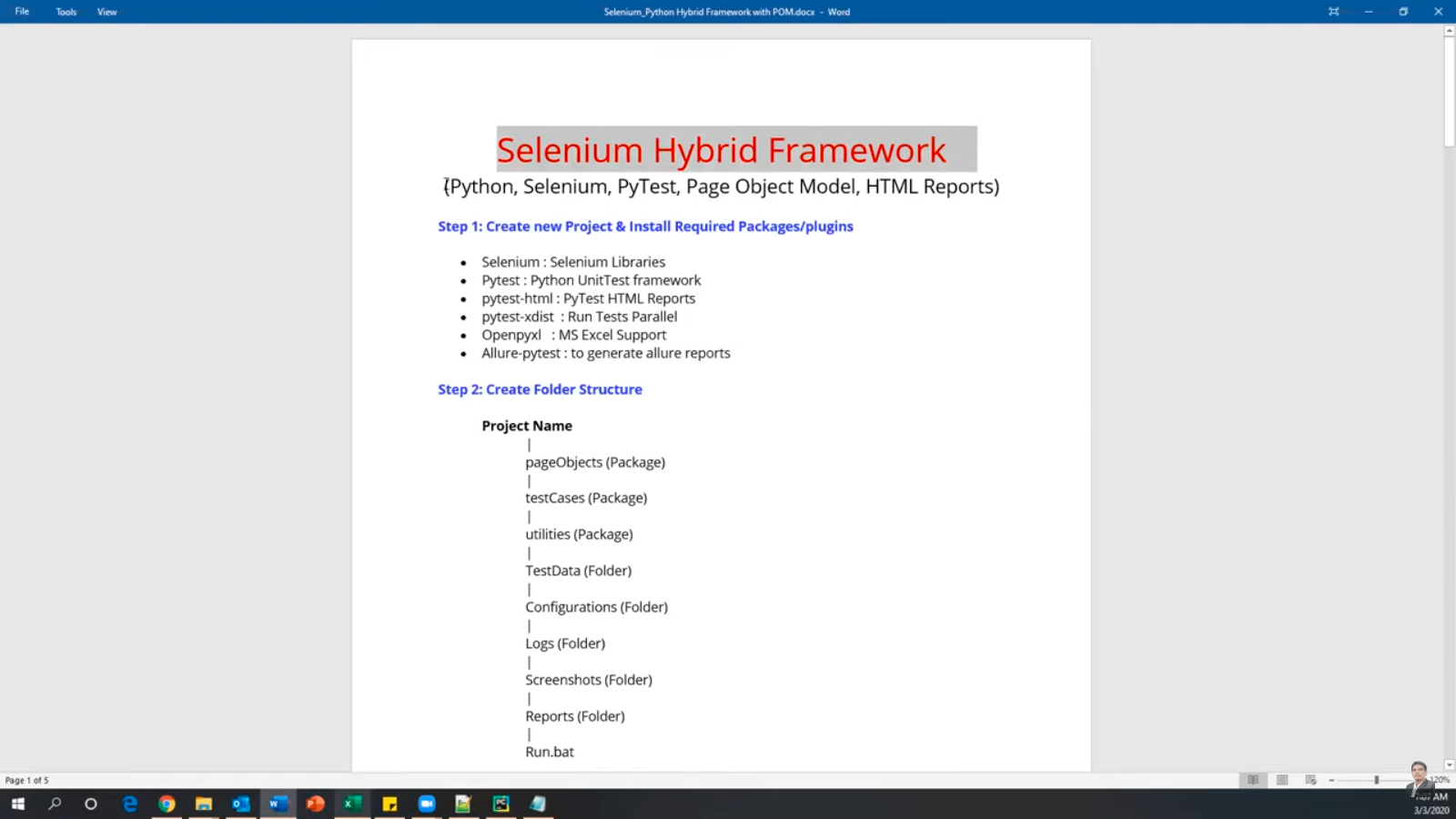
**How to identify what type of test case we need to automate?**

Sol: If I have 100 test cases then we need to do following things:

1. Re-test case ( test data) – test data driven that uses multiple test using the same test case
2. Regresssion test case: those are required to test after every build
3. Test cases can be automatable

**Is test case 100% can be automated?**

Sol: All test case cannot be automated



Install all the required libraries

<https://youtu.be/57pjD89IFXA>

**Go to the Files->Settings->Project[Your Project Name]-> Project interpreters->Click on “+” and enter selenium in search text box and click on install button**

