Intro

Automated testing is a vital component for larger projects, where multiple developers are creating solutions in multiple streams, ensuring quality and essentially not breaking what has gone before adds a confidence in the delivery process.

Triggering automated test scripts as part of the build process reduces manual efforts and greatly reduces the impact of change to projects of any size.

Power Apps comes with the Power Apps Test Studio, but it is limited (currently) to testing Canvas Apps. This is a great starting point if you are only worried about canvas apps.

Playwright is an open source framework, created by Microsoft to allow testing across multiple browsers and websites, and will work for Model Driven apps, first party apps (such as Customer Service and Field Service), Canvas Apps and Power Pages. It is ideal if your solution spans multiple parts of the platform.

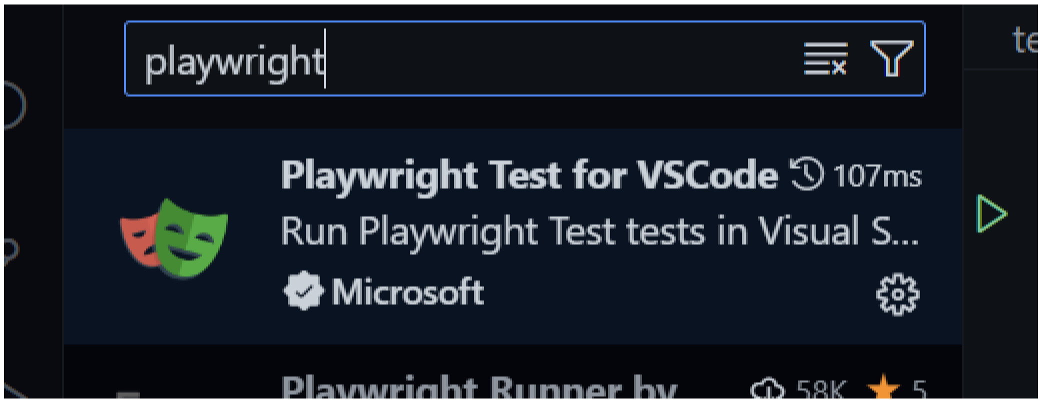
Using Power Apps Test Studio

1. Create a test script in Test Studio

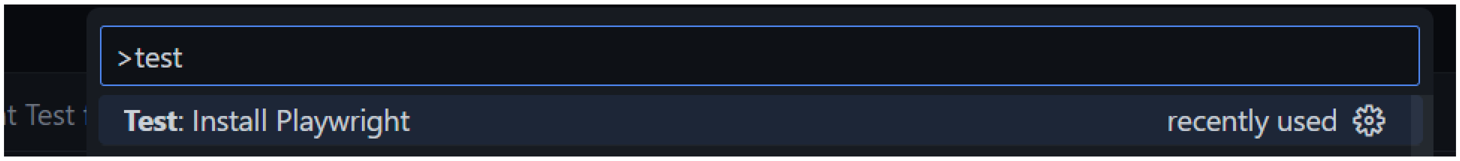
Using Playwright

1. Install Playwright

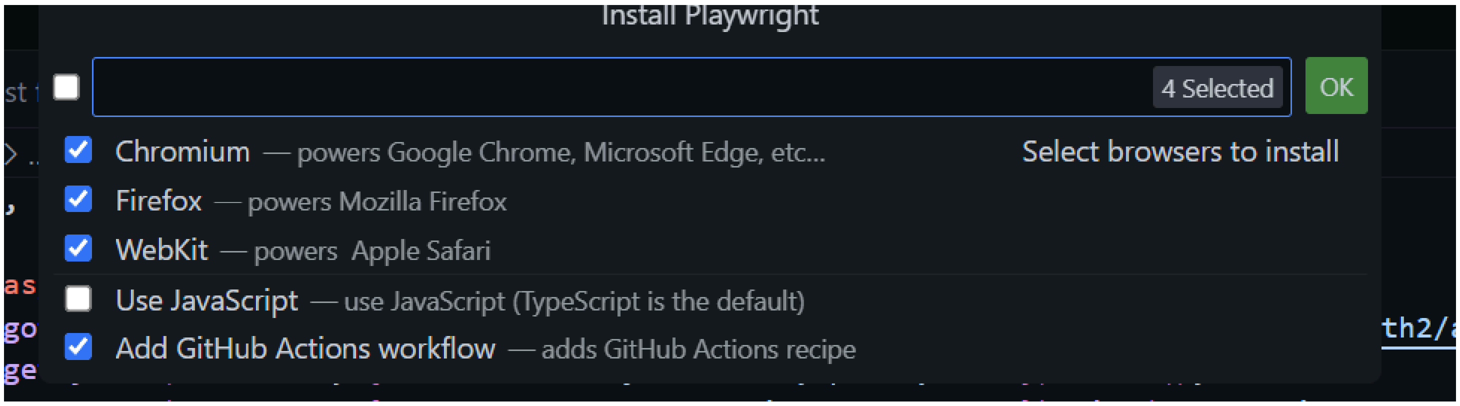
1. Open the Extensions Window (Ctrl + Shift + X)
2. Search for Playwright and select Playwright Test for VSCode
3. Withing VSCode create a new Window (Ctrl + Shift + N) and create a new folder (Ctrl + K, Ctrl + O) called TestScripts
4. Install the Playwright Extension for VSCode


playwright
Playwright Test for VSCode 107ms
Run Playwright Test tests in Visual S...
Microsoft
Playwright Runner by 58K 5

5. Once installed, open Commands (Ctrl + Shift + P) and type Test and run the command Test>: Install Playwright



6. In the menu, select all 3 browsers, leave Use Javascript blank (we want to use the default Typescript) and add the Github Actions workflow checkbox filled


Install Playwright

4 Selected OK

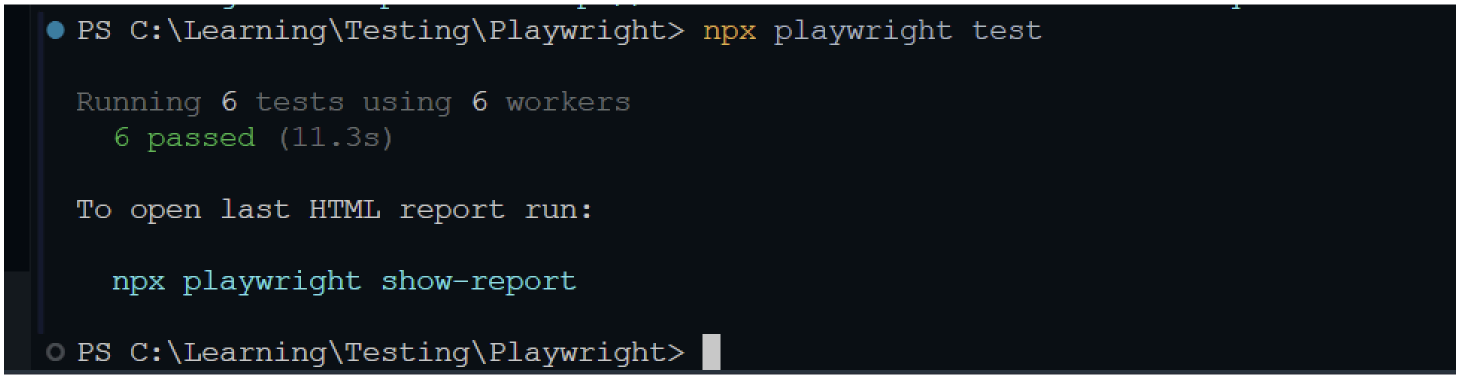
Select browsers to install

- [✓] Chromium — powers Google Chrome, Microsoft Edge, etc...
- [✓] Firefox — powers Mozilla Firefox
- [✓] WebKit — powers Apple Safari
- [ ] Use JavaScript — use JavaScript (TypeScript is the default)
- [✓] Add GitHub Actions workflow — adds GitHub Actions recipe

