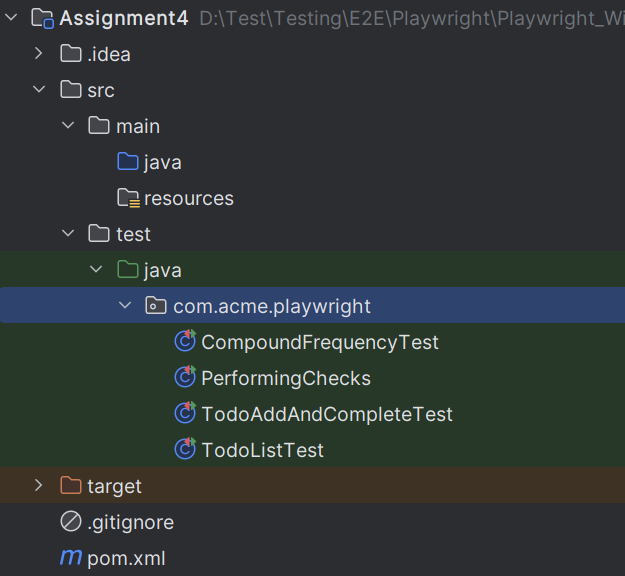
# Playwright for Java - Hands-On Assignment 4

# Project Structure



# Problem Statement 1: Automating a React-based TodoMVC App with Playwright Fixtures

## Objective:

Practice using Playwright’s JUnit 5 @UsePlaywright fixtures to test a modern JavaScript web app (React TodoMVC), ensuring core to-do list functionalities work as expected in a browser run in non-headless mode.

## Steps:

1. Use the @UsePlaywright annotation to enable Playwright’s built-in JUnit 5 fixture support for automatic Page injection.
2. Write the following test cases in the same class:

### Test 1: Verify Empty List on Load

a. Navigate to https://todomvc.com/examples/react/dist/

b. Assert that the .todo-list li elements count is 0.

### Test 2: Add a New Todo

a. Navigate to the TodoMVC app.

b. Fill the .new-todo input with "Learn Playwright" and press Enter.

c. Assert that the todo list contains exactly 1 item.

d. Assert that the first todo item contains the text "Learn Playwright".

### Test 3: Mark a Todo as Completed

a. Add a todo "Complete me".

b. Click the first .toggle checkbox.

c. Assert that the corresponding <li> element has a CSS class containing "completed".

### Test 4: Clear Completed Todos

a. Add a todo "To be cleared".

b. Mark it as completed and click the "Clear completed" button.

c. Assert that the todo list is now empty.

### Test 5: Filter Active and Completed Todos

a. Add two todos: "Active task" and "Completed task".

b. Mark the second task as completed.

c. Click the "Active" footer link and assert only "Active task" is shown.

d. Click the "Completed" footer link and assert only "Completed task" is shown.

## Expected Output:

* Test 1: Passes if the todo list is empty on initial page load.
* Test 2: Passes if the new todo appears correctly in the list.
* Test 3: Passes if the completed todo has the "completed" CSS class.
* Test 4: Passes if clearing completed todos removes them from the list.
* Test 5: Passes if the filters correctly show only active or completed todos.

# Problem Statement 2: Automating a React TodoMVC App with Playwright Fixtures

## Objective:

Practice automating a basic "Add and Complete Todo" flow in the React-based TodoMVC app using Playwright’s JUnit 5 fixtures, ensuring that newly added items appear correctly and can be marked as completed.

## Steps:

1. Use the @UsePlaywright annotation to enable Playwright’s built-in JUnit 5 fixture support for automatic Page injection.

1. Write a test method to:

**Test: Verify Adding and Completing a Todo**

a. Navigate to the React TodoMVC app at https://todomvc.com/examples/react/dist/.

b. Fill the new todo input (.new-todo) with the task "Buy milk" and press Enter.

c. Verify the newly added todo item appears in the list.

d. Mark the todo as complete by checking the toggle box (.toggle).

e. Assert that the completed todo has the CSS class "completed".

**Expected Output:**

Test passes if:

* The new todo "Buy milk" appears in the list.
* The todo’s HTML element has the "completed" class after being marked complete.

# Problem Statement 3: Automating a Textarea Submission and Verification with Playwright Fixtures

## Objective:

Practice using Playwright’s JUnit 5 @UsePlaywright fixtures to test textarea input on a live demo form page, ensuring the entered text is captured correctly after form submission.

## Steps:

1. Use the @UsePlaywright annotation to enable Playwright’s built-in JUnit 5 fixture support for automatic Page injection.
2. Navigate to the given sample page URL: https://deformdemo.pylonsproject.org/textarea/
3. Locate the <textarea> element using a CSS selector.
4. Assert that the textarea is visible and within the viewport.
5. Assert that the textarea is initially empty.
6. Fill the textarea with the message "Hello Demo".
7. Assert that the textarea value matches "Hello Demo".
8. Click the submit button.
9. Locate the <pre> block with id="captured" showing the submitted data.
10. Assert that this captured block contains the entered message "Hello Demo".

## Expected Output:

* The textarea is empty initially.
* The textarea value updates correctly after filling.
* The captured output on the page contains "Hello Demo" after submission.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*