Instanda QA-Take Home Challenge

Frameworks & Design:

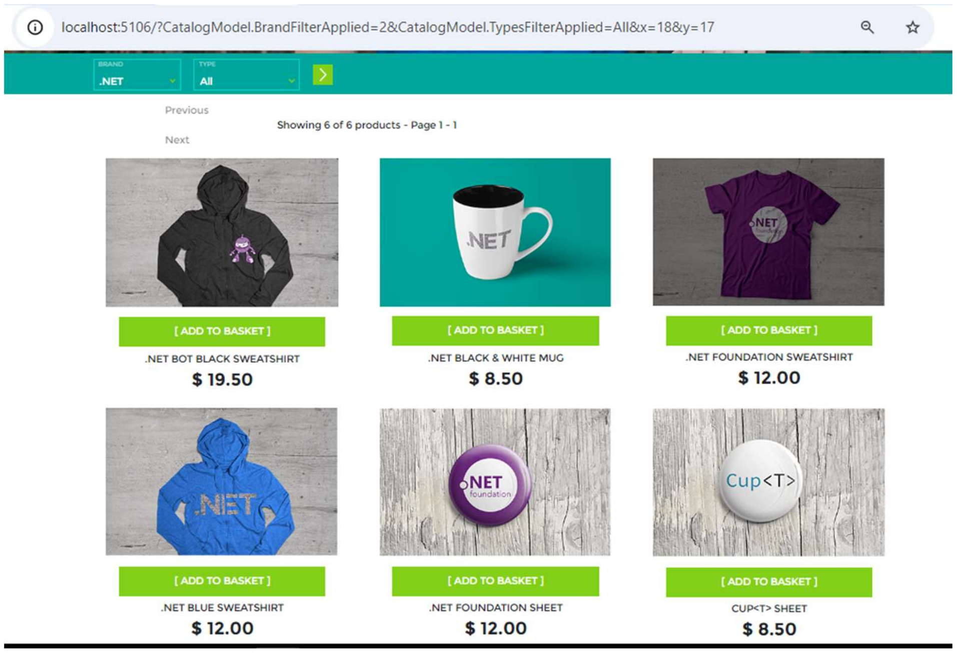
- The test solution is designed in Python using the Playwright automation framework and follows a Pytest structure. I adopted a modular approach, breaking tests into distinct functions like login, add_products_to_basket, and review_and_checkout. This ensures code reuse, easier debugging, and maintainability, with each function focusing on a specific task.
- To run the entire test suite, navigate to the project directory in the terminal and execute the command pytest, which will run all available tests. If you want to run a specific test, use pytest tests/test_file_name.py::test_method_name.

Findings:

- Once the home page is launched, we could see 12 product items listed in 4 rows and 3 columns fashion. Such that all the products should be fit into a single page rather the existing functionality shows 2 items are shifted to the next page leaving assumptions as below

    1. Alignment issue
    2. To test the options as next and previous are working without any issue, if the products count would increase in future.

- Regarding the filters, checked both Brand and Type options and found type is worked as expected for the provided products. Whereas, while trying to filter brand using .NET observed "Cup<T> Sheet" is being listed which seems to be an error under wrong grouping.


localhost:5106/?CatalogModel.BrandFilterApplied=2&CatalogModel.TypesFilterApplied=All&x=18&y=17