7. This will create a sample test script in your new folder and allow you to run the sample test scripts.

```
npx playwright run
```

This should run 6 tests using 6 workers as defined in the example.spec.ts file

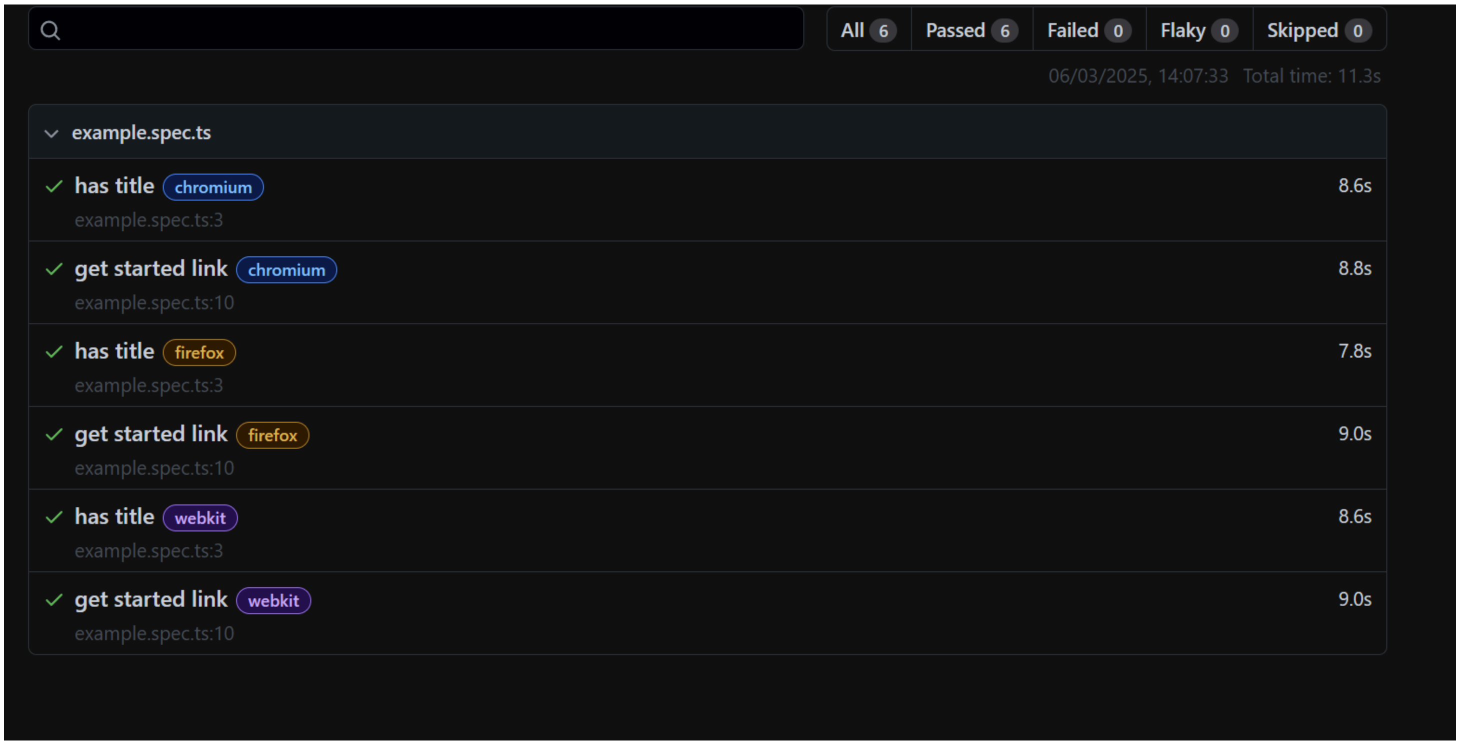


```
PS C:\Learning\Testing\Playwright> npx playwright test

Running 6 tests using 6 workers
  6 passed (11.3s)

To open last HTML report run:

  npx playwright show-report

PS C:\Learning\Testing\Playwright>
```

8. Once returned, take a look at the results

```
npx playwright show-report
```

This will display a webpage with the results for you to dissect.



All 6 | Passed 6 | Failed 0 | Flaky 0 | Skipped 0

06/03/2025, 14:07:33 Total time: 11.3s

example.spec.ts

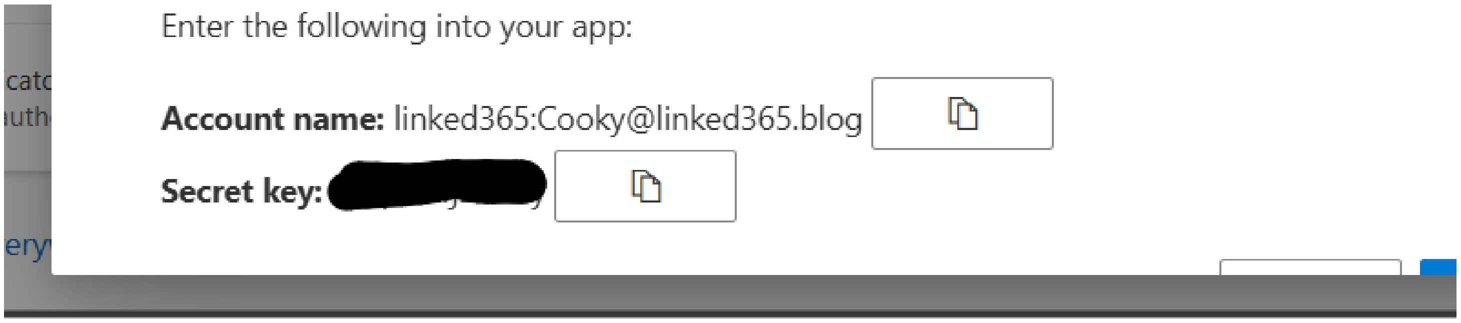
✓ has title chromium example.spec.ts:3	8.6s
✓ get started link chromium example.spec.ts:10	8.8s
✓ has title firefox example.spec.ts:3	7.8s
✓ get started link firefox example.spec.ts:10	9.0s
✓ has title webkit example.spec.ts:3	8.6s
✓ get started link webkit example.spec.ts:10	9.0s

2. Configure MFA for Playwright

For the Power Platform, you usually log in and use Multi-factor Authentication (MFA) to secure your account, and this is becoming the default. As Playwright can not interact with your secondary device, the authenticator app on your phone, we have to do a little work around (suitable for POC, but for your real-life projects, you should use a dedicated test account, with very limited access to your dev system)

For our lab, we will be adding a Timed One Time Passcode method, which will mimic getting a code from your authenticator.