**After installation save and close the window**

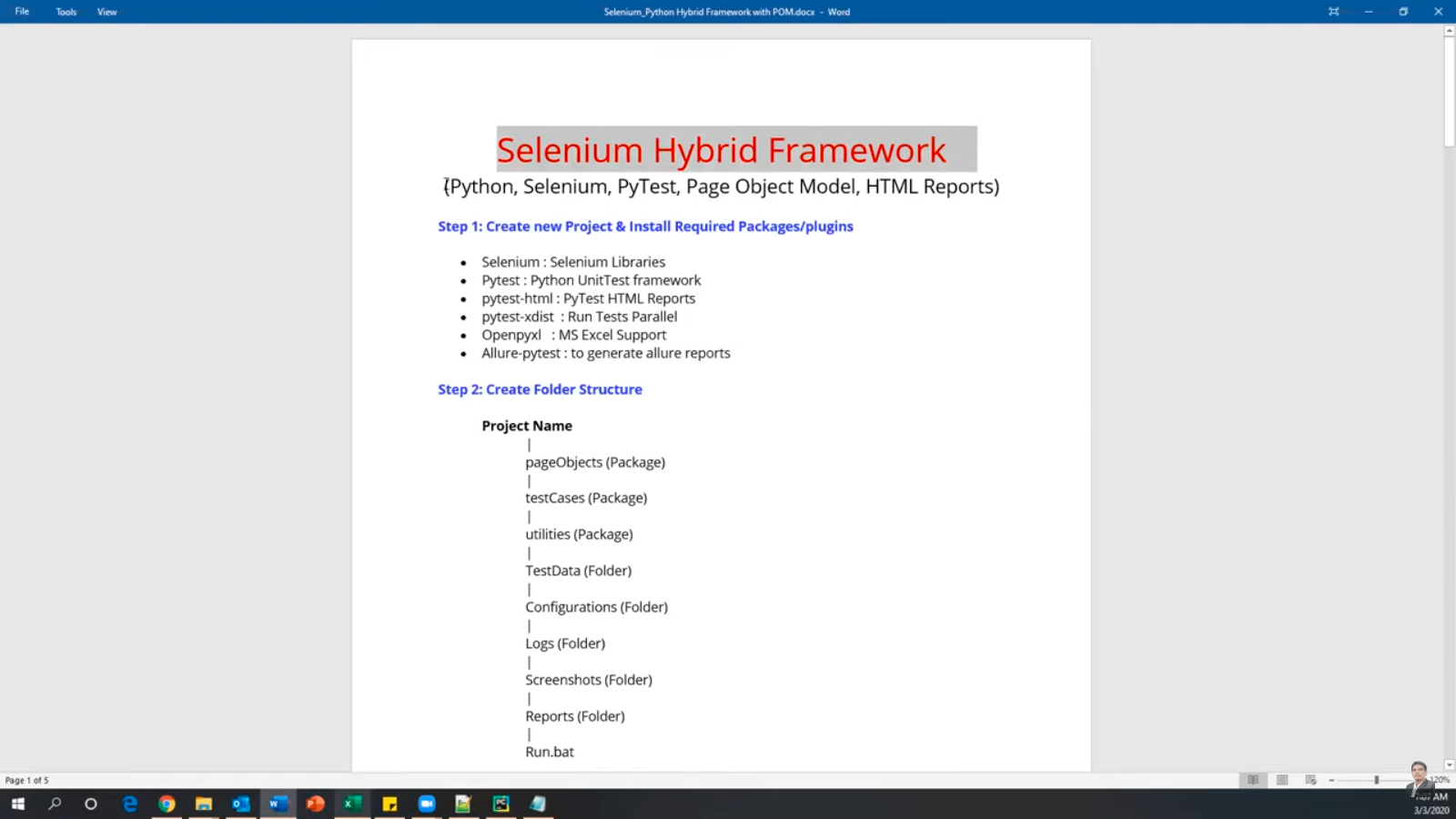
**Demo Application:**

[**https://demo.nopcommerce.com/**](https://demo.nopcommerce.com/)

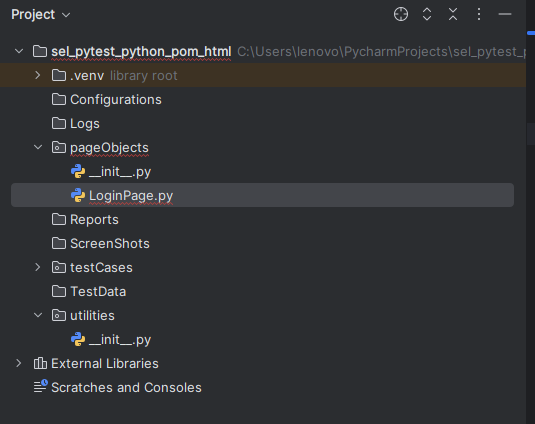
**Admin Page:** [**https://admin-demo.nopcommerce.com/login?ReturnUrl=%2Fadmin%2F**](https://admin-demo.nopcommerce.com/login?ReturnUrl=%2Fadmin%2F)

**Frontend Page:** [**https://demo.nopcommerce.com/**](https://demo.nopcommerce.com/)

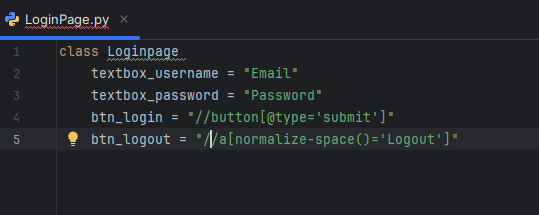
**Create a complete structure according to the below image**