BRAND .NET TYPE All

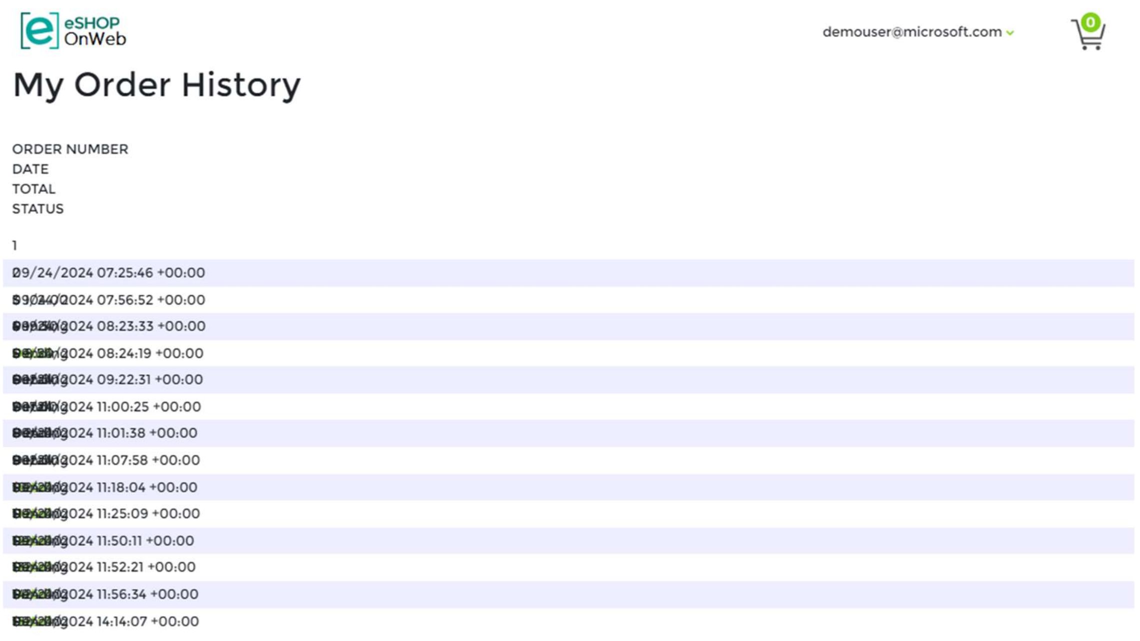
Previous
Next

Showing 6 of 6 products - Page 1 - 1

[ADD TO BASKET] .NET BOT BLACK SWEATSHIRT $ 19.50	[ADD TO BASKET] .NET BLACK & WHITE MUG $ 8.50	[ADD TO BASKET] .NET FOUNDATION SWEATSHIRT $ 12.00
[ADD TO BASKET] .NET BLUE SWEATSHIRT $ 12.00	[ADD TO BASKET] .NET FOUNDATION SHEET $ 12.00	[ADD TO BASKET] CUP<T> SHEET $ 8.50

- When tried to save phone number in profile section, observed that field accepts only numbers along with '+' special character which is a valid one, but the limit is not specified. Usually, the phone number should be accepting only 10 characters apart from country code.

- Regarding the My Orders section, if the user has a single order, then order status "Pending" and "Detail" link is visible and could select the link and view the single order details without any issue. But as the testing progress and User completes multiple order by Pay Now, then the order status - pending, timestamp & detail link is being overlapped on each order making automation test script to un-identify the present order details to view. Attached screenshot below.



eSHOP OnWeb

demouser@microsoft.com

My Order History

ORDER NUMBER
DATE
TOTAL
STATUS

1
09/24/2024 07:25:46 +00:00
[illegible]2024 07:56:52 +00:00
[illegible]2024 08:23:33 +00:00
[illegible]2024 08:24:19 +00:00
[illegible]2024 09:22:31 +00:00
[illegible]2024 11:00:25 +00:00
[illegible]2024 11:01:38 +00:00
[illegible]2024 11:07:58 +00:00
[illegible]2024 11:18:04 +00:00
[illegible]2024 11:25:09 +00:00
[illegible]2024 11:50:11 +00:00
[illegible]2024 11:52:21 +00:00
[illegible]2024 11:56:34 +00:00
[illegible]2024 14:14:07 +00:00

Below are the automated test cases written in Python using Playwright & Pytest design framework.

1. Overall functionality test coverage

   1. Launch home page
   2. Navigate to login
   3. Provide valid credentials and select login
   4. Add few products to the basket and update the quantity if needed
   5. Review the product, quantity & cost before checkout
   6. Complete the checkout by selecting pay now
   7. Success page should appear after order placement
   8. Navigate to user drop down and select My orders
   9. Verify ordered product is being displaced correctly
   10. Select logout option from user drop down.

2. Updating mobile number

   1. Login using valid credentials
   2. Navigate to my account
   3. Try changing the phone number by entering numbers and save
   4. Confirm the phone number.

