**QA Engineer Interview Automation Assignment**

In this assignment, you will use **Cypress** (*other frameworks are also acceptable, e.g. PlayWright) to write automated tests for a web application. The application consists of several pages with different types of interactions. You will write tests to ensure that key features of the website are working correctly and efficiently.

**Instructions:**

1. Implement the tests according to the Test Scenarios (see below). Both GUI and API testing.

2. Implement Complexity Additions and Advanced Optional Assignment if the Test Cases weren't challenging enough for you.

3. Save the source code to a Git repository and share it with us. If this is not possible, please zip the file and send it to us. Include instructions for launching the script.

**Objective:**

The objective is to create **Cypress* tests** for the above scenarios, ensuring that the features are tested end-to-end in an automated way. You should write clean, maintainable, and well-structured tests that a real QA Engineer might use for testing the web application.

Web application: https://github.com/metalukask/QA-interview-task

(instuctions on how to locally run the web app are in a README file)

---

**Test 1: Login Functionality**

1. **Test Case 1.1**: **Login with Valid Credentials**
    - Ensure that a user can log in with valid credentials (e.g., admin and admin321).
2. **Test Case 1.2**: **Login with Invalid Credentials**
    - Ensure that a user cannot log in with invalid.
3. **Test Case 1.3**: **Login with Missing Credentials**
    - Ensure that the form displays a validation error if the user submits the form with missing credentials.
4. **Test Case 1.4**: **Login with Valid Credentials API**
    - Ensure that a user can log in with valid credentials.
    - Send a POST request to the login API endpoint with the valid credentials.

**Test 2: Form Submission**

1. **Test Case 2.1**: **Submit Form with Missing Data**
    - Ensure the form doesn't submit if the required fields are empty.

2. **Test Case 2.2**: **Submit Form with Valid Data**
    - Ensure that the form is successfully submitted with valid data.

**Test 3: Button Interaction**

1. **Test Case 3.1**: **Button Text Changes After Click**
    - Test that a text changes whena button is clicked.

**Test 4: Checkbox Interaction**

1. **Test Case 4.1**: **Checkbox Toggling**
    - Test that the checkbox can be toggled on and off correctly.

**Complexity Additions:**

- Assertions
- Additonal tests – there are multiple pages and variations that can be tested
- Reporting – generate a detailed test report that includes pass/fail status and screenshots on failure

---

**Optional Advanced Assignment**

**Testing Drag and Drop Functionality**

The objective of this task is to test the **drag and drop** functionality of a list or element. The test will ensure that the drag-and-drop feature works correctly by simulating the drag action and verifying that the element has been dropped in the correct location.

**Steps to Complete the Assignment:**

1. **Identify the Drag and Drop Area**:
    - Locate the elements that can be dragged and the drop target on the page.
    - Verify that the draggable items are identified and can be moved.

2. **Simulate the Drag-and-Drop Action**:
    - Simulate dragging an item from the list or container.
    - Drop the item into a new target location.
3. **Validate the Drag-and-Drop Functionality**:
    - Verify that after the drag-and-drop operation, the item is positioned correctly.
4. **Edge Case Testing**:
    - Test if the drag-and-drop functionality works even if the item is dragged outside the visible area.