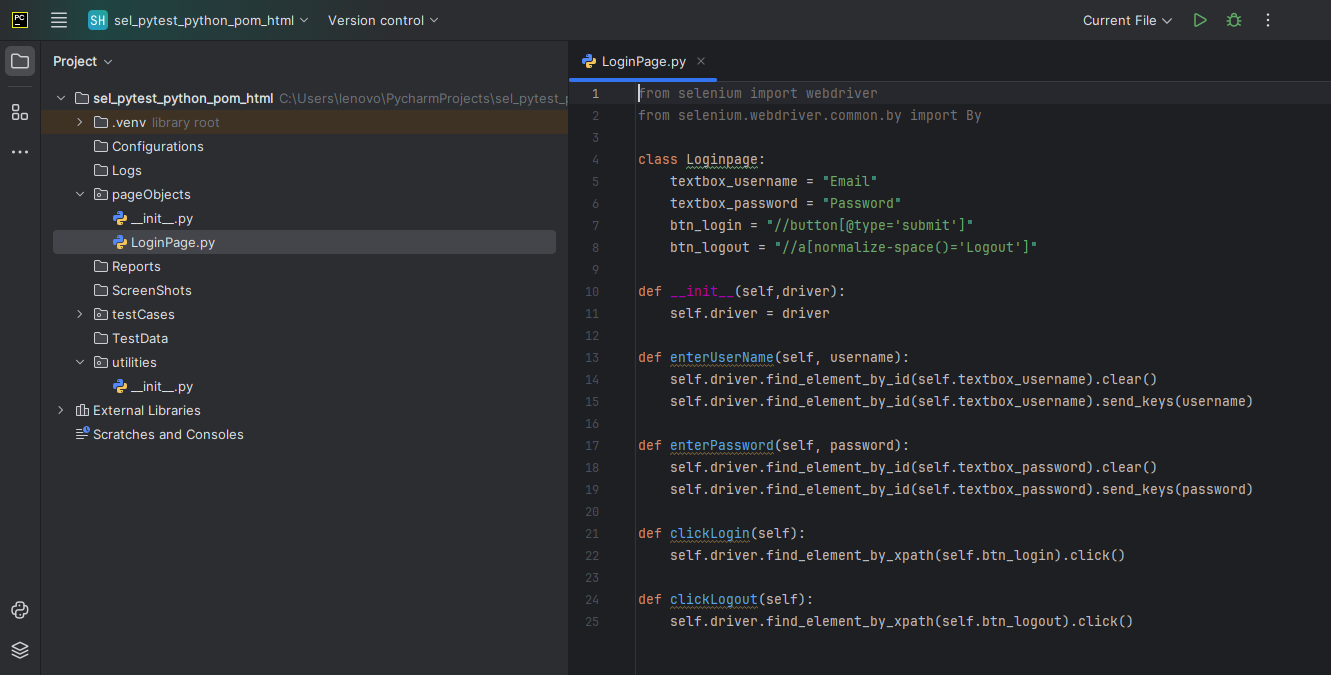


**Please make sure pageobject, testcases and utilities should be a python packages and rest of the items should be a folders**



**Now we need to create first test and before that we need to create the page objects for the particular page**

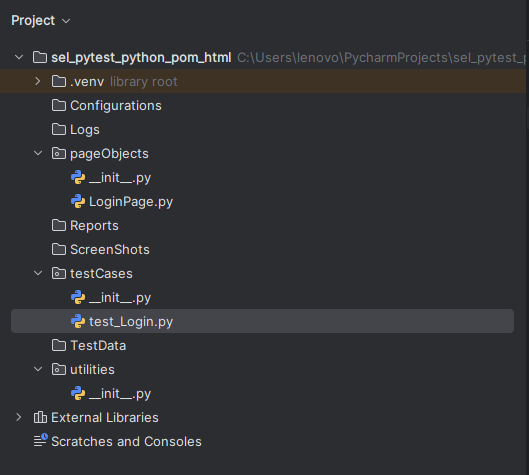
1. **Create a python file under pageobject python package in the name “LoginPage”**
2. **Then create a class LoginPage inside the file**
3. **After that define all the find elements of the elements of the pages like below**
4. 
5. **Now we going to initialize the class with the help of python constructor**
6. def \_\_init\_\_(self,driver): //python constructor for the class  
    self.driver = driver //initialize driver for the test
7. **First Page Object Class: LoginPage**



**Code**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
  
class Loginpage:  
 textbox\_username = "Email"  
 textbox\_password = "Password"  
 btn\_login = "//button[@type='submit']"  
 btn\_logout = "//a[normalize-space()='Logout']"  
  
def \_\_init\_\_(self,driver):  
 self.driver = driver  
  
def enterUserName(self, username):  
 self.driver.find\_element\_by\_id(self.textbox\_username).clear()  
 self.driver.find\_element\_by\_id(self.textbox\_username).send\_keys(username)  
  
def enterPassword(self, password):  
 self.driver.find\_element\_by\_id(self.textbox\_password).clear()  
 self.driver.find\_element\_by\_id(self.textbox\_password).send\_keys(password)  
  
def clickLogin(self):  
 self.driver.find\_element\_by\_xpath(self.btn\_login).click()  
  
def clickLogout(self):  
 self.driver.find\_element\_by\_xpath(self.btn\_logout).click()

