social_media_sharing_test.py

```python
#
#  Course: COSC 4P02
#  Assignment: Group Project
#  Group: 9
#  Version: 1.0
#  Date: April 2024
#
from selenium import webdriver # Import the webdriver module.
from selenium.webdriver.chrome.options import Options as ChromeOptions # Import the ChromeOptions class.
from selenium.webdriver.chrome.service import Service as ChromeService # Import the ChromeService class.
from selenium.webdriver.common.by import By  # Import the By class for locating elements.
from webdriver_manager.chrome import ChromeDriverManager # Import the ChromeDriverManager for managing ChromeDriver binaries.
from urllib.parse import urlparse, parse_qs, unquote # Import functions for parsing URLs and query strings.
import pytest # Import the pytest module for testing.

URL = "https://group9portal-eehbdxbhcgftezez.canadaeast-01.azurewebsites.net/index.php" # Our website URL to be tested. NOTE: This URL will change before the end of the project, as we will be changing Azure Subscriptions. The tests remain the same, and can be run on the new URL.

@pytest.fixture(scope="module")
def browser():
    #
    # Fixture Browser:
    # This fixture provides a browser instance using Selenium WebDriver. It uses the Chrome browser in headless mode for testing (there is a line that can be commented out to view the GUI). This fixture is automatically invoked by
    # pytest when a test function includes it as an argument. It allows test functions to interact with the browser and perform actions like navigating to URLs, finding elements, and executing JavaScript.
    # The client object is available for interacting with the app's elements, and it will be cleaned up after the test.
    #
    options = ChromeOptions() # Create an instance of ChromeOptions to configure the Chrome browser.
    options.add_argument("--headless")  # Run Chrome in headless mode (without a GUI). If this line is commented out, the browser will open in a GUI mode. The test moves very fast in headless mode, but it does show the site.
    service = ChromeService(executable_path=ChromeDriverManager().install()) # Create an instance of ChromeService to manage the ChromeDriver executable.
    driver = webdriver.Chrome(service=service, options=options) # Create an instance of the Chrome WebDriver with the specified service and options.
    yield driver # Yield the driver instance to the test function.
    driver.quit() # Quit the driver after the test function completes.

def test_share_buttons_present(browser):
```

```
34      #
35      # Test Case 1: Testing the presence of the social media share buttons. This will confirm
    that the buttons are present on the page.
36      # Execution: python -m pytest social_media_sharing_test.py -k "test_share_buttons_p-
    resent" -s -v # Only use -s to view the messages in the test.
37      # This method will check if the share buttons for Facebook, Twitter, and Email are
    present on the page. It uses the Selenium WebDriver (predefined as a fixture above) to
    navigate to our URL
38      # and check for the presence of the share buttons by their class names.
39      # Expected Result: Pass. The share buttons should be present on the page.
40      #
41      browser.get(URL) # Navigate to the specified URL in our browser instance.
42      assert browser.find_element(By.CLASS_NAME, "a2a_button_facebook").is_displayed() # Check
    if the Facebook share button is displayed.
43      assert browser.find_element(By.CLASS_NAME, "a2a_button_x").is_displayed() # Check if the
    X share button is displayed.
44      assert browser.find_element(By.CLASS_NAME, "a2a_button_email").is_displayed() # Check if
    the Email share button is displayed.
45
46  def test_share_buttons_clickable(browser):
47      #
48      # Test Case 2: Testing the clickability of the social media share buttons. This will
    confirm that the buttons are clickable while on the page.
49      # Execution: python -m pytest social_media_sharing_test.py -k "test_share_buttons_c-
    lickable" -s -v # Only use -s to view the messages in the test.
50      # This method will check if the share buttons for Facebook, Twitter, and Email are
    clickable on the page. It uses the Selenium WebDriver (predefined as a fixture above) to
    navigate to our URL
51      # and check for the enabled share buttons by their class names.
52      # Expected Result: Pass. The share buttons should be clickable on the page.
53      #
54      browser.get(URL) # Navigate to the specified URL in our browser instance.
55      assert browser.find_element(By.CLASS_NAME, "a2a_button_facebook").is_enabled() # Check if
    the Facebook share button is enabled/clickable.
56      assert browser.find_element(By.CLASS_NAME, "a2a_button_x").is_enabled() # Check if the X
    share button is enabled/clickable.
57      assert browser.find_element(By.CLASS_NAME, "a2a_button_email").is_enabled() # Check if
    the Email share button is enabled/clickable.
58
59  def test_facebook_button_url(browser):
60      #
61      # Test Case 3: Testing that the Facebook share button redirects to the correct Facebook
    Post URL. This will confirm that the opened URL is correct when the Facebook share button is
    clicked.
62      # Execution: python -m pytest social_media_sharing_test.py -k "test_facebook_button_url"
    -s -v # Only use -s to view the messages in the test.
63      # This method will check if the Facebook share button redirects to the correct Facebook
    Post URL when clicked. It uses the Selenium WebDriver (predefined as a fixture above) to
    navigate to our URL
64      # and click the Facebook share button. It then checks if the current (redirect) URL
    contains "facebook.com" to confirm that the redirect was successful.
