DAY 6 – Playwright

1. Playwright Installation

npm init playwright@latest

2. Page Object Model (POM)

It is a design pattern commonly used in **automation** testing

- Used with tools like **Playwright**, Selenium, Cypress
- Makes Tests more **maintainable**, **reusable**, and **readable**
- **test logic is separated from UI details.**
  In POM, each page or component of your application has a corresponding class (or object) that encapsulates:
- The **locators (selectors)** for elements on the page
- The **actions** you can perform on the page
- Any **assertions** or checks related to that page

3. Page Navigation

```
await page.goto('https://www.easemytrip.com/');
```

4. Title Assertion:

```
await expect(page).toHaveTitle(...)

//ensures the correct page is loaded.
```

5. Click Actions with Visibility Check

Always check visibility before clicking for stability:

```
await expect(button).toBeVisible();
await button.click();
```

6. Waiting Strategies

Use **waitForTimeout()** for debugging or better yet, wait for specific elements or states.

7. **Element locators**

XPath is powerful but use **CSS selectors** when possible for better performance.

```
const getStarted = page.getByText('Try Typescript Now');
const username = page.locator("#user-name");
```

8. **Form Interactions / Filling Inputs**

```
await username.fill("standard_user");
await password.fill("secret_sauce");
```

Features:

- Auto wait
- Cross browser Compatibility – Chrome, Firefox, Edge, Webkit.
- Multi-platform – Mac OS, Windows, Linux
- Multilingual Flexibility – Supports Java, Python, Java Script, C#, .NET

Advance Feature:

- Tracing and Debugging – take screenshot, video recording
- Network Interception
- Browser Context Management
- Codegen Tool
- Java Script – Asynchronous this one does not execute the order wise.
- **await – used to wait each action**

Installation of Playwright:

- npm install playwright@latest

Run the test:

- **npx playwright test**

Run the test headed mode:

- **npx playwright test –headed**

Run the test UI mode:

- npx playwright test –ui

Get the report in html page:

- npx playwright show-report

Import the playwright and expect in .ts file

- import {test, expect} from "@playwright/test";

1. **Navigate to the website**

```
import { test, expect } from '@playwright/test';

test("Place the Order", async({page})=>{
    await page.goto("https://www.saucedemo.com/");
```

2. **Assertion in Playwright:**

```
const orderConfirmation = await page.locator(".complete-header");
await expect(orderConfirmation).toHaveText("Thank you for your order!");
```

3. **Different type of assertion:**

await expect(locator).toBeChecked();

await expect(locator).not.toBeChecked();

await expect(locator).toBeDisabled();

await expect(locator).toBeEnabled();

await expect(locator).toBeEditable();

await expect(locator).toBeEmpty();

Example Automation Script:

```
test('End to end test for cart functionality', async ({ page }) => {
    await page.goto('https://www.saucedemo.com/');
    await expect(page).toHaveTitle("Swag Labs");
    const username = page.locator("#user-name");
    await username.fill("standard_user");
    const password = page.locator("#password");
    await password.fill("secret_sauce");
    const login = page.locator("#login-button");
    await login.click();
    const firstItem = page.locator(".inventory_item_name").first();
    await firstItem.click();
    const addToCart = page.locator("#add-to-cart");
    await addToCart.click();
    //I will now verify cart quantity is equals to one
    const cartIcon = page.locator(".shopping_cart_badge");
    await expect(cartIcon).toBeVisible();
    const cartQuantity = await cartIcon.textContent();
    if (cartQuantity === "1") {
        console.log("Cart quantity is 1");
    }
    cartIcon.click();
    const checkoutBtn = page.locator("#checkout");
    await expect(checkoutBtn).toBeVisible();
    await checkoutBtn.click();
    await page.fill("#first-name", "John");
    await page.fill("#last-name", "Doe");
    await page.fill("#postal-code", "12345");
    await page.click("#continue");
    const finishBtn = page.locator("#finish");
    await expect(finishBtn).toBeVisible();
    await finishBtn.click();
    const orderConfirmation = page.locator(".complete-header");
    await expect(orderConfirmation).toBeVisible();
    const orderText = await orderConfirmation.textContent();
    await expect(orderText).toContain("Thank you for your order!");


})
```

End-to-End Cart Functionality Test

This automated test script, written using Playwright, verifies the full cart and checkout flow on the **Swag Labs** demo website. The test covers the following steps:

1. **Navigation & Login:**

- Navigates to https://www.saucedemo.com/
- Asserts that the page title is "Swag Labs"
- Logs in using standard credentials: standard_user / secret_sauce

2. **Item Selection & Cart Interaction:**
   - Clicks on the first listed item
   - Adds the item to the cart
   - Asserts the cart badge is visible
   - Confirms that the cart quantity is correctly updated to **1**
3. **Checkout Process:**
   - Proceeds to the cart and initiates checkout
   - Fills out checkout information with mock user data (name and postal code)
   - Continues to the order summary and completes the purchase
4. **Order Confirmation:**
   - Verifies that the final confirmation message appears
   - Asserts the confirmation contains the expected message: *"Thank you for your order!"*

This test ensures that the cart functionality—from product selection to successful order placement—works as intended.