**CREATE FIRST TEST CASE**

1. **Create a python file “test\_login.py” under testcases python package**
2. 
3. **Create a class name inside the test\_Login.py file**
4. import pytest  
     
   from selenium import webdriver  
   from pageObjects.LoginPage import LoginPage  
     
   class Test\_001\_Login:  
    baseURL= "https://admin-demo.nopcommerce.com/login?ReturnUrl=%2Fadmin%2F"  
    username = "admin@yourstore.com"  
    password = "admin"  
     
    def test\_homePageTitle(self):  
    self.driver = webdriver.Chrome()  
    self.driver.get(self.baseURL)  
    get\_title = self.driver.title  
    self.driver.close()  
    if get\_title == "nopCommerce demo store. Login":  
    assert True  
    else:  
    assert False  
     
    def test\_login(self):  
    self.driver = webdriver.Chrome()  
    self.driver.get(self.baseURL)  
    self.lp = LoginPage(self.driver)  
    self.lp.enter\_user\_name(self.username)  
    self.lp.enter\_password(self.password)  
    self.lp.click\_login()  
    act\_title = self.driver.title  
    self.driver.close()  
    if act\_title == "Dashboard / nopCommerce administration":  
    assert True  
    else:  
    assert False

**OR WITH THE FIXTURE LOGIC**

import pytest  
  
from selenium import webdriver  
from pageObjects.LoginPage import LoginPage  
  
class Test\_001\_Login:  
 baseURL= "https://admin-demo.nopcommerce.com/login?ReturnUrl=%2Fadmin%2F"  
 username = "admin@yourstore.com"  
 password = "admin"  
  
 def test\_homepage\_title(self,setup):  
 self.driver = setup  
 self.driver.get(self.baseURL)  
 get\_title = self.driver.title  
 self.driver.close()  
 if get\_title == "nopCommerce demo store. Login":  
 assert True  
 else:  
 assert False  
  
 def test\_login(self,setup):  
 self.driver = setup  
 self.driver.get(self.baseURL)  
 self.lp = LoginPage(self.driver)  
 self.lp.enter\_user\_name(self.username)  
 self.lp.enter\_password(self.password)  
 self.lp.click\_login()  
 act\_title = self.driver.title  
 self.driver.close()  
 if act\_title == "Dashboard / nopCommerce administration":  
 assert True  
 else:  
 assert False

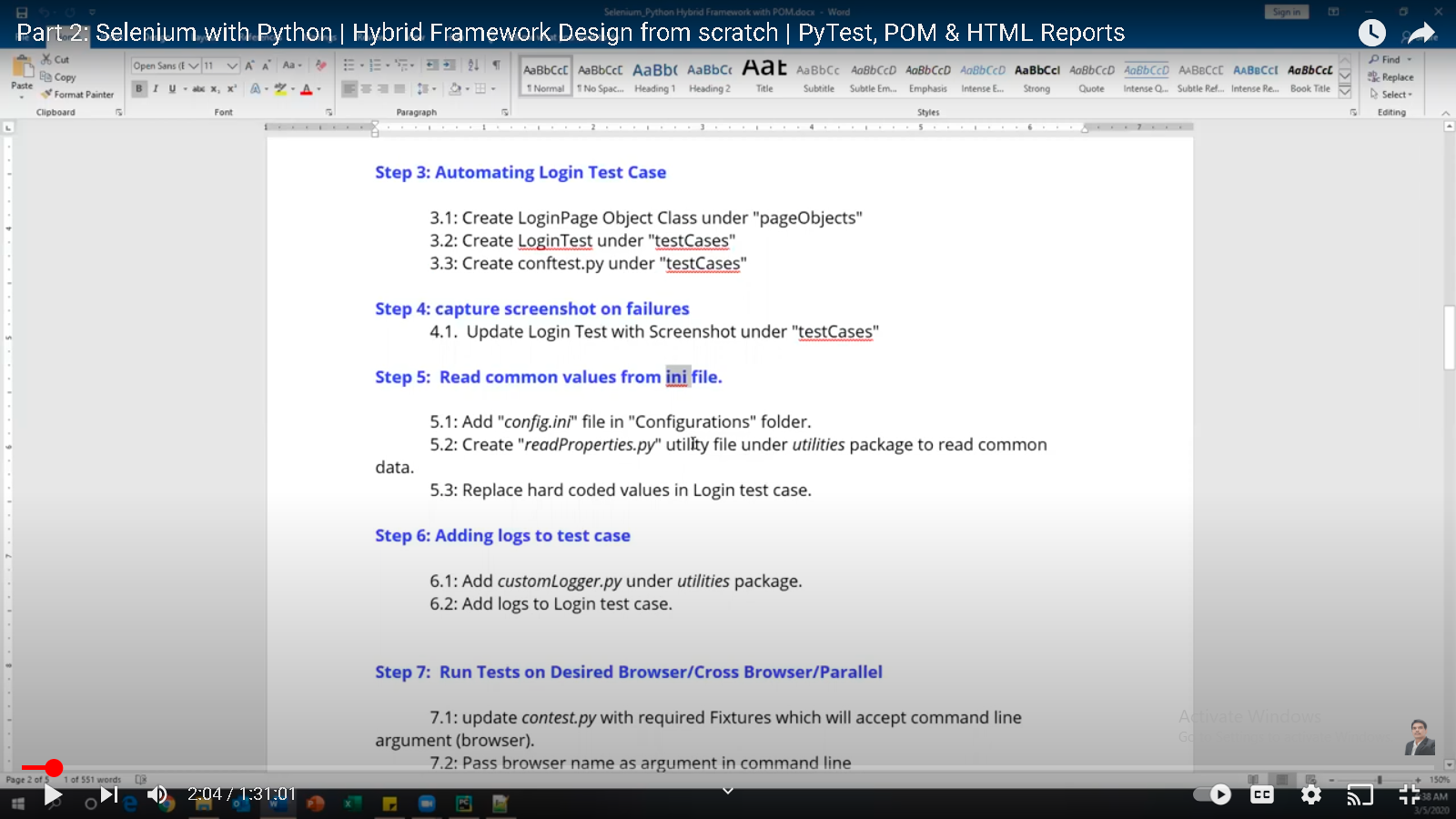
**CREATE ONE FILE UNDER TEST CASES FOLDER CONFTEST.PY**

