Unit Test - Templates

Test Cases

Test Case 1: Display Loading Indicator

- **Objective:** Validate that the Templates component correctly displays a loading indicator while fetching data.
- **Method:** `render`
- **Test Steps:**
  a. Mock the API to delay the response.
  b. Render the Templates component within a BrowserRouter.
  c. Check for the presence of the "Loading..." text immediately after rendering.
- **Expected Result:** The "Loading..." text is visible upon initial render, indicating that data fetching is in progress.

Test Case 2: Display Templates After Fetching

- **Objective:** Ensure that the Templates component correctly displays templates after data is fetched.
- **Method:** `waitFor`
- **Test Steps:**
  a. Mock the API to resolve with a list of templates.
  b. Render the Templates component.
  c. Wait for the templates to be displayed.
- **Expected Result:** The component should display each template's name, confirming that data is successfully fetched and rendered.

Test Case 3: Handle Fetching Errors

- **Objective:** Verify that the Templates component handles errors during the data fetching process.
- **Method:** `waitFor`
- **Test Steps:**
  a. Mock the API to reject with an error.
  b. Render the Templates component.
  c. Wait for an error message to be displayed.
- **Expected Result:** An error message "[ERROR] Failed to fetch templates." is shown, indicating proper error handling.

Test Case 4: Navigate on Template Selection

- **Objective:** Test the navigation functionality when a template is selected.
- **Method:** `userEvent.click`
- **Test Steps:**
  a. Render the Templates component with a list of templates.
  b. Simulate a user click on one of the template buttons.
  c. Check the navigation function is called with the correct path.
- **Expected Result:** Upon clicking a template button, the application navigates to the detailed view of the selected template.