1. Navigate to https://mysignins.microsoft.com/security-info in the browser you have been connecting to Power Platform with.
2. Select the Add sign-in method button.
3. Choose Microsoft Authenticator
4. Select 'I want to use a different authenticator app'
5. Select Next
6. Select Can't scan image
7. Copy and keep safe the Secret Key, you will need this later.



Enter the following into your app:

Account name: linked365:Cooky@linked365.blog

Secret key: [redacted]

●

8. Open your authenticator app on your phone and scan the QR code.
9. Click Next
10. Enter the code from your authenticator app and click Next.

This should now appear as another authenticator in your list.

> [!WARNING]
> Enusre you remove this authenticator when you are done. Whilst you need the secret, it should not be an option.

> {!TIP] All these secrets and passwords are passed as plain text in our lab, in a production and alm scenarios, these will be kept within a key vault to ensure they are not shared.

Create a starter Test Script

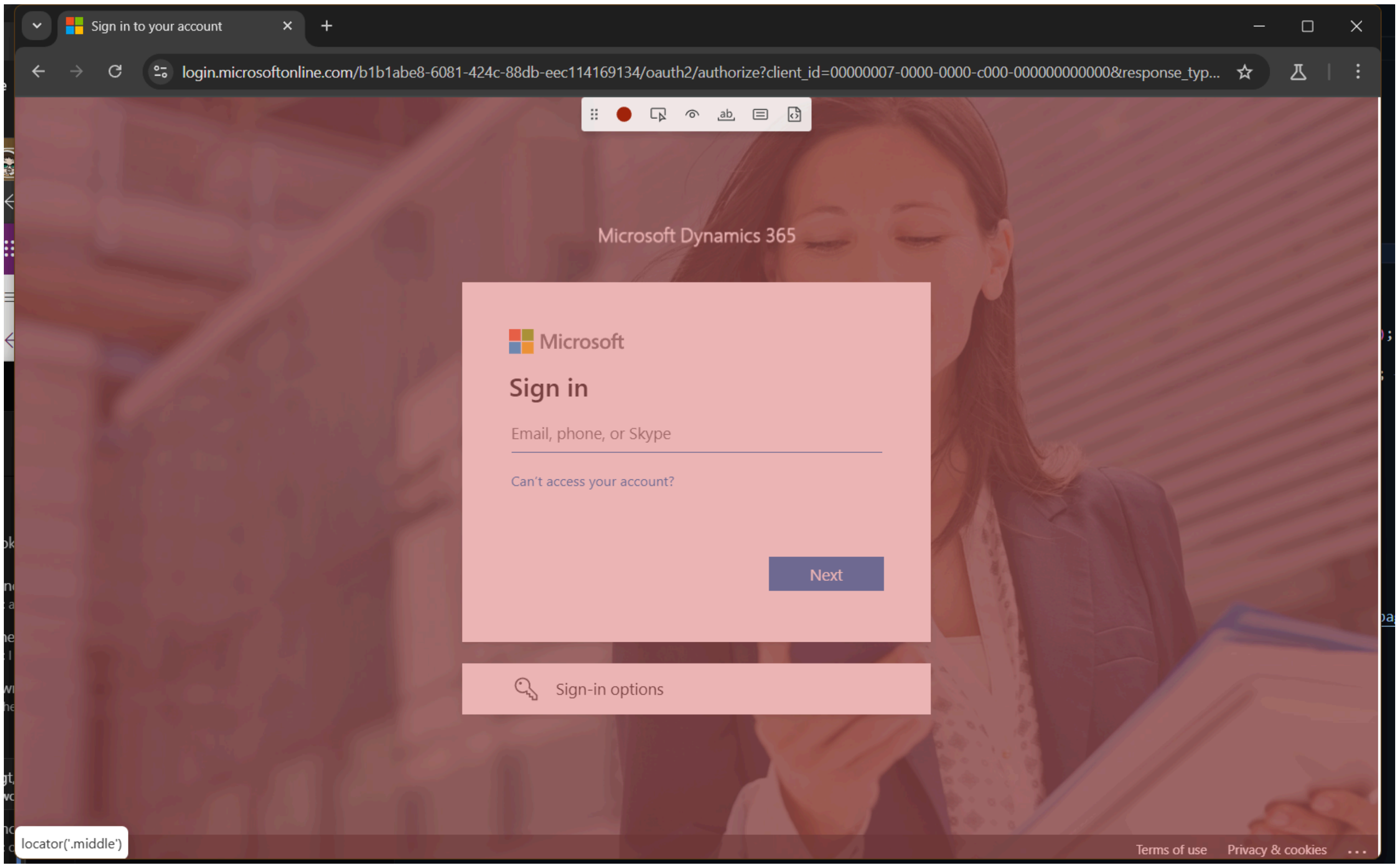
To enable automated testing, you need to tell Playwright what to do and what to check once you have done it.

Playwight can create scripts from your actions in a browser and we will use this technique to firstly authenticate then expand to ensure what we test is accurate.

1. Open the recording harness with your org

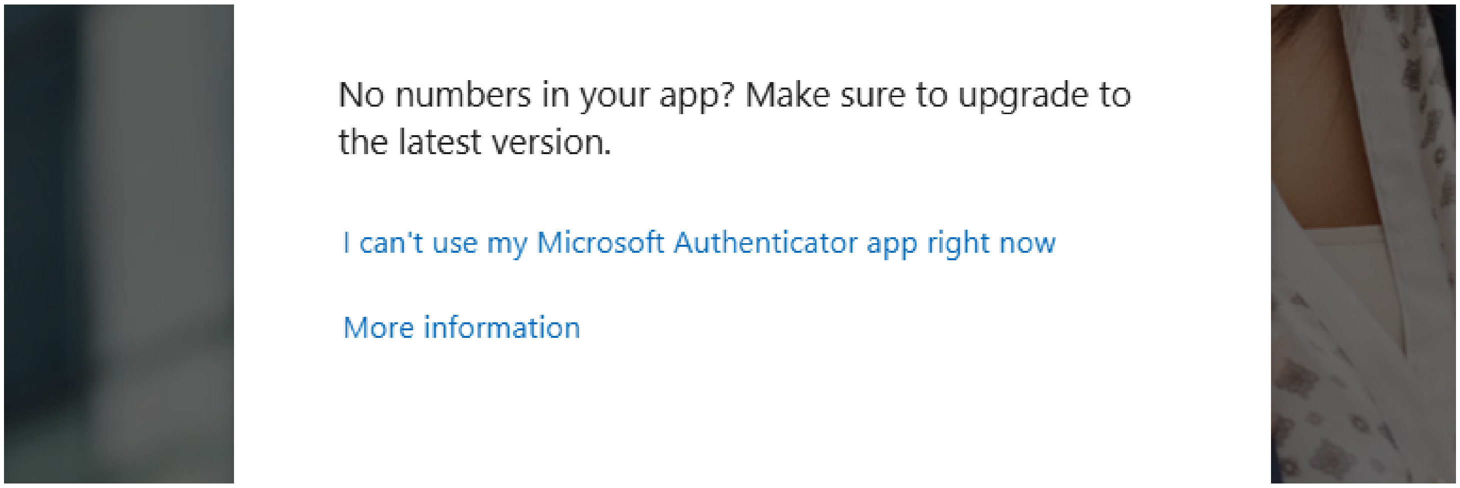
```
npx playwright codegen https://<your org>.crmXX.dynamics.com
```

This should open up a chromium (Not Chrome or Safari etc) browser with a log in box


Sign in to your account
login.microsoftonline.com/b1b1abe8-6081-424c-88db-eec114169134/oauth2/authorize?client_id=00000007-0000-0000-c000-000000000000&response_typ...
Microsoft Dynamics 365
Microsoft
Sign in
Email, phone, or Skype
Can't access your account?
Next
Sign-in options
locator('.middle')
Terms of use
Privacy & cookies

As you move your cursor around, the selection items are highlighted and the Playwright inspector shows you what it wants to do.