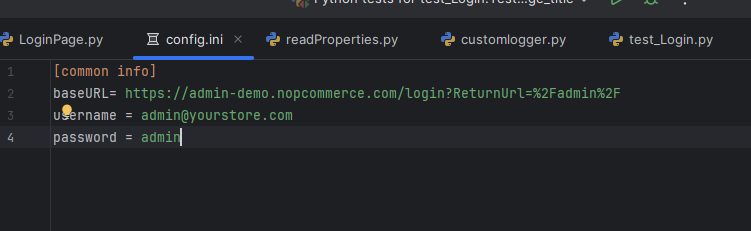
from selenium import webdriver  
import pytest  
  
@pytest.fixture()  
def setup():  
 driver = webdriver.Chrome()  
 return driver

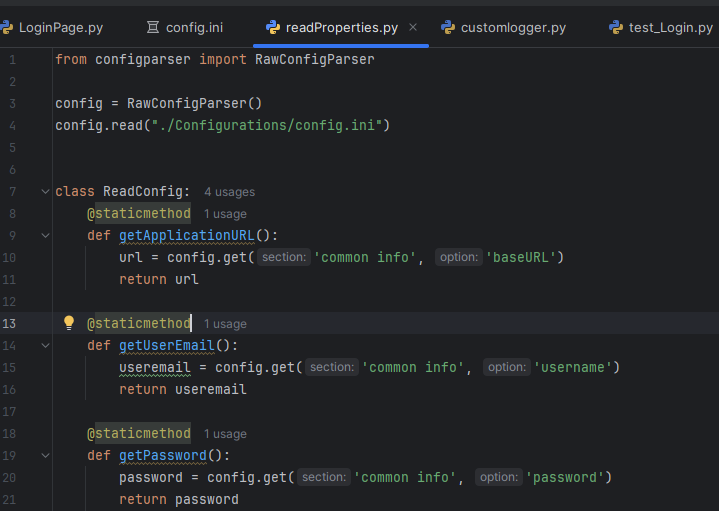
**FINAL CODE WITH FOLLOWING DETIALS**

import pytest  
  
from selenium import webdriver  
from pageObjects.LoginPage import LoginPage  
  
class Test\_001\_Login:  
 baseURL= "https://admin-demo.nopcommerce.com/login?ReturnUrl=%2Fadmin%2F"  
 username = "admin@yourstore.com"  
 password = "admin"  
  
 def test\_homepage\_title(self,setup):  
 self.driver = setup  
 self.driver.get(self.baseURL)  
 get\_title = self.driver.title  
  
 if get\_title == "nopCommerce demo store. Login111":  
 assert True  
 self.driver.close()  
 else:  
 self.driver.save\_screenshot(".\\ScreenShots\\" + "test\_homepage\_title.png")  
 self.driver.close()  
 assert False  
  
 def test\_login(self,setup):  
 self.driver = setup  
 self.driver.get(self.baseURL)  
 self.lp = LoginPage(self.driver)  
 self.lp.enter\_user\_name(self.username)  
 self.lp.enter\_password(self.password)  
 self.lp.click\_login()  
 act\_title = self.driver.title  
  
 if act\_title == "Dashboard / nopCommerce administration":  
 assert True  
 self.driver.close()  
 else:  
 self.driver.save\_screenshot(".\\ScreenShots\\" + "test\_login.png")  
 self.driver.close()  
 assert False

**READ COMMON DATA FROM THE config.ini file**

1. 
2. **Create config,ini file under Configurations folder and file type is just a basic file it NOTA PYTHON FILE.**
3. **It contains all user pre-requisite test data that use for the entire test suites**
4. [common info]  
   baseURL= https://admin-demo.nopcommerce.com/login?ReturnUrl=%2Fadmin%2F  
   username = admin@yourstore.com  
   password = admin



1. **Create another file name readProperties and define the functions**
2. from configparser import RawConfigParser  
     
   config = RawConfigParser()  
   config.read("./Configurations/config.ini")  
     
     
   class ReadConfig:  
    @staticmethod  
    def getApplicationURL():  
    url = config.get('common info', 'baseURL')  
    return url  
     
    @staticmethod  
    def getUserEmail():  
    useremail = config.get('common info', 'username')  
    return useremail  
     
    @staticmethod  
    def getPassword():  
    password = config.get('common info', 'password')  
    return password
3. 

