Framework Explanation

It is data driven framework with page object model (POM) design pattern and we are using Pytest as a unit testing framework.

Page object model is not a framework it is a design pattern

“Design pattern is a way of organizing your code for better maintainability and scalability purpose”

We are maintaining locators and test data in excel sheet and we have written some generic function in separate library file that are applicable for all web application like edit, click , select ,mouse hover all those thing and we have a something known as conftest.py so inside conftest.py we have written a fixtures which performs setup and teardown method (in setup =opening the browser ,launching maximizing it , in teardown = after yield keyword we have closing the browser )and we have something known as config.py inside config.py we have written a class in that class we are maintaining all class variables all constants

How are you maintain the test data, locators, how are you writing the scripts, which unit test framework you are using? , how are you executing your suite , and how are you generating the reports?

we create separate worksheet for each page and each worksheet we’ll maintained the locators type and locator value of all the objects that are present in that web page

according to our module we are dividing our test for each module we are creating a separate worksheet in that worksheet we are maintaining the test data of the test cases pertaining to that module (first column was test case name and second column was execute and rest or the column was actual data) so this is how we were maintaining the test data and we are using xlrd a third party library to help us in reading from excel sheet. And all configuration related thing we are maintaining in a separate module by name config.py in side that we have a class name by config in side that class we are maintaining all projects related constant like url path of the library path of the chrome driver maximum timeout period etc…

first explain locators , second test data , third one conftest , fourth one configuration, fifth one actual test methods and sixth on POM classes(inside pom we are creating all pom classes for each page we are creating a separate module by name loginpage.py and we are writing pom methods inside that pom and all the general functionalities we are wting in the homepage )

how you are synchronization ? we have a decorator which does the synchronization so we are decorating all our generic methods that are present in seleniumwrapper.py using that @wait decorator (before executing the original function click or edit or select it will wait )

1 .data driven approach with pom design model

2. locators

3. excel\_sheet

4.Generic Library seleniumwrapper (generic function)

5 .conftest

6. config (we a module by name config.py inside that we have a class in that class we don’t have any method but rather we are maintaining some constants)

7. synchronization (@wait.py)

8. POM (we are converting each user action into a method inside corresponding class)

How are you parametrizing it?(we are going to parametrizing by using a decorator called @mark.parametrize and you have to pass the headers in a comma separated format and actual data in list of lists or list of tuple)

9. generating the HTML reports (we have Pytest html plugin using that we are going to generating the report using Pytest –html” ”)

how did you implemented oops in your framework?

Where did you used inheritance in your framework?

We have something known as homepage inside homepage we are maintain all the functionality that applicable for all the pages to the application

So homepage actually inherits from selenium wrapper and all pom classes inherits from homepage (this is multi level inheritance)

Method overriding ---

We have used method overriding , visibility\_of\_element located class it was only checking whether the element is loaded in the DOM and visible on the web page but it is not checking for whether the element is enable or not that is the reason we have created our own custom class \_visibility\_of\_element and then we are inheriting from visibility\_of\_element located class and we reusing the functionality that is given by --call—method that is present in visibility\_of\_element located class and top of it we are adding extra functionality to check the enablement