interact

Test Automation Technical Test

Blog Post Creation & Verification

Objective: Demonstrate your ability to automate web interactions using Playwright, such as logging into a web application, creating a blog post with text and an image, publishing it, and verifying the post's existence and content.

NOTE: Website URL and credentials will be sent via email.

Tasks:

1. Test Setup:

    - Install Playwright and necessary dependencies.
    - Set up the environment for running the tests.

2. Test Steps:

    - Launch the browser using Playwright.
    - Navigate to the specified URL.
    - Log in using the provided credentials.
    - Navigate to the profile menu and select "Add Blog Post."
    - Fill in the blog post form with the required text and image.
    - Click “Continue” to submit the blog post.
    - Set “Publish” and “Make Feature Post” to Yes
    - Navigate to “All Posts” and validate that the blog post exists.
    - Verify the content of the blog post matches the input data.

3. Expected Results:

    - Each step should execute without errors.
    - The blog post should be visible with the correct text and image.

4. Documentation:

    - Provide a README file with detailed instructions on how to execute the tests.
    - Include any configuration files or setup procedures.

5. Submission:

    - Submit all Playwright test scripts and related files.
    - Provide a video of the test execution showing the browser and Playwright test in action.
    - Zip all your files into one file, upload it to Google Drive, and submit your link to the Talent Acquisition person you have been speaking with.

Guidelines:

- Include assertions to verify each scenario step, such as successful navigation, login confirmation, blog post creation, and post-publication checks.
- Structure your code for readability and maintainability.
- Handle any potential exceptions or edge cases that could occur during test execution.
- Ensure your script is compatible with at least two different web browsers.

Evaluation Criteria:

- Correctness: The script should accurately automate the scenario.
- Code Quality: The script should be well-structured and follow best practices.
- Robustness: The script should handle edge cases and potential test flakiness.
- Documentation: The documentation should clearly explain how to set up and execute the tests and describe what the tests are checking for at each step.

Technical Considerations:

1. Selector Resilience: Choose selectors less likely to change over time, such as data attributes designed explicitly for tests (data-test-id), over brittle ones like CSS classes that may change with redesigns.
2. Sensitive Information Management: Avoid hardcoding sensitive information. Use environment variables or configuration files for credentials and URLs and ensure these are encrypted or stored securely if in a shared repository.
3. Design Patterns: Implement design patterns like the Page Object Model to remove the page structure from the test scripts, enhancing maintainability and readability.
4. Cross-Browser Compatibility: Ensure scripts are functional across multiple browsers, at least Chrome and Firefox, which Playwright supports.
5. Robustness & Flakiness Handling: Include wait conditions or assertions for asynchronous operations to minimise test flakiness.

Notes:

- Successful candidates typically spend between 2-4 hours on this exercise.
- Feel free to include any observations or ideas about how you would extend the solution in your README file.