**NOW WE NEED TO RUN OUR TEST IN DIFFERENT BROWSER USING COMMAND LINE**

1. **UPDATE OUR CONFTEST.PY FILE UNDER TESTCASE FOLDER**
2. from selenium import webdriver  
   import pytest  
     
   @pytest.fixture()  
   def setup(browser):  
    if browser == 'chrome':  
    driver = webdriver.Chrome()  
    elif browser == 'firefox':  
    driver = webdriver.Firefox()  
    return driver  
     
   def pytest\_addoption(parser): #This will get the value from the cli/hook  
    parser.addoption("--browser")  
     
   @pytest.fixture()  
   def browser(request): #This will return the browser value to setup method  
    return request.config.getoption("--browser")
3. **FOR PARALLEL EXECUTION YOU CAN RUN THE BELOW COMMAND**
4. **pytest -v -s -n=2 .\testCases\test\_Login.py --browser chrome**

**GENERATE THE HTML REPORT:**

**pytest -v -s --html=.\Reports\repot.html .\testCases\test\_Login.py --browser chrome**

**FILE NAME:Conftest.py**

from selenium import webdriver  
import pytest  
  
@pytest.fixture()  
def setup(browser):  
 if browser == 'chrome':  
 driver = webdriver.Chrome()  
 elif browser == 'firefox':  
 driver = webdriver.Firefox()  
 return driver  
  
def pytest\_addoption(parser): #This will get the value from the cli/hook  
 parser.addoption("--browser")  
  
@pytest.fixture()  
def browser(request): #This will return the browser value to setup method  
 return request.config.getoption("--browser")  
  
########## It is hook for adding environment info to html report  
def pytest\_configure(config):  
 if hasattr(config, 'metadata'):  
 config.metadata['Project Name'] = 'Pytest Learning'  
 config.metadata['Module Name'] = 'Login'  
 config.metadata['Tester Name'] = 'Demo'  
  
#it is hook for delete and modify the html report  
#@pytest.mark.optionalhook  
@pytest.hookimpl(optionalhook=True)  
def pytest\_metadata(config):  
 if hasattr(config, 'metadata'):  
 config.metadata.pop("JAVA\_HOME", None)  
 config.metadata.pop("Plugins",None)

**TEST DATA DRIVEN**

**READ DATA FROM EXCEL, CREATE UTILITIES FILES**

**FILE NAME: test\_login\_ddt.py**

import time  
import pytest  
  
from selenium import webdriver  
from pageObjects.LoginPage import LoginPage  
from utilities.XLUtils import getRowCount  
from utilities.readProperties import ReadConfig  
from utilities.customlogger import LogGen  
from utilities import XLUtils  
  
class Test\_002\_ddt\_Login:  
  
 baseURL = ReadConfig.getApplicationURL()  
 path = ".//TestData//logintestdata.xlsx"  
  
 username = ReadConfig.getUserEmail()  
 password = ReadConfig.getPassword()  
 logger = LogGen.loggen()  
  
  
 def test\_login(self,setup):  
 self.logger.info("\*\*\*\*\* Test Passed: Test\_001\_Login \*\*\*\*\* ")  
 self.driver = setup  
 self.driver.get(self.baseURL)  
  
 self.lp = LoginPage(self.driver)  
  
 self.getRowCount = XLUtils.getRowCount(self.path,"Sheet1")  
 print("Number of Rows in a excel:", self.getRowCount)  
  
 lst\_status=[] #Empty list  
  
 for r in range(2,self.getRowCount+1):  
 self.user = XLUtils.readData(self.path,"Sheet1",r,1)  
 self.password = XLUtils.readData(self.path,"Sheet1",r,2)  
 self.exp = XLUtils.readData(self.path, "Sheet1", r, 3)  
  
 self.lp.enter\_user\_name(self.user)  
 self.lp.enter\_password(self.password)  
 self.lp.click\_login()  
 time.sleep(5)  
  
 act\_title = self.driver.title  
 exp\_title = "Dashboard / nopCommerce administration"  
  
 if act\_title == exp\_title:  
 if self.exp == "Pass":  
 self.logger.info("Test Passed")  
 self.lp.click\_logout()  
 lst\_status.append("Pass")  
 # elif self.exp == "Fail":  
 #  
 # self.logger.info("\*\*\* failed \*\*")  
 # self.lp.click\_logout()  
 # lst\_status.append("Fail")

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Page Object file: AddCustomerRolesPage.py**

from selenium.webdriver.common.by import By  
  
class AddCustomerRoles:  
 lnkCustomerMenu = (By.XPATH, "/html[1]/body[1]/div[3]/aside[1]/div[1]/nav[1]/ul[1]/li[4]/a[1]/p[1]")  
 lnkCustomerRoles = (By.XPATH, "//p[normalize-space()='Customer roles']")  
 btnAddNew = (By.XPATH, "//i[@class='fas fa-square-plus']")  
 # txtName = "Name"  
 txtName = (By.ID,"Name")  
 btnSave = (By.XPATH, "//button[@name='save']//i[@class='far fa-floppy-disk']")  
  
 def \_\_init\_\_(self,driver):  
 self.driver = driver  
  
 def clickOnCustomerMenu(self):  
 self.driver.find\_element(\*self.lnkCustomerMenu).click()  
  
 def clickOnCustomerRoles(self):  
 self.driver.find\_element(\*self.lnkCustomerRoles).click()  
  
 def clickOnAddNew(self):  
 self.driver.find\_element(\*self.btnAddNew).click()  
  
 def enterName(self, name):  
 self.driver.find\_element(\*self.txtName).send\_keys(name)  
  
 def clickOnSave(self):  
 self.driver.find\_element(\*self.btnSave).click()

