**Instructions for the Candidate:**

Please complete the following tasks using a test automation framework of your choice. Bonus points will be awarded for the use of Playwright. Assume that the person running your tests has no knowledge of your chosen tool, so instructions must be provided on dependencies, installation instructions, and how to execute the tests.

Develop your tests with reusability in mind, creating functions or methods for common actions (like navigating to the page, filling a specific field, etc.).

Submissions are to be made by providing a link to a public GitHub repository containing your test code and instructions.

Test Automation Tasks:

Based on the provided React form application code, automate the following test scenarios:

1. Name Field Tests:
   * **Task 1.1**: Enter Name: Create a test that navigates to the form page, locates the "Name" text field, and enters a specific name (e.g., "Jane Doe").
   * **Validation 1.1**: Console Output: After entering the name and submitting the form, validate that the entered name appears correctly in the browser's developer console output (as the code simulates submission by logging).
   * **Task 1.2**: Empty Name Validation: Create a test that attempts to submit the form without entering anything into the "Name" field.
   * **Validation 1.2**: Required Field Modal: Validate that the validation modal appears and contains the specific message indicating the Name field is required ("Name field is required."). Ensure you can locate and verify the text within this modal.
2. Automation Tools Radio Button Tests:
   * **Task 2.1**: Select "Yes": Create a test to locate and select the "Yes" radio button under the "Have you used automation tools in a real setting?" question.
   * **Task 2.2**: Select "No": Create a test to locate and select the "No" radio button for the same question.
   * **Validation 2.1 & 2.2**: Console Output: After selecting either "Yes" or "No" and submitting the form, validate that the correct value ("yes" or "no") appears in the browser's developer console output.
   * **Locator Consideration**: The radio buttons do not have id attributes. Explain your strategy for reliably locating and selecting these radio buttons in your test code.
3. Years of Experience Slider Tests:
   * **Task 3.1**: Set Value to 3: Create a test that locates the "How many years of automation experience do you have?" slider and sets its value to 3.
   * **Validation 3.1**: Console Output: After setting the value and submitting, validate that "3" appears in the console output for "Years Experience".
   * **Task 3.2**: Set Value to 6: Create a test that sets the slider's value to 6.
   * **Validation 3.2**: Console Output: After setting the value and submitting, validate that "6" appears in the console output.
   * **Task 3.3**: Test Boundary (10+): Create a test that sets the slider's value to 10.
   * **Validation 3.3**: Console Output & Label: Validate that "10" appears in the console output. Also, validate that the text displayed next to the slider correctly shows "10+".
   * **Task 3.4**: Attempt Invalid Value (11+): Describe or demonstrate how you would attempt to set the slider value to 11 (or any value outside the 0-10 range) using your automation tool. Explain what you would expect to happen based on standard HTML input behavior and the provided code.
4. CV File Upload Test:
   * **Task 4.1**: Attach File: Create a test that locates the "Please attach your CV" file upload field and simulates attaching a dummy file (you do not need to provide a real file in your repository, just the test code to handle a theoretical file).
   * **Validation 4.1**: File Name Display: Validate that the name of the simulated file appears on the page next to the file upload input after the action.
5. Personal Statement Text Area Tests:
   * **Task 5.1:** Enter Text: Create a test that locates the "Personal Statement" text area and enters a short paragraph of text.
   * **Validation 5.1**: Text Content & Count: Validate that the text area contains the entered text. Validate that the character count displayed below the text area correctly reflects the number of characters entered.
   * **Task 5.2**: Test Boundary (Max Length): Create a test that enters exactly 100 characters into the text area. Validate the text content and the character count (should be "100 / 100").
   * **Task 5.3**: Test Exceeding Max Length: Describe or demonstrate how you would attempt to enter more than 100 characters. Explain what you would expect to happen based on the maxLength attribute in the provided code.
6. Completed Task Checkbox Test:
   * **Task 6.1**: Check and Uncheck: Create a test that locates the "Completed the task" checkbox. First, click it to check it, and then click it again to uncheck it.
   * **Validation 6.1**: State Change: Validate the checked/unchecked state of the checkbox after each click.

Additional Tests (Optional):

If you feel there are other important test cases for this form application that are not covered above, please describe or implement them. Consider aspects like form reset, accessibility checks (e.g., tab order), or testing the modal's closing functionality.

Also please make sure you note down (in your readme, for example) anything you have observed which isn’t quite right, or you would question if this was delivered to you by a colleague. (And also include any assumptions which you have made.)

Remember to include clear instructions in your GitHub repository on how to set up the environment, install dependencies, and execute your test suite.