```

```
    # Expected Result: Pass. The Facebook share button should redirect to the correct Facebook Post URL.
    #
    browser.get(URL) # Navigate to the specified URL in our browser instance.
    fb_button = browser.find_element(By.CLASS_NAME, "a2a_button_facebook") # Find the Facebook share button by its class name.
    fb_button.click() # Click the Facebook share button.
    browser.switch_to.window(browser.window_handles[-1]) # Switch to the new window that opens after clicking the button.
    assert "facebook.com" in browser.current_url # Check if the current (redirect) URL contains "facebook.com" to confirm the redirect.

def test_x_button_url(browser):
    #
    # Test Case 4: Testing that the X share button redirects to the correct X Post URL. This will confirm that the opened URL is correct when the X share button is clicked.
    # Execution: python -m pytest social_media_sharing_test.py -k "test_x_button_url" -s -v # Only use -s to view the messages in the test.
    # This method will check if the X share button redirects to the correct X Post URL when clicked. It uses the Selenium WebDriver (predefined as a fixture above) to navigate to our URL
    # and click the X share button. It then checks if the current (redirect) URL contains "x.com" to confirm that the redirect was successful.
    # Expected Result: Pass. The X share button should redirect to the correct X Post URL.
    #
    browser.get(URL) # Navigate to the specified URL in our browser instance.
    x_button = browser.find_element(By.CLASS_NAME, "a2a_button_x") # Find the X share button by its class name.
    x_button.click() # Click the X share button.
    browser.switch_to.window(browser.window_handles[-1]) # Switch to the new window that opens after clicking the button.
    assert "x.com" in browser.current_url # Check if the current (redirect) URL contains "x.com" to confirm the redirect.

def test_email_button_url(browser):
    #
    # Test Case 5: Testing that the email share button redirects to the correct blank URL, and then closes this URL. This will confirm that the redirect window opens and closes correctly when the email share button is clicked.
    # Execution: python -m pytest social_media_sharing_test.py -k "test_email_button_url" -s -v # Only use -s to view the messages in the test.
    # This method will check if the email share button redirects to the about:blank URL when clicked. It uses the Selenium WebDriver (predefined as a fixture above) to navigate to our URL
    # and click the email share button. It then checks if the current (redirect) URL contains "about:blank" to confirm that a new blank window opens and closes correctly.
    # It also checks if the number of windows is the same as before clicking the email button, which would confirm that the redirect was successful, since Selenium does not check for non-browser windows. This is a workaround to test this functionality.
    # Expected Result: Pass. The email share button should redirect to the correct about:blank URL, and then close this URL.
    #
```

```
96      browser.get(URL) # Navigate to the specified URL in our browser instance.
97      email_button = browser.find_element(By.CLASS_NAME, "a2a_button_email") # Find the email
share button by its class name.
98      initial_window_count = len(browser.window_handles) # Store the initial number of windows
before clicking the email button.
99      email_button.click() # Click the email share button.
100     browser.switch_to.window(browser.window_handles[-1]) # Switch to the new window that
opens after clicking the button.
101     assert "about:blank" in browser.current_url # Check if the current (redirect) URL
contains "about:blank" to confirm the redirect.
102     browser.close()  # Close the redirected window
103     browser.switch_to.window(browser.window_handles[0]) # Switch back to the original window.
104     assert len(browser.window_handles) == initial_window_count # Check if the number of
windows is the same as before clicking the email button. This would confirm that the redirect
was successful, since Selenium does not check for non-browser windows.
105
106 def test_x_template(browser):
107     #
108     # Test Case 6: Testing the pre-populated X template when the share button is selected.
This will confirm that the X template is correct.
109     # Execution: python -m pytest social_media_sharing_test.py -k "test_x_template" -s -v #
Only use -s to view the messages in the test.
110     # This method will check if the X template is pre-populated with the correct text when
the X share button is clicked. It uses the Selenium WebDriver (predefined as a fixture above)
to navigate to our URL
111     # and check for the pre-populated X template by executing JavaScript to retrieve the text
of the X template.
112     # Expected Result: Pass. The X template should be pre-populated with the correct text.
113     #
114     browser.get(URL) # Navigate to the specified URL in our browser instance.
115     x_button = browser.find_element(By.CLASS_NAME, "a2a_button_x") # Find the X share button
by its class name.
116     x_button.click() # Click the X share button.
117     browser.switch_to.window(browser.window_handles[-1]) # Switch to the new window that
opens after clicking the button.
118     assert "x.com" in browser.current_url # Check if the current (redirect) URL contains
"x.com" to confirm the redirect.
119
120     parsed_url = urlparse(browser.current_url) # Parse the current URL to extract its
components.
121     query = parse_qs(parsed_url.query) # Parse the query string of the URL to extract the
parameters.
122     shared_text = unquote(query.get("text", [""])[0]) # Get the value of the "text" parameter
from the query string and decode it.
123     assert len(shared_text) >= 30 # Check if the length of the shared text is greater than or
equal to 30 characters. Could change this value
124     browser.close()  # Close the redirected window
```