**Test Class: test\_AddCustomerRoles.py**

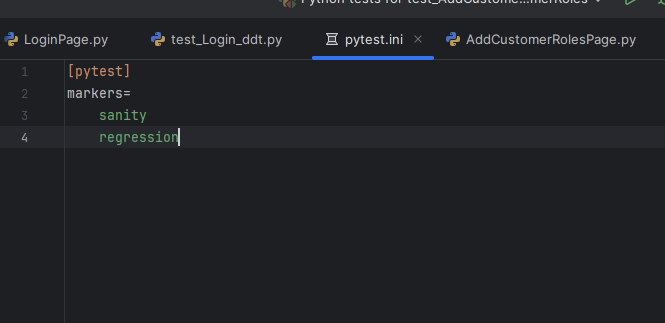
import logging  
  
from pageObjects.LoginPage import LoginPage  
from utilities.readProperties import ReadConfig  
from utilities.customlogger import LogGen  
from pageObjects.AddCustomerRolesPage import AddCustomerRoles  
  
class Test\_003\_CustomerRoles:  
 baseURL = ReadConfig.getApplicationURL()  
 username= ReadConfig.getUserEmail()  
 password = ReadConfig.getPassword()  
  
 logger = LogGen.loggen()  
  
 def test\_AddCustomerRoles(self,setup):  
 self.logger.info("\*\*\*\*\* Started: Test\_003\_CustomerRoles \*\*\*\*\*")  
 self.driver = setup  
  
 self.driver.get(self.baseURL)  
 self.driver.maximize\_window()  
  
 self.lp = LoginPage(self.driver)  
 self.lp.enter\_user\_name(self.username)  
 self.lp.enter\_password(self.password)  
 self.lp.click\_login()  
 self.logger.info("\*\*\*\* Login Successful \*\*\*\*")  
  
 self.logger.info("\*\*\* Start Test\_003\_CustomerRoles \*\*\*\*")  
  
 self.addcustomerroles= AddCustomerRoles(self.driver)  
 self.addcustomerroles.clickOnCustomerMenu()  
 self.addcustomerroles.clickOnCustomerRoles()  
 self.addcustomerroles.clickOnAddNew()  
 self.addcustomerroles.enterName("user1")  
 self.addcustomerroles.clickOnSave()  
 # // div[ @  
  
 # class ='alert alert-success alert-dismissable']

**COMMAND: pytest -v -s --html=.\Reports\repot.html .\testCases\test\_AddCustomerRoles.py --browser chrome**

**GROUPING THE TEST “@pytest.mark.customename”**

@pytest.mark.sanity  
@pytest.mark.regression

**WE NEED TO ADD CUSTOM MARKER THEREFORE WE NEED TO CREATE NEW BASIC FILE UNDER TESTCASE FOLDER IN THE NAME OF pytest.ini**



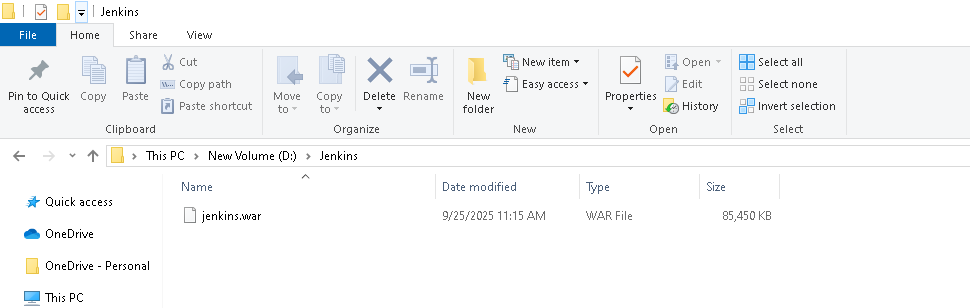
**COMMAND: pytest -v -s -m "sanity" --html=./Reports/report.html testCases/ --browser=chrome**

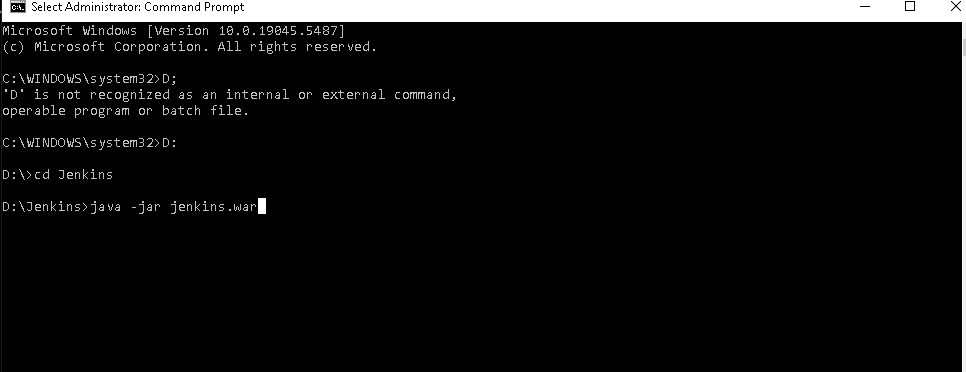
**CREATE A RUN.BAT FILE**

**GIT HUB DOCS**

**JENKINS INSTALLATION:** [**https://www.jenkins.io/download/**](https://www.jenkins.io/download/)

**Download the .war file for temp practice and place in oneparticular location:**

 **after that open a command prompt and move the Jenkins location and pass the command and hit enter to run the Jenkins:**

