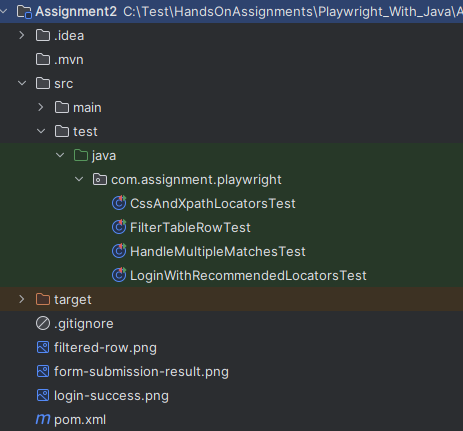
# Playwright for Java - Hands-On Assignments 2

# Project Structure



### **Problem Statement 1: Practice Using Recommended Locators**

**Objective:** Learn to interact with form elements using Playwright’s recommended semantic locators like getByLabel, getByRole, and getByText, ensuring better test readability and stability.

**Steps:**

1. **Create a test script** – LoginWithRecommendedLocatorsTest.java.
2. **Navigate to**:  
    <https://practicetestautomation.com/practice-test-login/>
3. **Fill in the form using recommended locators:**
   1. Use getByLabel() to enter:
      1. Username: student
      2. Password: Password123
   2. Use getByRole() to click the **Submit** button (role: button, name: "Submit").
4. **Assert login success:**
   1. Use getByText() to check that the message "Logged In Successfully" is visible.
   2. Alternatively, assert that the current URL contains /logged-in-successfully/.
5. **Use the assertion style:**

assertThat(page.getByText("Logged In Successfully")).isVisible();

1. **Take a screenshot** after successful login and save it as login-success.png.

**Deliverable:**

* A runnable Java test class named LoginWithRecommendedLocatorsTest.java that:
  + Uses Playwright semantic locators (getByLabel, getByRole, getByText)
  + Asserts successful login
  + Captures and stores a screenshot of the result

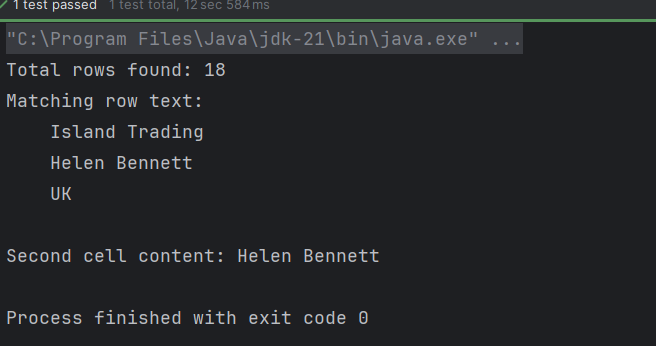
### **Problem Statement 2: Filter Table Rows Using Role-Based Locators**

**Objective:** Practice locating table rows and cells using getByRole() and refining them with filter() using hasText.

**Steps:**

1. **Create a test script** – FilterTableRowTest.java.
2. **Navigate to**:  
    <https://www.w3schools.com/html/html_tables.asp>
3. **Use getByRole(ROW)** to:
   1. Select all rows in the table.
   2. Print the total number of rows using .count().
4. **Filter the row containing specific text**:
   1. Use .filter(new Locator.FilterOptions().setHasText("Island Trading")) to locate the row.
5. **Print the filtered row’s full text** using .textContent().
6. **Extract and print the second cell** (i.e., Contact Name) using .getByRole(CELL).nth(1).

**Expected Output:**



**Deliverable:**

* A test script FilterTableRowTest.java that:
  + Uses semantic locators to interact with a table
  + Filters a row by content
  + Extracts and logs specific cell data

### **Problem Statement 3: Automate Form Interaction Using CSS and XPath Locators**

**Objective:** Practice locating web elements using **CSS selectors** and **XPath expressions** instead of semantic locators like getByLabel() or getByRole().

**Steps:**

1. **Create a test script** – CssAndXpathLocatorsTest.java.
2. **Navigate to**:  
    <https://www.techlistic.com/p/selenium-practice-form.html>
3. **Using only CSS and XPath locators**:
   1. Fill in the **First Name** and **Last Name** fields.
   2. Select the **Gender** and **Years of Experience** radio buttons.
   3. Enter a date in the **Date** field.
   4. Choose a profession checkbox (e.g., "Automation Tester").
   5. Click the **Submit** button.

# **Problem Statement 4: Handling Multiple Matches with getByRole**

URL to Use:  
<https://www.w3schools.com/html/html_forms.asp>

In this task, you are required to explore how Playwright handles locators that match multiple elements on a webpage. Specifically, you will be working with the getByRole(AriaRole.BUTTON) locator, which is likely to return multiple matches on the given page.

## **1️⃣ multipleMatchesStrictModeError()**

- Navigate to the given URL.  
- Attempt to click a button using a broad locator: getByRole(BUTTON).  
- Since multiple buttons exist on the page, this action will cause a strict mode violation error.  
- Catch this error and print a message indicating that multiple matches were found and the error was expected.

***What to print:***

A message like:  
"Expected error due to multiple matches: <actual error message>"

## **2️⃣ handleMultipleMatchesWithIndexing()**

- Again, navigate to the same URL.  
- Use getByRole(BUTTON) to get all matching buttons.  
- From the set of buttons, extract and print the following:

***What to print:***

- The total number of buttons found.  
- The text of the first button.  
- The text of the last button.  
- The text of the second button (i.e., the one at index 1).

## **Expected Output**

* For multipleMatchesStrictModeError():

Expected error due to multiple matches: Error {  
 message='Error: strict mode violation: getByRole(AriaRole.BUTTON) resolved to 10 elements:

* For handleMultipleMatchesWithIndexing():

Total buttons found: 10  
First button text:   
 Tutorials  
   
   
   
Last button text: Submit Answer »  
Second button text:   
 Exercises

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