1. **Setting Up the Environment**

Installed Playwright and TypeScript in the project directory using below command in Git Bash.

npm init -y

npm install playwright

npm install typescript ts-node @types/node

1. **Installed VS Code as an editor.**
2. **Created a tsconfig.json File**
3. **Test Automation Code (Page Object Model)**

Created a folder structure with the following files:

* tests/lead\_tests.ts (main test runner)
  + The LeadTests class handles the setup (browser launch and login), execution of the tests, and teardown (closing the browser).
  + The runTests() method runs the four required test cases sequentially:
    - **Creating a new lead**.
    - **Editing the lead**.
    - **Filtering the leads by company**.
    - **Deleting the lead**.
* pages/lead\_page.ts (page object model for the Leads module)
  + This file contains the methods that handle each operation on the Leads module (create, edit, filter, and delete). This helps in keeping the tests modular and reusable.
* credentials.ts (to keep dynamic values in single file)

1. **Running the Tests**

To run the tests used below-mentioned command in gitBash:

npx ts-node tests/lead\_tests.ts

1. **Run the Tests with Jest**

You can now run the tests using Jest:

npx jest

or with specific test files:

npx jest tests/lead\_tests.ts

1. **Reporting**

Adding **Jest** to your **Playwright** test cases is a good idea to enhance test assertions, reporting, and overall test management.

* Install Jest and Required Dependencies

First, you need to install Jest and related dependencies.

Run the following command in your project directory:

npm install jest ts-jest @types/jest --save-dev

* Configure Jest for TypeScript

Create a jest.config.js file in the root of your project to configure Jest for TypeScript and the Playwright environment.

1. **Additional Considerations**
   * Error Handling: If there are steps that might fail (e.g., login), you can handle them by adding proper error handling and assertions.
   * Test Reports: Jest can be configured to generate reports or integrate with CI/CD tools like Jenkins, Travis, or CircleCI for continuous testing.
2. **Completion Checklist**
   * Automated Tests
   * Successfully Running Tests
   * Github Repo with Readme
   * Test Run Results
   * Submit to Greenhouse - Link to the github