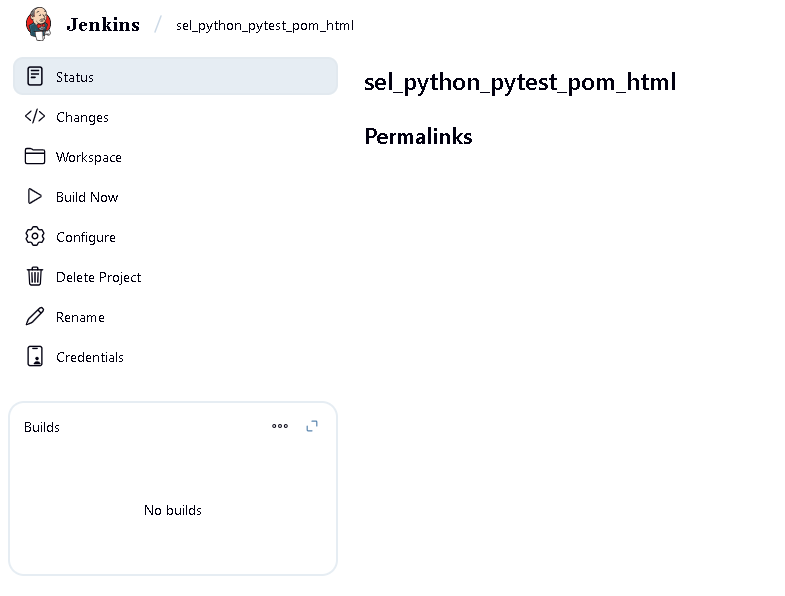
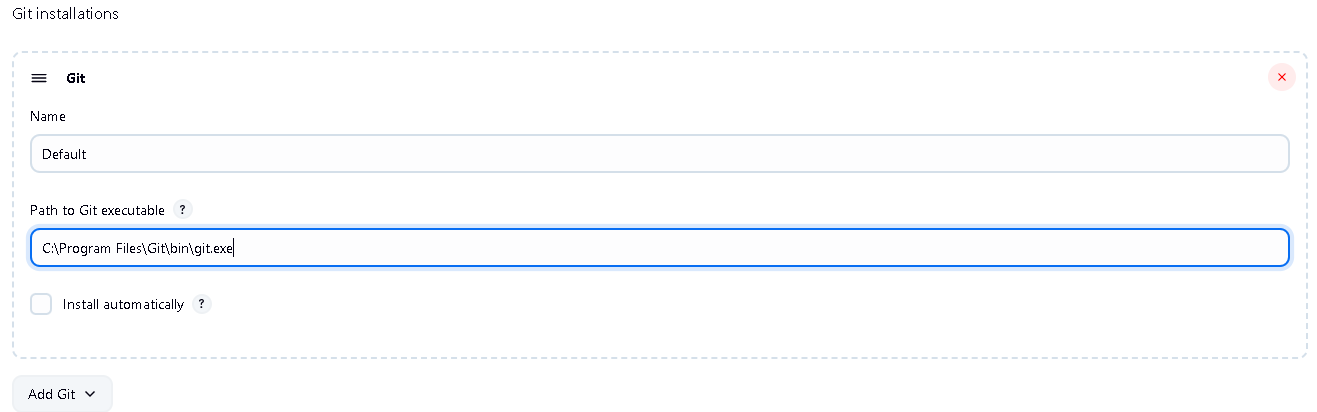


**You need to enter the password which you can see the command prompt**

**Selec the suggestsed plugin and continue.**

**It will takesoem time andsetup the Jenkins**

**NOW HOW TO RUN YOUR GITHUB PROJECT INTO THE JENKINS**

1. **Click on a new item and create a name and select the free style project**
2. **Project Name: sel\_python\_pytest\_pom\_html**
3. **Click on OK button**
4. **Donot enter anything above just go to the source control management section and select the gitcheck box**
5. [**https://github.com/sainiashu/sel\_pytest\_python\_pom\_html\_v2.git**](https://github.com/sainiashu/sel_pytest_python_pom_html_v2.git)
6. 
7. **Keep it master branch do not changeas you are using the master branch righ tnow**
8. **Select execute windows bacth command under the build section**
9. **Select apply and save**
10. **Now your project is ready :**
11. 
12. 
13. **Before execution is you need to setup the git configuration**
14. **Copy the git location till bin folder:** **C:\Program Files\Git\bin**
15. **Click on settings tools and Go to manage Jenkins**
16. **Goto the tools configuration**
17. **Got to git hub section and**
18. 
19. **Apply and save**
20. **Go to the dash board and the project run thorugh build now**
21. 