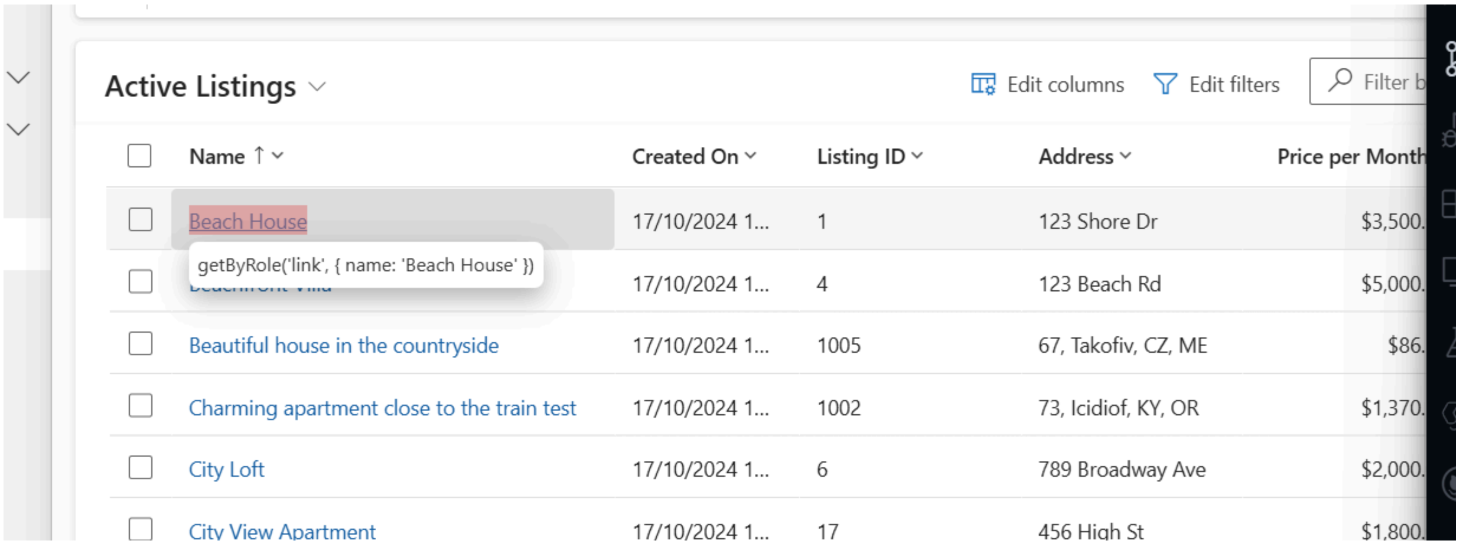
2. Enter your Email and hit next
3. Enter your password and select Sign in
4. This should present you with the standard Approve Sign in Request box.
5. Select the 'I can't use my Microsoft Authenticator app right now'


No numbers in your app? Make sure to upgrade to the latest version.

I can't use my Microsoft Authenticator app right now

More information

6. Select Use a verification code
7. In your authenicator app, select the profile for your app you created earlier(you will have a popup for the normal route, but ignore that) and enter the code that is displayed, then choose Verify
8. Select Yes on the 'Stay Signed in?' Page
9. You should be presented with the App selector screen, lets go to the Contoso Real Estate Admin app we imported.
10. Select the first listing to go to the record (It depends on what records you created in the pre-requisites stages on what you see here)

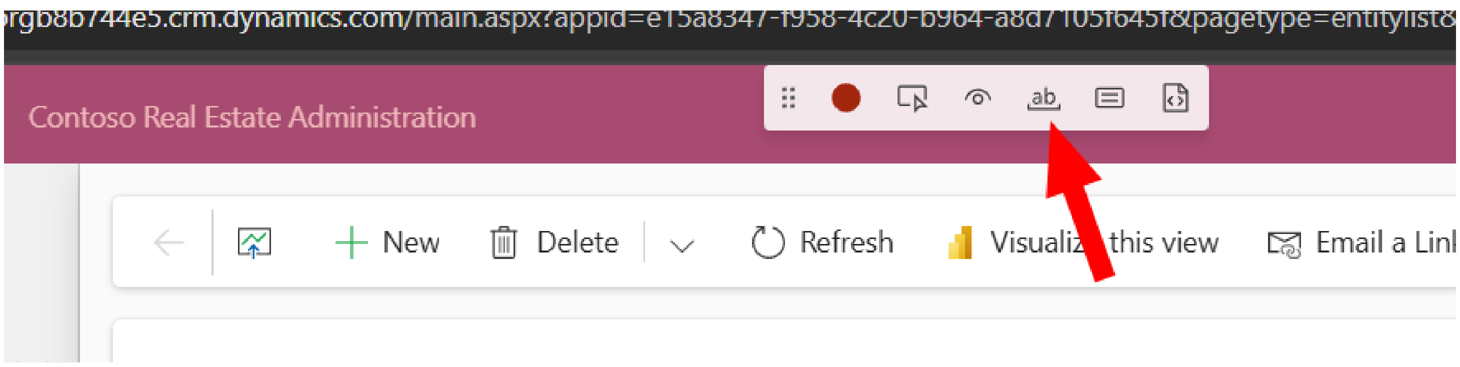


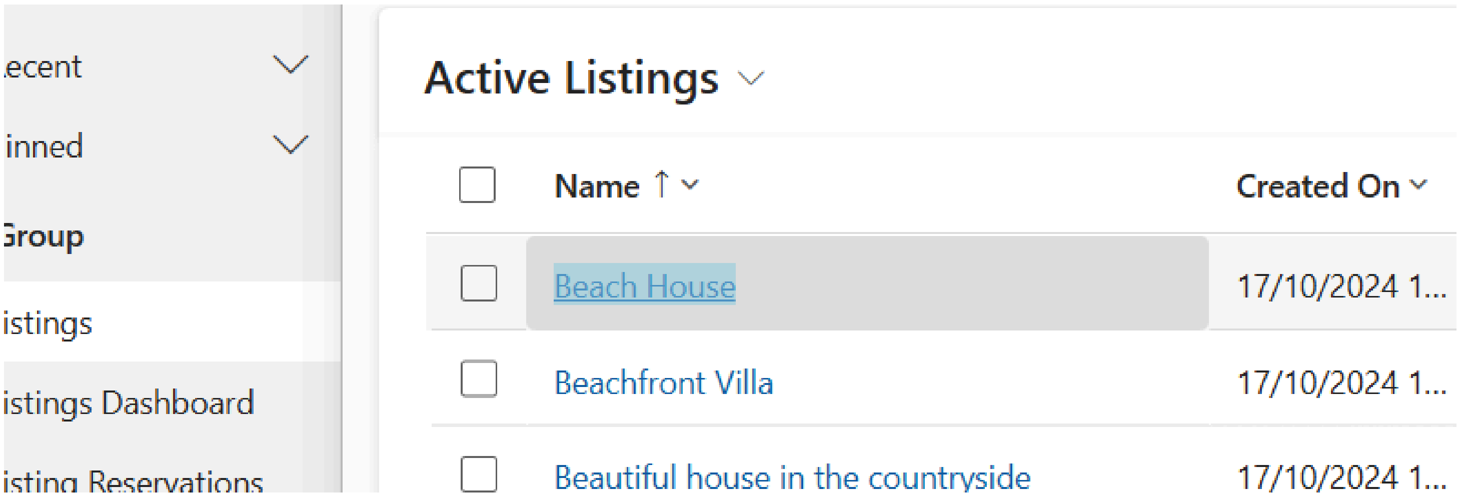
getByRole('link', { name: 'Beach House' })