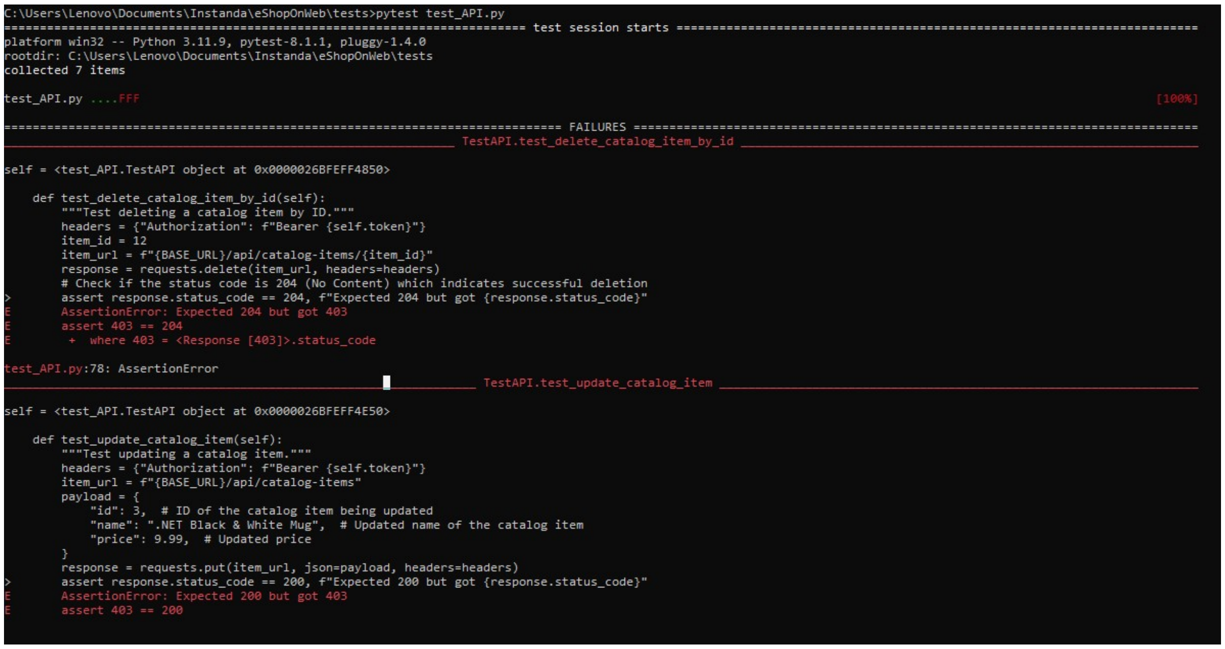
3. Checking filters in the home page

   1. Launch home page.
   2. Select Brand and Type filter as required and hit ‘>’ for search
   3. Products should be filtered based on the filters provided.

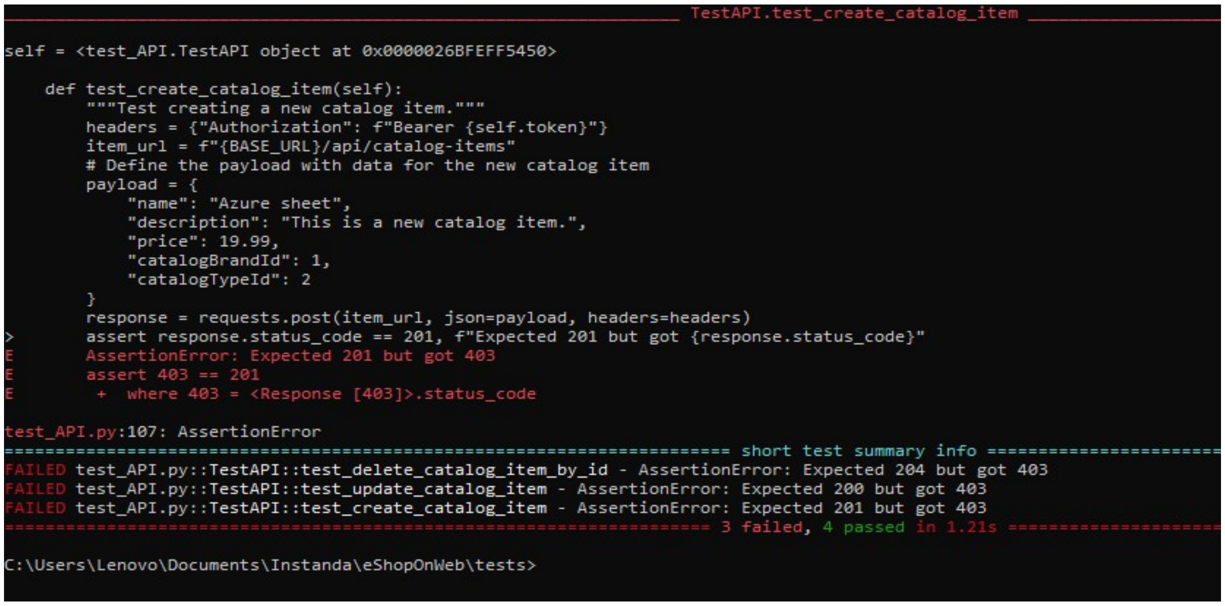
API Testing:

