Apply here:
https://genesys.wd1.myworkdayjobs.com/Genesys/job/Anderson-Lab/QA-Software-Engineering-Intern_JR102846

Getting started with Playwright/automated UI testing

- Installing Node
    - Go to https://nodejs.org/en/download/ and download and install the recommended version of Node for your operating system
- Installing Visual Studio Code(Different than Visual Studios)
    - Go to https://code.visualstudio.com/Download and download the version of VS code for your system.
    - Once you have VScode installed you want to navigate to extensions (The tab with four squares
    - **Optional:** install Prettier-Code Formatter
        - How to configure prettier
            - User Settings
                - Windows: Ctrl-Shift-P
                - Mac: Cmd-Shift-P
                - Preferences: Open Settings (JSON)

```
{
    "editor.formatOnSave": true,
    "editor.wordWrapColumn": 120,
    "prettier.printWidth": 120,
    "files.autoSave": "afterDelay",
    "window.zoomLevel": 1,
    "prettier.tabWidth": 4,
    "javascript.updateImportsOnFileMove.enabled": "always",
   "editor.defaultFormatter": "esbenp.prettier-vscode",
   "prettier.trailingComma": "none"
   }
```

- Paste in above JSON config:

Creating your project

- Create a folder/directory in an easy to locate place called playwright
- Once you have the directory created go back to VScode and open the folder
- Create a new file inside of that directory. Name the file InterviewTest.spec.js
- Once you have the file created you want to navigate to your terminal/command prompt and cd to the directory you made.

- Once you are in the directory you will want to type the following commands:
    - npm init playwright@latest
    - Choose between TypeScript or JavaScript (default is TypeScript)
    - Name of your Tests folder: "playwright" in this case

Writing your first test

- Navigate to InterviewTest.spec.js in vscode
- You want to start by declaring what modules you want to use for the test
    - Playwright has you generally use two which can be declared by const { test, expect } = require('@playwright/test');
- Use the links below to mess around with playwright and see what you can come up with
- How to run the test

    - **Click terminal →new terminal in VScode and type "npx playwright test InterViewTest.spec.js --headed"**
    - You can also remove the headed at the end to run the test headless if desired
    - If you run into issues and want to see your test run step by step you can replace --headed with --debug to run the test in debug mode

Example "Hello World" UI test

```
const { test, expect } = require('@playwright/test');


test('WikiPedia Page has Hello World Title', async ({ page }) => {
   await page.goto('https://en.wikipedia.org/wiki/%22Hello,_World!%22_program');
   // create a locator
   let header = page.locator('.mw-page-title-main');
   // Expect header to have correct text
   await expect(header).toHaveText('"Hello, World!" program')
});
```

- Where to go from here?
    - Once you have your first test written and passing you will want to start writing your own tests
    - To do this you will want to create a new .spec.js file in the same directory as InterviewTest.spec.js(Name this file something descriptive to what your test actually does)

    - Once you have the new file created write your test and run it the same way as you did InterviewTest.spec.js
- **Required**
    - Automate the following tests for the website https://magento.softwaretestingboard.com/
        - Take the example below. Make the test verify that it logged in successfully (or made it to the next page)
        - Try some other use cases and automate them (in separate files). These will all require logging in first
            - add an item to the cart. verify it got added
            - view an item and cancel out of it
            - see if you can complete a purchase and verify it
            - verify that the compare feature works on different products
            - come up with your own
        - Some important notes about this:
            - Make sure you have a separate file per use case
            - Try and make your use cases as specific as possible(In other words only test one thing)
            - make sure to use assertions/expect statements in each test. It's not a test if you're not verifying an expected outcome.

Useful Links	
Playwright Documentation	https://playwright.dev/docs/intro
Selectors	https://playwright.dev/docs/selectors

```
"use strict";

// @ts-check
const { test, expect } = require("@playwright/test");

 test("Standard User Can Log Into ", async ({ page }) => {
    /**
     * In Chrome, right click on an element to inspect:
     *
     *  You will interact with items by the ID, class, or other elements.
     *
     *  to find an element by ID, you'll use the character: #
     *  to find an element by the class, you use the character: .
     *  to find by another attribute, (example, data-test), you'll put the full field name = the value you're looking for in brackets
     *       ex: [data-test='some-value']
     *
 *  javascript variable declarations:
     *  - var -   it's "unscoped" meaning that it's not very safe to use
     *  - let - it's a "block" scope, meaning that it's safe within the context of your code.  Use this when you have a variable that you might need to change
     *  - const - block scope declaration that you cannot reassign after initial assignment
     *
     */

    await page.goto("https://saucedemo.com");

    //verifying that we found the logo (we're on the right page)
    const logInButton = page.locator("#login-button");

    // Compare the text of selected element to expected text
    await expect(logInButton).toHaveText("Login");

    // verify that the first element you're interacting with is on the screen
    await page.waitForSelector("#user-name");

    // type in user name & password
    await page.locator("#user-name").fill("standard_user");
    await page.locator("#password").fill("secret_sauce");

    //click the login button
    await logInButton.click();

    // verifying new page has been reached
```

```
    // todo: enter your code here
});
```