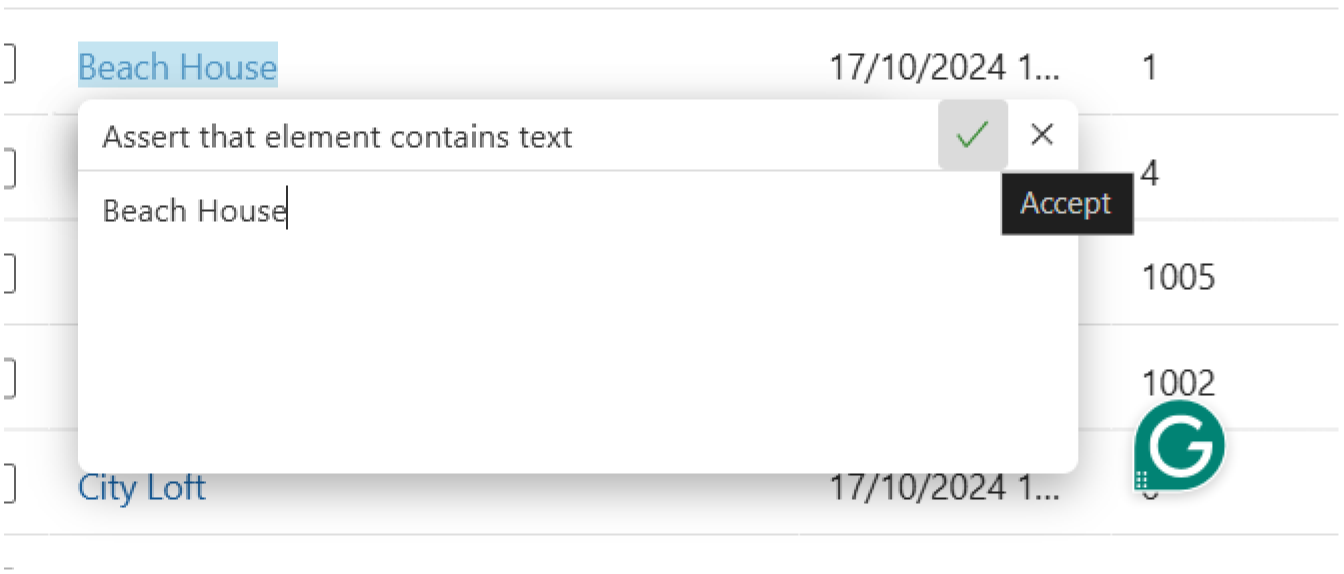
Name	Created On	Listing ID	Address	Price per Month
Beach House	17/10/2024 1...	1	123 Shore Dr	$3,500.
Beachfront Villa	17/10/2024 1...	4	123 Beach Rd	$5,000.
Beautiful house in the countryside	17/10/2024 1...	1005	67, Takofiv, CZ, ME	$86.
Charming apartment close to the train test	17/10/2024 1...	1002	73, Icidiof, KY, OR	$1,370.
City Loft	17/10/2024 1...	6	789 Broadway Ave	$2,000.
City View Apartment	17/10/2024 1...	17	456 High St	$1,800.

So this should mean we are in a listing record. Let us check we are getting the results we expect by asserting (Testing term for making sure) a few things.

Select the Assert Text button then the first listing (This should be Beach House if you have imported the data from the pre-requisites)


rgb8b744e5.crm.dynamics.com/main.aspx?appid=e15a8347-f958-4c20-b964-a8d7105f645f&pagetype=entitylist&
Contoso Real Estate Administration
New
Delete
Refresh
Visualize this view
Email a Link

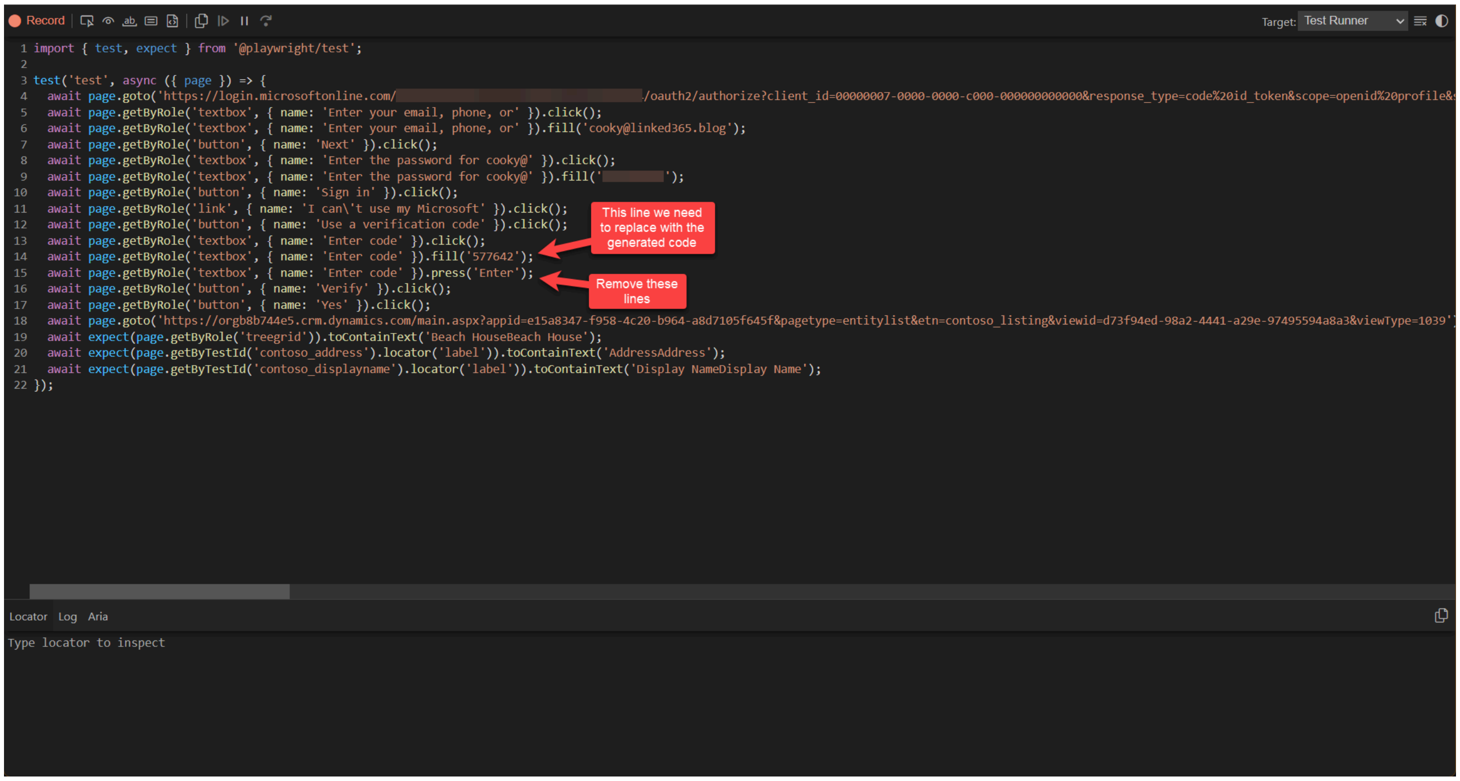




Ensure you select the Accept tick.