I conducted a series of API tests to cover significant endpoints for the eShopOnWeb application. The tests included GET operations for retrieving catalog brands, catalog types, and a specific catalog item by ID, which all passed successfully, indicating the API's ability to fetch data is working as expected. I also tested POST, PUT, and DELETE operations for creating, updating, and deleting catalog items. These tests returned a 403 Forbidden error. Overall, the authentication mechanism works correctly, and I was able to perform a variety of read operations, but access to write, update, and delete operations is restricted due to permission limitations.

Please find the 403 error code failures in the below screenshots provided.



```
C:\Users\Lenovo\Documents\Instanda\eShopOnWeb\tests>pytest test_API.py
======================================================================= test session starts =======================================================================
platform win32 -- Python 3.11.9, pytest-8.1.1, pluggy-1.4.0
rootdir: C:\Users\Lenovo\Documents\Instanda\eShopOnWeb\tests
collected 7 items

test_API.py ....FFF                                                                                                                                          [100%]

============================================================================ FAILURES =============================================================================
_______________________________________________________________ TestAPI.test_delete_catalog_item_by_id ________________________________________________________________

self = <test_API.TestAPI object at 0x0000026BFEFF4850>

    def test_delete_catalog_item_by_id(self):
        """Test deleting a catalog item by ID."""
        headers = {"Authorization": f"Bearer {self.token}"}
        item_id = 12
        item_url = f"{BASE_URL}/api/catalog-items/{item_id}"
        response = requests.delete(item_url, headers=headers)
        # Check if the status code is 204 (No Content) which indicates successful deletion
>       assert response.status_code == 204, f"Expected 204 but got {response.status_code}"
E       AssertionError: Expected 204 but got 403
E       assert 403 == 204
E        +  where 403 = <Response [403]>.status_code

test_API.py:78: AssertionError
__________________________________________________________________ TestAPI.test_update_catalog_item __________________________________________________________________

self = <test_API.TestAPI object at 0x0000026BFEFF4E50>

    def test_update_catalog_item(self):
        """Test updating a catalog item."""
        headers = {"Authorization": f"Bearer {self.token}"}
        item_url = f"{BASE_URL}/api/catalog-items"
        payload = {
            "id": 3,  # ID of the catalog item being updated
            "name": ".NET Black & White Mug",  # Updated name of the catalog item
            "price": 9.99,  # Updated price
        }
        response = requests.put(item_url, json=payload, headers=headers)
>       assert response.status_code == 200, f"Expected 200 but got {response.status_code}"
E       AssertionError: Expected 200 but got 403
E       assert 403 == 200
```



```
_____________________________________________________________ TestAPI.test_create_catalog_item ________________

self = <test_API.TestAPI object at 0x0000026BFEFF5450>

    def test_create_catalog_item(self):
        """Test creating a new catalog item."""
        headers = {"Authorization": f"Bearer {self.token}"}
        item_url = f"{BASE_URL}/api/catalog-items"
        # Define the payload with data for the new catalog item
        payload = {
            "name": "Azure sheet",
            "description": "This is a new catalog item.",
            "price": 19.99,
            "catalogBrandId": 1,
            "catalogTypeId": 2
        }
        response = requests.post(item_url, json=payload, headers=headers)
>       assert response.status_code == 201, f"Expected 201 but got {response.status_code}"
E       AssertionError: Expected 201 but got 403
E       assert 403 == 201
E        +  where 403 = <Response [403]>.status_code

test_API.py:107: AssertionError
============================================================ short test summary info ==================
FAILED test_API.py::TestAPI::test_delete_catalog_item_by_id - AssertionError: Expected 204 but got 403
FAILED test_API.py::TestAPI::test_update_catalog_item - AssertionError: Expected 200 but got 403
FAILED test_API.py::TestAPI::test_create_catalog_item - AssertionError: Expected 201 but got 403
============================================================= 3 failed, 4 passed in 1.21s =============

C:\Users\Lenovo\Documents\Instanda\eShopOnWeb\tests>
```