Test Execution Report for OpenMRS Login Automation

Project: Quality Assurance Associate Practical Interview

Task: Automate the Login Page of OpenMRS

Framework Used: Selenium with Python

Test Environment:

Platform: https://qa.kenyahmis.org/openmrs/spa/login

Credentials:

Username: admin

Password: Admin123

Important Links:

- https://github.com/Data-DrivenQA/OpenMRS-Automation.git
- https://github.com/Data-DrivenQA/OpenMRS-Automation/actions/runs/10340582487/job/28621446044

1. Setup Automation Framework

Objective:

Set up the necessary environment and framework to automate the login process for the OpenMRS platform.

Steps:

Choosing the Framework:

I chose Selenium with Python as my automation framework – this is because Selenium offers extensive documentation, community support, and ability to automate web browsers.

Environment Setup:

Installed Selenium: Used pip to install the Selenium package in Python.

WebDriver Installation: Downloaded and set up the Chrome WebDriver for browser automation.

Created Virtual Environment: Set up a Python virtual environment to manage dependencies.

Initial Script Development:

Navigating to the Login Page: Automated the browser to open the OpenMRS login page using Selenium.

Element Identification: Identified the necessary elements (username, password fields, and buttons) using CSS selectors.

Test Execution:

Login Automation: Implemented the script to automate the input of credentials and submission of the login form.

Outcome:

The environment was successfully set up, and the initial automation script was able to navigate to the OpenMRS login page and perform basic actions like entering the username and password.

2. Automate the Login Process

Objective:

Automate the login process on the OpenMRS platform by interacting with the necessary UI elements, including the username and password fields and the login button.

Steps:

Navigating to the Login Page:

Developed the script to automatically open the OpenMRS login page using the Selenium WebDriver.

Locating UI Elements:

Username Field: Identified the username input field using its unique CSS selector.

Password Field: Identified the password input field using its unique CSS selector.

Login Button: Located the "Continue" button (used after entering both username and password) by its CSS selector.

Credential Input and Submission:

Designed the Scripts to enter valid credentials (username: admin, password: Admin123) into the respective fields.

The script clicked the login button to submit the credentials.

Error Handling:

Invalid Username/Password Detection: Implemented error handling to detect the "Invalid username or password" message if incorrect credentials were provided.

Challenges Faced: Initially, the script failed to correctly trace the error message when incorrect credentials were used. After debugging and refining the CSS selectors, the issue was resolved, allowing the script to accurately detect login failures.

Outcome:

The automation script successfully navigated to the login page, located the necessary UI elements, and automated the login process. The script was also able to handle cases where incorrect credentials were provided, ensuring accurate detection of errors.

3. Verify Successful Login

Objective:

Ensure that the login process was successful by verifying the presence of a specific element on the landing page after login.

Steps:

Determine Verification Method:

Successful Login Indicator: Identified a unique element on the landing page that signifies a successful login. In this case, the presence of an element with the data-extension-id='active-visits-tile' attribute was used as an indicator of a successful login.

Implement Verification in the Script:

Successful Login Check: Added a verification step in the script to check for the presence of the unique element on the landing page.

Error Handling for Verification: Implemented error handling to catch cases where the verification element was not found, indicating a potential login failure.

Outcome:

The verification step was successfully implemented. The script was able to confirm a successful login by checking for the presence of the specific element on the landing page. In cases where the element was not found, the script correctly reported that the login might have failed.

4. Error Handling

Objective:

Implement error handling for invalid login attempts and verify that appropriate error messages are displayed for incorrect username/password combinations.

Steps:

Implement Error Handling for Invalid Login Attempts:

Error Detection: Added functionality to detect when an error message is displayed on the login page. This involved identifying the error message element using a CSS selector.

Error Handling Logic: Implemented logic to handle scenarios where an error message appears after submitting incorrect credentials.

Verify Error Messages for Incorrect Login:

Error Message Verification: Configured the script to check for the presence of an error message indicating "Invalid username or password" after submitting the password. Designed this step to confirm that the application displays the correct error message for invalid login attempts.

Error Handling Implementation: Used try-except blocks to manage cases where the error message is not found or where unexpected issues arise during the test.

Challenges Encountered:

Timing Issues:

Description: The script initially had trouble identifying the error message because it appeared immediately after submitting the password, sometimes before the script had a chance to check.

Solution: Adjusted the waiting times and improved the script's logic to ensure it correctly identified the error message after the password submission. This involved fine-tuning the WebDriverWait settings.

Selector Accuracy:

Description: Identifying the correct CSS selector for the error message was challenging due to the dynamic nature of the page and varying classes.

Solution: Refined the selector by inspecting the error message element and using the most stable and reliable attributes to locate it.

Handling Edge Cases:

Description: There were issues handling cases where the error message might not be displayed correctly or other unexpected errors occurred.

Solution: Added comprehensive exception handling to capture and report errors, including screenshots for further analysis.

Outcome:

The error handling implementation was successful despite the challenges. The script was able to correctly identify and report invalid login attempts by detecting the appropriate error message. Adjustments to timing and selector accuracy were crucial in ensuring reliable error handling. Screenshots were used to diagnose issues during script execution.

Challenges Encountered :(continued)

Faced challenges setting up GitHub Actions to run the test scripts because Selenium requires opening a browser window, which was interrupting the build process. Additionally, encountered difficulties configuring Selenium to run in headless mode, which is necessary for CI/CD environments.