Shopping cart test report

Test file github link: https://github.com/AliEAKarimi/react-shopping-cart/blob/main/tests/test.py (https://github.com/AliEAKarimi/react-shopping-cart/blob/main/tests/test.py)

Introduction

This report provides a detailed explanation of the code that automates the testing of the Shopping cart project using Selenium, a web testing framework. The code performs various tests to verify the functionality of our project

Code explanation

ShoppingCartTest class

The code creates an instance of the ShoppingCartTest class and runs all test-methods to perform the tests.

```
class ShoppingCartTest(unittest.TestCase):
```

setUp method

This method will be called before each test. It will create a new Chrome session, maximize the window size, delete all cookies and navigate to the application home page.

```
def setUp(self):
    # create a new Chrome session
    self.driver = webdriver.Chrome()
    # maximize the window size
    self.driver.maximize_window()
    # delete all cookies
    self.driver.delete_all_cookies()
    # navigate to the application home page
    self.driver.get("http://localhost:3000")
```

tearDown method

This method will be called after each test. It will close the browser window.

```
def tearDown(self):
    # close the browser window
    self.driver.quit()
```

Test methods

We have 8 test methods in this project. Each test method performs a specific test.

1. test_title(self) method

This test verifies that the title of the page is correct or not. The test is successful if the title of the page is "Typescript React Shopping cart". The test is unsuccessful if the title of the page is not "Typescript React Shopping cart".

```
def test_title(self):
    time.sleep(2)
    self.title = self.driver.title
    self.assertEqual(self.title, "Typescript React Shopping cart")
```

2. test_display_16_products_correctly(self) method

This test verifies that the products are displayed correctly. It verifies that there are 16 products on the page and that each product card has an image, title, price and add to cart button. The test is successful if all the products are displayed correctly. The test is unsuccessful if any of the products are not displayed correctly.

```
def test_display_16_products_correctly(self):
    time.sleep(2)
    # Find all product cards on the page
    self.product_cards = self.driver.find_elements(By.XPATH, "//*[contains(@class, 'product-card')]")
    # Verify that there are 16 products on the page
    assert len(self.product_cards) == 16
    # Verify that each product card has an image, title, price and add to cart button
    for product_card in self.product_cards:
        assert product_card.find_element(By.XPATH, "//*[contains(@id, 'product-image')]")
        assert product_card.find_element(By.XPATH, "//*[contains(@id, 'product-title')]")
        assert product_card.find_element(By.XPATH, "//*[contains(@id, 'product-price')]")
        assert product_card.find_element(By.XPATH, "//*[contains(@id, 'add-to-cart-button')]")
```

3. test_empty_cart(self) method

This test verifies that the cart is empty when the page is loaded by checking that the cart icon shows 0 items. The test is successful if the cart icon shows 0 items. The test is unsuccessful if the cart icon shows any other number than 0.

```
def test_empty_cart(self):
    time.sleep(2)
    # Find the cart icon
    cart_quantity = self.driver.find_element(By.XPATH, "//*[contains(@id, 'cart-quantity')]")
    self.assertEqual(cart_quantity.text, '0')
```

4. test_add_product_to_cart(self) method

This test verifies that the product is added to the cart by clicking the Add to Cart button and checking that the cart icon shows 1 item. The test is successful if the cart icon shows 1 item. The test is unsuccessful if the cart icon shows any other number than 1.

```
def test_add_product_to_cart(self):
    # Find the first product card on the page
    product_card = self.driver.find_element(
      By.XPATH, "//*[contains(@class, 'product-card')]"
    )
    # Click the Add to Cart button
    product_card.find_element(
      By.XPATH, "//*[contains(@id, 'add-to-cart-button')]"
    ).click()
    time.sleep(2)
    # Close the cart modal
    cartButton = self.driver.find_element(By.XPATH, "//*[@id=\"open-cart-button\"]")
    cartButton.click()
    time.sleep(2)
    # Verify that the cart icon shows 1 item
    cart_quantity = self.driver.find_element(By.XPATH, "//*[contains(@id, 'cart-quantity')]")
    assert cart_quantity.text == '1'
```

5. test_filter_products_by_size(self) method

This test verifies that the products are filtered by size by clicking the checkbox of the size XXL and checking that only 4 products are displayed. The test is successful if only 4 products are displayed. The test is unsuccessful if any other number of products is displayed.

```
def test_filter_products_by_size(self):
    checkbox = self.driver.find_element(By.XPATH, "//*[contains(@id, 'XXL')]")
    checkbox.click()
    time.sleep(2)
    product_cards = self.driver.find_elements(By.XPATH, "//*[contains(@class, 'product-card')]")
    assert len(product_cards) == 4
```

6. test_cart_list(self) method

This test verifies that the cart list displays all products added to the cart by clicking the Add to Cart button and checking that the cart list displays the product added to the cart with correct information. The test is successful if the cart list displays the product added to the cart. The test is unsuccessful if the cart list does not display the product added to the cart.

```
def test_cart_list(self):
    # Find the first product card on the page
    product_card = self.driver.find_element(
      By.XPATH, "//*[contains(@class, 'product-card')]"
    )
    productName = product_card.find_element(By.XPATH, "//*[contains(@id, 'product-title')]").text
    productPrice = product_card.find_element(By.XPATH, "//*[contains(@id, 'product-price')]").text
    # Click the Add to Cart button
    product_card.find_element(
      By.XPATH, "//*[contains(@id, 'add-to-cart-button')]"
    ).click()
    time.sleep(2)
    # Verify that the cart list displays the product added to the cart
    cartProducts = self.driver.find_elements(By.XPATH, "//*[contains(@class, 'cart-product')]")
    assert len(cartProducts) == 1
    # Verify that the product name is displayed correctly
    cartProductName = cartProducts[0].find_element(By.XPATH, "//*[contains(@id, 'cart-product-title')]").text
    assert productName == cartProductName
    # Verify that the product price is displayed correctly
    cartProductPrice = cartProducts[0].find_element(By.XPATH, "//*[contains(@id, 'cart-product-price')]").text
    assert cartProductPrice.replace(" ", "") == productPrice.split("\n")[0]
    # Verify Remove button works
    cartProducts[0].find_element(By.XPATH, "//*[contains(@id, 'remove-product')]").click()
    time.sleep(2)
```

7. test_checkout_button_alert_message(self) method

This test verifies that the checkout button displays an alert message when the cart is empty by clicking the checkout button and checking that an alert message is displayed and be correct. The test is successful if an alert message is displayed with correct message. The test is unsuccessful if an alert message is not displayed or displayed with incorrect message.

```
def test_checkout_button_alert_message(self):
    # Click the cart button
    cartButton = self.driver.find_element(By.XPATH, "//*[contains(@id, 'open-cart-button')]")
    cartButton.click()
    time.sleep(2)
    # Click the checkout button
    checkoutButton = self.driver.find_element(By.XPATH, "//*[contains(@id, 'checkout-button')]")
    checkoutButton.click()
    time.sleep(2)
    # Verify that an alert message is displayed
    alert = self.driver.switch_to.alert
    assert alert.text == 'Add some product in the cart!'
    time.sleep(2)
```

8. test_checkout_button_redirect(self) method

This test verifies that the link to the github star page is working by clicking the Star link and checking that the github page is displayed. The test is successful if the github page is displayed. The test is unsuccessful if the github page is not displayed.

```
def test_github_link(self):
    # Click the github link
    githubLink = self.driver.find_element(By.XPATH, "//*[contains(@id, 'github-star-link')]")
    githubLink.click()
    time.sleep(2)
    # Verify that the github page is displayed
```