11. Repeat this action to assert the Address and Display Name columns are going to be there.
12. Take a look at the Playwright Inspector window, it should be very similar to mine below.

> {!TIP] If you see Enter options in your field entries, ensure you remove them. We have a happy of just pressing enter when entering passwords and other information, which works great when Playwright runs the test, but can be hit and miss when you are walking through the test scripts.



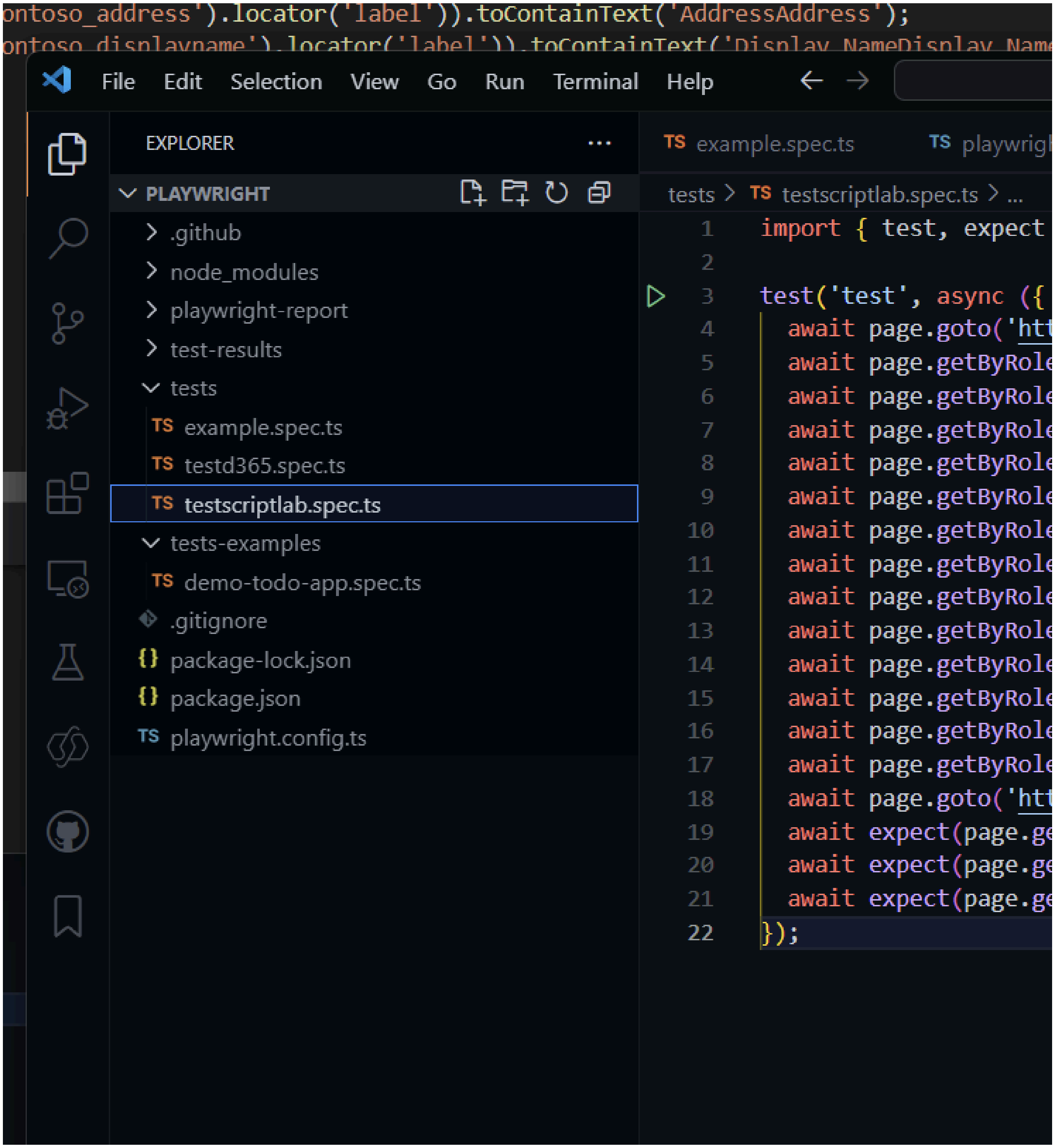
```
import { test, expect } from '@playwright/test';

test('test', async ({ page }) => {
  await page.goto('https://login.microsoftonline.com/[redacted]/oauth2/authorize?client_id=00000007-0000-0000-c000-000000000000&response_type=code%20id_token&scope=openid%20profile&
  await page.getByRole('textbox', { name: 'Enter your email, phone, or' }).click();
  await page.getByRole('textbox', { name: 'Enter your email, phone, or' }).fill('cooky@linked365.blog');
  await page.getByRole('button', { name: 'Next' }).click();
  await page.getByRole('textbox', { name: 'Enter the password for cooky@' }).click();
  await page.getByRole('textbox', { name: 'Enter the password for cooky@' }).fill('[redacted]');
  await page.getByRole('button', { name: 'Sign in' }).click();
  await page.getByRole('link', { name: 'I can\'t use my Microsoft' }).click();
  await page.getByRole('button', { name: 'Use a verification code' }).click();
  await page.getByRole('textbox', { name: 'Enter code' }).click();
  await page.getByRole('textbox', { name: 'Enter code' }).fill('577642');
  await page.getByRole('textbox', { name: 'Enter code' }).press('Enter');
  await page.getByRole('button', { name: 'Verify' }).click();
  await page.getByRole('button', { name: 'Yes' }).click();
  await page.goto('https://orgb8b744e5.crm.dynamics.com/main.aspx?appid=e15a8347-f958-4c20-b964-a8d7105f645f&pagetype=entitylist&etn=contoso_listing&viewid=d73f94ed-98a2-4441-a29e-97495594a8a3&viewType=1039'
  await expect(page.getByRole('treegrid')).toContainText('Beach HouseBeach House');
  await expect(page.getByTestId('contoso_address').locator('label')).toContainText('AddressAddress');
  await expect(page.getByTestId('contoso_displayname').locator('label')).toContainText('Display NameDisplay Name');
});
```

Automate login using a generated code

To automate this script, so it runs everytime, on every browser without interaction, we need to tweak the code entry to ensure it uses that which we generated.

