Automation Assessment Report

Overview

This report presents the automation assessment completed using Python and Selenium for API and web interface testing. The tasks were executed as per the provided requirements, and all functionalities were implemented successfully.

API Automation Tasks

Tools Used

- **Language**: Python
- **Library**: Requests

Task 6: API Test Cases

1. Retrieve Access Token

- **Function**: `retrieve_access_token(email, password)`
- **Inputs**: email, password
- **Outputs**: token, time_taken
- **Description**: This function sends a POST request to the Login API to retrieve an access token using provided credentials.

2. Create a User

- **Function**: `create_user(name, job)`
- **Inputs**: name, job
- **Outputs**: name, job, id, created_at, time_taken
- **Description**: This function sends a POST request to the Create User API to create a new user and retrieves details of the created user.

3. Update a User

- **Function**: `update_user(user_id, name, job)`
- **Inputs**: name, job
- **Outputs**: name, job, id, updated_at, time_taken
- **Description**: This function sends a PUT request to the Update User API to update an existing user's details and retrieves the updated information.

4. Delete a User

- **Function**: `delete_user(user_id)`
- **Inputs**: NA
- **Outputs**: time_taken
- **Description**: This function sends a DELETE request to the Delete User API to remove a user and measures the time taken for the operation.

Web Interface Automation Tasks

Tools Used

- **Language**: Python
- **Library**: Selenium

Task 7: Web Interface Test Cases

1. Log In to the Site

- **Function**: `login(driver, username, password)`
- **Description**: This test automates the login process for the Swag Labs interface, verifying successful login by checking the page title.

2. Add and Remove Item from Cart

- **Function**: `add_item_to_cart_and_verify(driver, item_name)`
- **Description**: This test adds an item to the cart, verifies its presence, removes it, and then verifies that it has been removed from the cart.

Detailed Explanation

API Tasks

1. **Retrieve Access Token**
    - **Implementation**: The function sends a POST request with user credentials and extracts the access token from the response.
    - **Code**:

    ```
    def retrieve_access_token(email, password):
        url = "https://reqres.in/api/login"
        payload = {"email": email, "password": password}
        response = requests.post(url, json=payload)
        token = response.json().get("token")
        time_taken = response.elapsed.total_seconds()
        return token, time_taken
    ```

2. **Create a User**
   - **Implementation**: The function sends a POST request to create a new user and returns the user's details and creation time.
   - **Code**:

```
def create_user(name, job):
    url = "https://reqres.in/api/users"
    payload = {"name": name, "job": job}
    response = requests.post(url, json=payload)
    user_data = response.json()
    time_taken = response.elapsed.total_seconds()
    return user_data["name"], user_data["job"], user_data["id"],
user_data["createdAt"], time_taken
```

3. **Update a User**
   - **Implementation**: The function sends a PUT request to update user details and retrieves the updated information.
   - **Code**:

```
def update_user(user_id, name, job):
    url = f"https://reqres.in/api/users/{user_id}"
    payload = {"name": name, "job": job}
    response = requests.put(url, json=payload)
    user_data = response.json()
    time_taken = response.elapsed.total_seconds()
    return user_data["name"], user_data["job"], user_data["id"],
user_data["updatedAt"], time_taken
```

4. **Delete a User**
   - **Implementation**: The function sends a DELETE request to remove a user and measures the time taken for the operation.
   - **Code**:

```
def delete_user(user_id):
    url = f"https://reqres.in/api/users/{user_id}"
    response = requests.delete(url)
    time_taken = response.elapsed.total_seconds()
    return time_taken
```

Web Interface Tasks

1. **Log In to the Site**
   - **Implementation**: Uses Selenium to automate login, checks the page title to confirm successful login.
   - **Code**:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
```

```
import time

def setup_browser():
    options = Options()
    # options.add_argument("--headless")
    # options.add_argument("--disable-gpu")
    service = Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=service, options=options)
    return driver

def login(driver, username, password):
    driver.get("https://www.saucedemo.com/")
    driver.find_element(By.ID, "user-name").send_keys(username)
    driver.find_element(By.ID, "password").send_keys(password)
    driver.find_element(By.ID, "login-button").click()
    assert "Swag Labs" in driver.title, "Login failed"

def main():
    driver = setup_browser()
    try:
        login(driver, "standard_user", "secret_sauce")
        print("Login test passed.")
    except AssertionError as e:
        print(e)
    finally:
        driver.quit()

if __name__ == "__main__":
    main()
```

2. **Add and Remove Item from Cart**
    - **Implementation**: Adds an item to the cart, verifies its presence, then removes it and checks if it's no longer in the cart.
    - **Code**:

```
def add_item_to_cart_and_verify(driver, item_name):
    driver.get("https://www.saucedemo.com/")
    driver.find_element(By.ID, "user-
name").send_keys("standard_user")
    driver.find_element(By.ID,
"password").send_keys("secret_sauce")
    driver.find_element(By.ID, "login-button").click()
    driver.find_element(By.ID, f"add-to-cart-
{item_name.lower().replace(' ', '-')}").click()
    driver.find_element(By.CLASS_NAME,
"shopping_cart_link").click()
    time.sleep(2)
    cart_items = driver.find_elements(By.CLASS_NAME, "cart_item")
    item_names = [element.find_element(By.CLASS_NAME,
"inventory_item_name").text for element in cart_items]
    assert item_name in item_names, f"Item '{item_name}' was not
found in the cart."
    driver.find_element(By.ID, f"remove-
{item_name.lower().replace(' ', '-')}-button").click()
    time.sleep(2)
    cart_items = driver.find_elements(By.CLASS_NAME, "cart_item")
    item_names = [element.find_element(By.CLASS_NAME,
"inventory_item_name").text for element in cart_items]
    assert item_name not in item_names, f"Item '{item_name}' was
not removed from the cart."
```

Recording

Due to the constraints of time, a detailed video recording explaining the approach and demonstrating the solution is not available. However, the provided code and detailed explanations should offer a comprehensive understanding of the implemented test cases and their functionalities.

Conclusion

All tasks have been completed successfully. The code has been modularized for reusability and clarity. For any further questions or clarifications, feel free to reach out.

GitHub Repository

Link to GitHub Repository

Please ensure you have access to the repository using the provided email addresses.