I am using **POM file model to automate the workflow using selenium. As Page object model file structure gives more clarity to the user.** There will be four components to this. Please let me know if any clarifications required

1. Feature file. (using **Specflow**)
2. Step definition file
3. Page object model file(where the functions will be declared and locators will be enclosed).
4. Test data – CSV file to read the values [dynamic values]

**Feature file (Gherkin format)**

**Background:**

**GIVEN** I Navigate to Amazon's website:  
**AND** work on the Amazon website using Selenium WebDriver.

**@UseCSVData**  
**Scenario**: TC01 **Search for a particular product in Amazon**  
**AND** Search for a specific product *laptop***AND** I Enter the search query in the search bar and submit the search query.  
**THEN** I Verify that the search results page contains relevant products:  
**WHEN** I Click on one of the products: and open in one Browser **THEN** Ensure that the product details page loads correctly with the necessary information (e.g., product title, description).

**@UseCSVData**  
**Scenario**: TC02 **Testing the pagination functionality**  
**AND** Search for a specific product *laptop***AND** I Enter the search query in the search bar and submit the search query.  
**AND** I Test pagination functionality to verify navigation between search result pages.  
**AND** I Implement filter testing to validate the functionality of filters (e.g., brand, price range).  
**AND** Open the product details page of multiple products in different browsers and verify consistent  
behavior. Add assertions to validate page elements such as headers, footers, and logos.

………………………………..

**STEP DEFINITION FILE  
*Step definition class file to the above feature file where all the functions will be defined  
using Nunit.Framework;  
using PageObjectModels.AmazonPage;  
using TechTalk.Specflow***

**//Using the CSV file to read username and password**public void GivenILogintoAmazon (String username, String Password)  
{

If (\_scenariocontext.scenarioInfo.Tags.Contains(“UseCSVData”))  
{User = \_dataService.GetTestData(User.TrimStart());  
}  
If(Password.ToLower() ==”password”)  
{\_**loginPage**  
 .**NavigateTo**(\_appsettings.Url)  
.login(User, password)  
}

**[And (@ ”Search for a specific product (.\*) ”) ]**  
public void SearchForProduct (string Product)  
{  
if(\_scenarioContext.scenarioInfo.Tags.Contains(“UseCsvData”))  
{Product = \_dataService.GetTestData(Product.TrimStart());}  
Amazonpage. SearchProduct(Product)  
}

**[And (@ ”** **I Verify that the search results page contains relevant products:”) ]**  
public void SearchForProduct ()  
{Amazonpage.VerifyResults();  
}  
  
**[When (@ ” I Click on one of the products: and open in one Browser”) ]**  
public void ClickOnProduct ()  
{Amazonpage.ClickOnProduct();  
}

**[And (@”I Test pagination functionality to verify navigation between search result pages.”) ]**  
public void VerifyPagination ()  
{Amazonpage.NavigatePagination();  
}  
  
**[And (@”I Implement filter testing to validate the functionality of filters (e.g., brand, price range).”) ]**  
public void VerifyFilters (String Brand)  
{ if(\_scenarioContext.scenarioInfo.Tags.Contains(“UseCsvData”))  
{Brand = \_dataService.GetTestData(Product.TrimStart());}  
Amazonpage. VerifyFilter(Brand)  
  
}

**Pageobject file - -AmazonPage**  
using Nunit.Framework;  
using OpenQA.Selenium;  
using OpenQA.Selenium.Interactions  
using TechTalk.Specflow

//for landing on the url page  
public **LoginPage** **NavigateTo**(Uri uri)

{ **GoToUrl(“**[**https://www.amazon.in/**](https://www.amazon.in/)**”);**return this;  
}  
//for login  
public void Login (String username, string password)

{

}

Public void SearchProduct (String Product)

{   
 Driver.WaitForElementToBeDisplayed(By.XPath(Locators.Search),60);  
 Driver.Click(By.XPath(Locators.Search);  
 //Product value coming from csv file- can be anything  
 Driver.sendkeys(By.XPath(Locators.Search),Product);

}

Public void VerifyResults ()

{

// store all the results in the list and check in the description for laptop word per pagination

//taking the count in the variable  
var ResultCount = Driver.FindElements(By.XPath(Locators. SearchList).Count;  
for( var i=1;i<=ResultCount;i++)

{  
**//storing the description in the list**List<String> SearchResultProduct = new List <String>();  
**//adding the description of the results here in the list**SearchResultProduct.Add(Driver.FindElement(By.XPath(Locators. SearchList);  
}  
  
**//to check whether the list contains the word ‘laptop’**  
String text = “Laptop”;  
if (SearchResultProduct.Contains(text)  
{  
 Console.Writeln(SearchResultProduct.count)

}

Public void ClickOnProduct()

{  
 Driver.Click(By.XPath(Locators. FirstProduct);  
**//opens the test result in new tab so switching to new window**  
Driver.SwitchTo().defaultContent();  
**//wait for product title to be visible**  
Driver.WaitForElementUntiltobeVisible(By.Xpath(Locators. ResultonLandingPage));

}

Public void NavigatePagination()

{  
Driver.WaitUntilElementClickable(By.XPath(Locators.NextPage));  
Driver.click(By.XPath(Locators.NextPage));  
//checking for the previous button  
Driver.WaitUntilElementClickable(By.XPath(Locators.PrevPage));

}

Public void VerifyFilter (String Brand)

{  
 //navigating to the filters and selecting the brand  
//retrieving brands from the csv based on selection – verifying/comparing all the brands are coming on UI  
  
var BrandResultCounter = Driver.FindElements(By.XPath(Locators.BrandList)).Count;  
for (var i=1;i<= BrandResultCounter;i++)  
{List<String> BrandListValues = Brands.Split(‘;’).ToList();  
var ExpectedBrandListCount = BrandListValues.Count;  
List<String>Actual = new List<String>();  
Actual.Add(Driver.FindElement(…);  
var ActualCount = Actual.Count;  
  
for (var k=0;k<ActualCount;k++)  
{for(var j=k;j< ExpectedBrandListCount;j++)  
{Assert.AreEqual(Actual[k], BrandListValues[j]);  
Console.writeln(Actual[k] +” is visible”);  
}}  
}

Public static class Locators

{

Public const string Search = “//div/label/following-sibling::input[@aria-label='Search Amazon.in']”;  
Public const string SearchList = “//div[@data-cy='title-recipe']”;  
public const string FirstProduct = “(//div[@data-cy='title-recipe'])[1]”;  
public const string ResultonLandingPage = “//span[@id='productTitle']”;  
public const string NextPage = “//a[@aria-label='Go to next page, page 2']”  
public const string PrevPage = “//a[@aria-label='Go to previous page, page 1']”;  
public const string BrandList = “//div[@id='brandsRefinements']/div/following-sibling::ul/span/span”;

}