1. Copy the code in the Playwright Inspector Window.
2. Create a new file in your VS Code Tests folder and call it testscriptlab.spec.ts

Paste your code into that box.



```
ontoso_address').locator('label')).toContainText('AddressAddress');
ontoso_displayname').locator('label')).toContainText('Display NameDisplay Name
```

```
import { test, expect

test('test', async ({
  await page.goto('htt
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.getByRole
  await page.goto('htt
  await expect(page.ge
  await expect(page.ge
  await expect(page.ge
});
```

At the top of the code, below the first line copy the following code

```
import * as OTPAuth from 'otpauth';

let totp = new OTPAuth.TOTP({
    issuer: "Microsoft",
    algorithm: "SHA1",
    digits: 6,
    period: 30,
    secret: "<your secret>",
  })
```

Ensure you replace your secret above with the secret key you generated in step 2.7 above.

This code is a small method that creates that 6 digit number for you. The code is a timestamp and secret combination, so if you know the secret, you can generate the code. This is used throughout the MFA world and is industry standard.

3. Find the line in your code where your script enters the code, it should look like

```
await page.getByRole('textbox', { name: 'Enter code' }).fill('577642');
```

Change this line to be

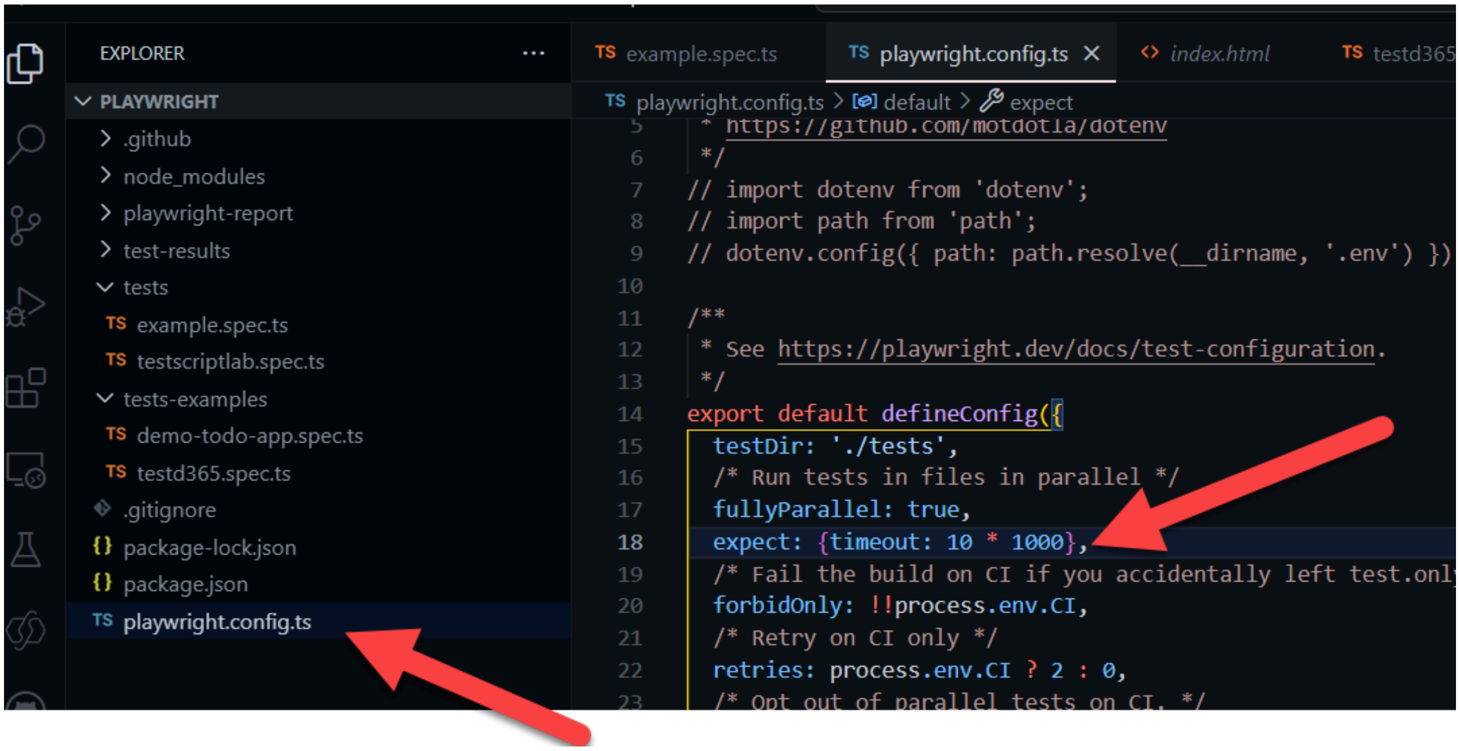
```
  await page.getByRole('textbox', { name: 'Enter code' }).fill(totp.generate());
```

This will use your secret and create a code and replicate you entering a code.

4. Finally, Playwright is not as forgiving on loading as Power Platform is, so we need to just tweak the time outs for the expect action (when something is asserted or expected to be there to twice the normal)

