EX.NO.: 08
DATE: 28.07.2025

SELENIUM AUTOMATION USING POM FOR E-COMMERCE WEBSITE

AIM

To automate the end-to-end testing of a basic ecommerce website workflow using Selenium and Pytest. The test will verify that a user can successfully search for a product, filter results, select a product, add it to the cart, proceed to checkout, and place an order, ensuring all critical functionalities work as expected.

ALGORITHM

1. Setup WebDriver: Launch the browser and open the ecommerce home page.
2. Search Product: Enter the product name in the search box and submit the search.
3. Filter Search Results: Apply price and category filters to narrow down product listings.
4. Select Product: Choose the first product from the filtered search results.
5. Add to Cart: On the product page, add the selected product to the shopping cart.
6. Go to Cart: Navigate to the cart page.
7. Verify Cart Total: Retrieve and verify the total price matches the expected product price.
8. Proceed to Checkout: Click the checkout button to go to the checkout page.
9. Place Order: Confirm the order placement and verify success (e.g., alert or confirmation message).
10. Tear down: Close the browser.

CODE AND OUTPUT

```
import pytest
import time
from pages.home_page import HomePage
from pages.search_results_page import SearchResultsPage
from pages.product_page import ProductPage
from pages.cart_page import CartPage
from pages.checkout_page import CheckoutPage

def test_end_to_end_shopping_flow(setup):
    driver = setup

    print("Starting: Search for Laptop")
    home = HomePage(driver)
    home.search_product("Laptop")
    time.sleep(2)

    print("Applying filters")
    search = SearchResultsPage(driver)
    search.apply_price_filter()
    time.sleep(1)
    search.apply_category_filter()
    time.sleep(1)
    search.select_first_product()
    time.sleep(2)

    print("Adding product to cart")
    product = ProductPage(driver)
    product.add_to_cart()
```

```
    time.sleep(1)
    product.go_to_cart()
    time.sleep(2)

    print("Validating cart total")
    cart = CartPage(driver)
    total = cart.get_cart_total()
    assert total == "$999.99"
    time.sleep(1)
    cart.proceed_to_checkout()
    time.sleep(2)

    print("Placing order")
    checkout = CheckoutPage(driver)
    checkout.place_order()
    time.sleep(2)

    print("Test completed successfully")
```

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

class BasePage:
    def __init__(self, driver):
        self.driver = driver

    def enter_text(self, by_locator, text):
        WebDriverWait(self.driver, 10).until(
            EC.visibility_of_element_located(by_locator)
        ).clear()
        WebDriverWait(self.driver, 10).until(
            EC.visibility_of_element_located(by_locator)
        ).send_keys(text)

    def click(self, by_locator):
        WebDriverWait(self.driver, 10).until(
            EC.element_to_be_clickable(by_locator)
        ).click()

    def get_text(self, by_locator):
        element = WebDriverWait(self.driver, 10).until(
            EC.visibility_of_element_located(by_locator)
        )
        return element.text
```

```
(.venv) PS D:\TARU\V th year\Software Testing lab\Ex 8> pytest -v -s
============================= test session starts ==============================
platform win32 -- Python 3.11.9, pytest-8.4.1, pluggy-1.6.0 -- d:\TARU\V th year\Software Testing lab\.venv\Scripts\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.11.9', 'Platform': 'Windows-10-10.0.26100-SP0', 'Packages': {'pytest': '8.4.1', 'pluggy': '1.6.0'}, 'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'},
 'JAVA_HOME': 'C:\\Users\\Hema\\Downloads\\OpenJDK-24'}
rootdir: D:\TARU\V th year\Software Testing lab\Ex 8
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 1 item

test_ecommerce_flow.py::test_end_to_end_shopping_flow
DevTools listening on ws://127.0.0.1:60218/devtools/browser/c8d3f635-af9e-4d88-8241-5c139d47d1fa
Starting: Search for Laptop
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1753693949.992546   19796 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
[5880:4192:0728/144230.392:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response error message: DEPRECATED_ENDPOINT
Applying filters
Adding product to cart
Validating cart total
Created TensorFlow Lite XNNPACK delegate for CPU.
Attempting to use a delegate that only supports static-sized tensors with a graph that has dynamic-sized tensors (tensor#-1 is a dynamic-sized tensor).
Placing order
Test completed successfully
PASSED

---------- Generated html report: file:///D:/TARU/V%20th%20year/Software%20Testing%20lab/Ex%208/report.html ----------
============================== 1 passed in 19.48s ==============================
```

INFERENCE

The automated test script successfully validates the key user journey in the ecommerce website from searching a product to placing an order. It ensures that critical UI elements respond correctly to user actions, filters work properly, product selections update the cart accurately, and the checkout process completes. Automating this flow helps catch regression issues early and improves testing efficiency by reducing manual effort.