Other questions to answer:

1. Why did you choose the testing framework you did?

2. What are other tests you would recommend creating for this example?

3. Could you find a bug on the test site?

1. Why did you choose the testing framework you did?

**I chose Playwright because:** it is easy to implement, supports cross-browser testing,

and includes a built-in reporting plugin with no need for additional dependencies.

It provides descriptive error messages and offers an intuitive UI for checking behavior when the console output is insufficient.

**Structure POM:** The locators are organized in the elements folder instead of being divided into individual page files. This approach allows the locators to be reused across multiple page files and helps maintain clean and manageable page files, especially when dealing with a large number of locators."

**CI/CD pipelines:**

Playwright significantly simplifies the integration of automated testing into CI/CD pipelines:

|  |  |
| --- | --- |
| pipline1 | pipline2 |

2. What are other tests you would recommend creating for this example?

- Utilities module: A module designed to store methods that are reused across various page tests.

- External Test Data: A file with all the hard-coded test data like credentials, urls or products names.

- Define Constants or Variables instead of using hard-coded numbers.

- Testing Unit Test: A test group to test out methods to ensure that there are no false positives or false negatives.

1. Authentication & Login Flows

- Test with multiple valid sets of credentials to ensure consistency. For example, test with standard and problem user accounts.

- Check login for a user that is locked out.

- Test logging in with an incorrect username and verify that the appropriate error message is displayed.

- Test logging in with an incorrect password and verify that the system gives a correct error message.

2. Product Page & Cart Interactions

- Ensure that all products on the product listing page have correct images and names.

- Verify that the cart displays the correct total price and number of items after adding/removing products.

3. Cart Page Functionality

- Verify that the quantity of products can be updated in the cart and that the total price is recalculated accordingly.

- Check that the cart shows the correct product name, price, and quantity for each item.

- Ensure that when the cart is empty, the correct message ("Your cart is empty") is displayed and the user can continue shopping.

- Check that the Continue Shopping button redirects users back to the product listing page.

4. Checkout Process

- Ensure that all required fields (e.g., name, address, postal code) are properly validated on the checkout page (limits and special characters).

- Confirm that the order summary displays the correct items, (titles and descriptions)

5. Edge Cases

- Verify that if a user adds products to the cart and then logs out, the cart is empty when they log back in with other user.

- Test what happens if a user is logged out due to inactivity and attempts to check out.

6. UI & Responsiveness

- Ensure that the site, including the product page and cart, is responsive and functions correctly on smaller screen sizes.

7. Logout Flow

- Verify that the user can log out from any page of the site and is redirected to the login page.

8. Additional Functionality Testing

- Test that clicking a product name or image navigates to the correct product detail page.

- Ensure that all the sorting options (by name, price, etc.) function as expected on the products page.

- Ensure that the product details page shows accurate information, such as price, description, and available stock.

- Test that clicking each link in the sidebar leads to the expected page and performs the correct behavior

- API Testing

9. Error Handling

- Ensure that if required fields are left blank, an appropriate error message is shown during the checkout process.

3. Could you find a bug on the test site?

Bugs:

1 - Cart is not clenned when user log out

2- Visual\_user has visual error on purpose

3- A product with the name: Test.allTheThings() T-Shirt (Red)

4- Log in error message with the text : Epic sadface: You can only access '/inventory.html' when you are logged in.

5- Datata test locatars behavior are not homogeneous.