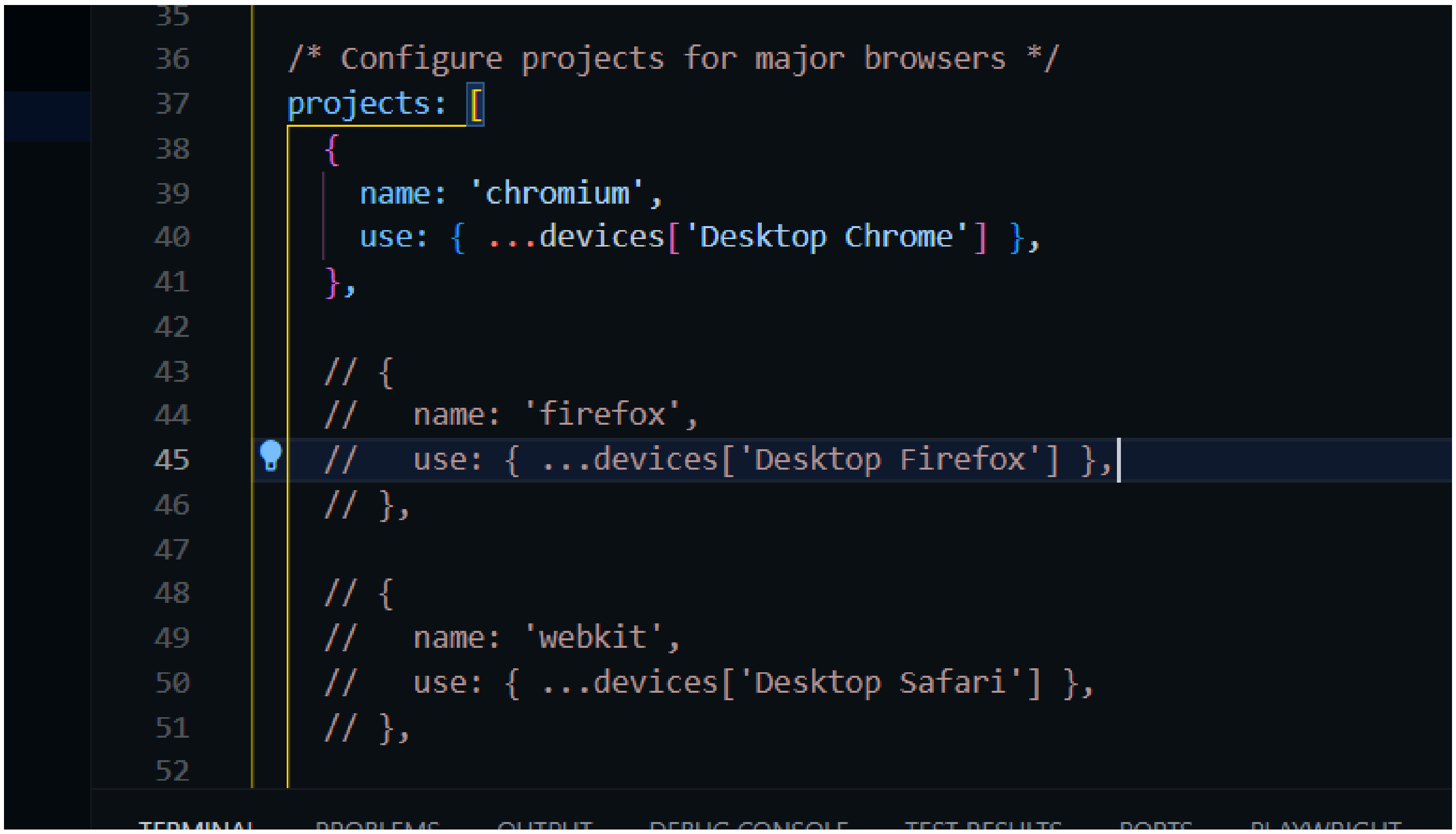
Find the playwright.config.ts file in your top level directory, and add the line below the fullyParrallel line

```
expect: {timeout: 10 * 1000},
```



```
 * https://github.com/motdotla/dotenv
 */
// import dotenv from 'dotenv';
// import path from 'path';
// dotenv.config({ path: path.resolve(__dirname, '.env') })

/**
 * See https://playwright.dev/docs/test-configuration.
 */
export default defineConfig({
  testDir: './tests',
  /* Run tests in files in parallel */
  fullyParallel: true,
  expect: {timeout: 10 * 1000},
  /* Fail the build on CI if you accidentally left test.onl
  forbidOnly: !!process.env.CI,
  /* Retry on CI only */
  retries: process.env.CI ? 2 : 0,
  /* Opt out of parallel tests on CI. */
```

> {!TIP] If you are struggling with getting your tests running, disabling one or more of the projects (the browser bases that your tests run on) a little further down by commenting out the Safari entry for example



```
  /* Configure projects for major browsers */
  projects: [
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    },

    // {
    //   name: 'firefox',
    //   use: { ...devices['Desktop Firefox'] },
    // },

    // {
    //   name: 'webkit',
    //   use: { ...devices['Desktop Safari'] },
    // },
```

5. Make sure all your files are saved (Ctrl + K, S) and re-run your test by running in the terminal the test command

```
npx playwright run
```

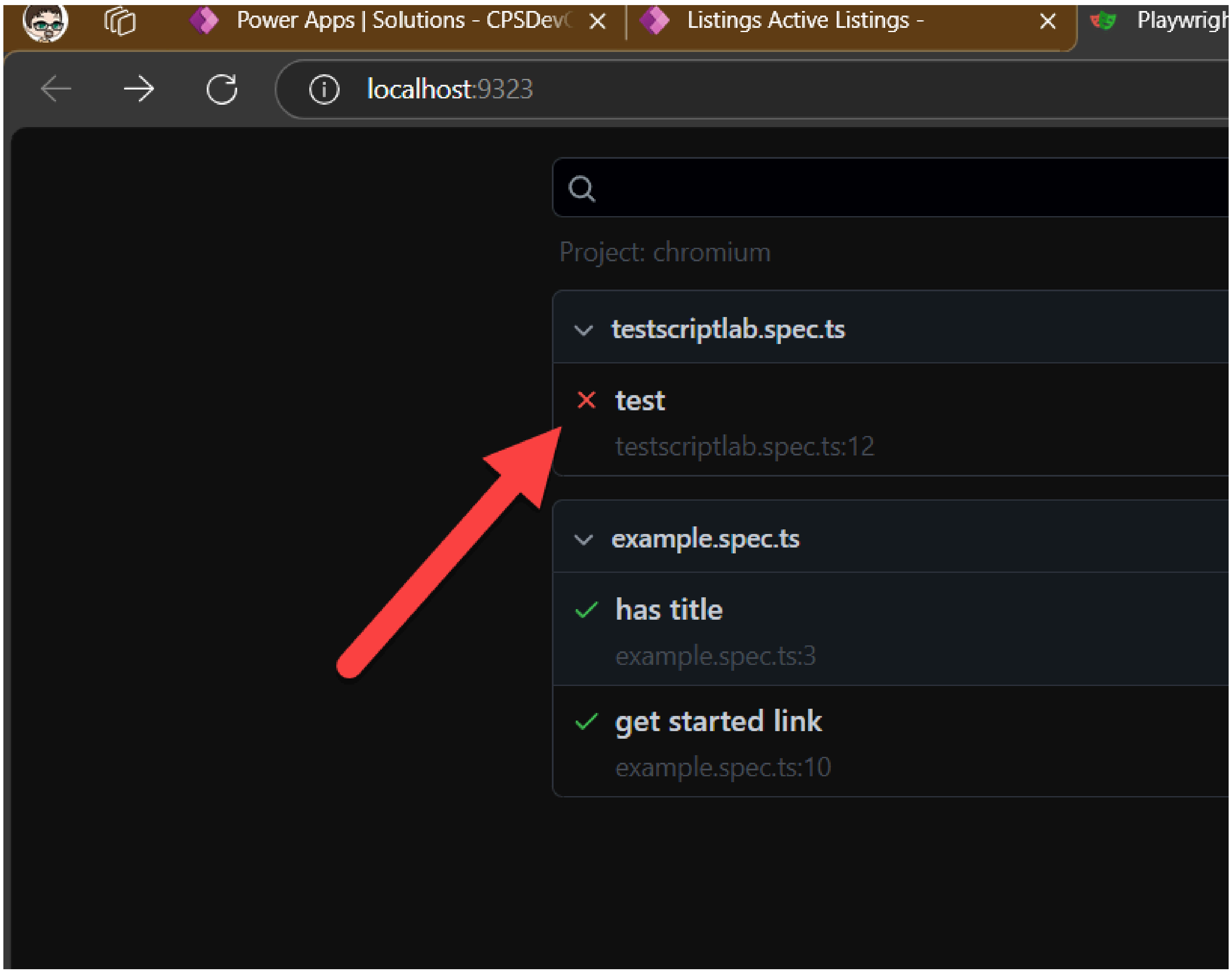
Hopefully everything works and you get a positive result!

Change your solution and run the script

As you are testing to ensure a solution always contains certain things, in our instance, we have a view with 2 columns, Address and Display Name, To make the test fail, edit the view and retest.

1. go and tweak the Active Listings View to remove the Address column.
2. Save and Publish the view.
3. Check in your application the address view is no longer present.
4. Re-run the test in Playwright

You should get a failure and the report is automatically displayed



Project: chromium

testscriptlab.spec.ts

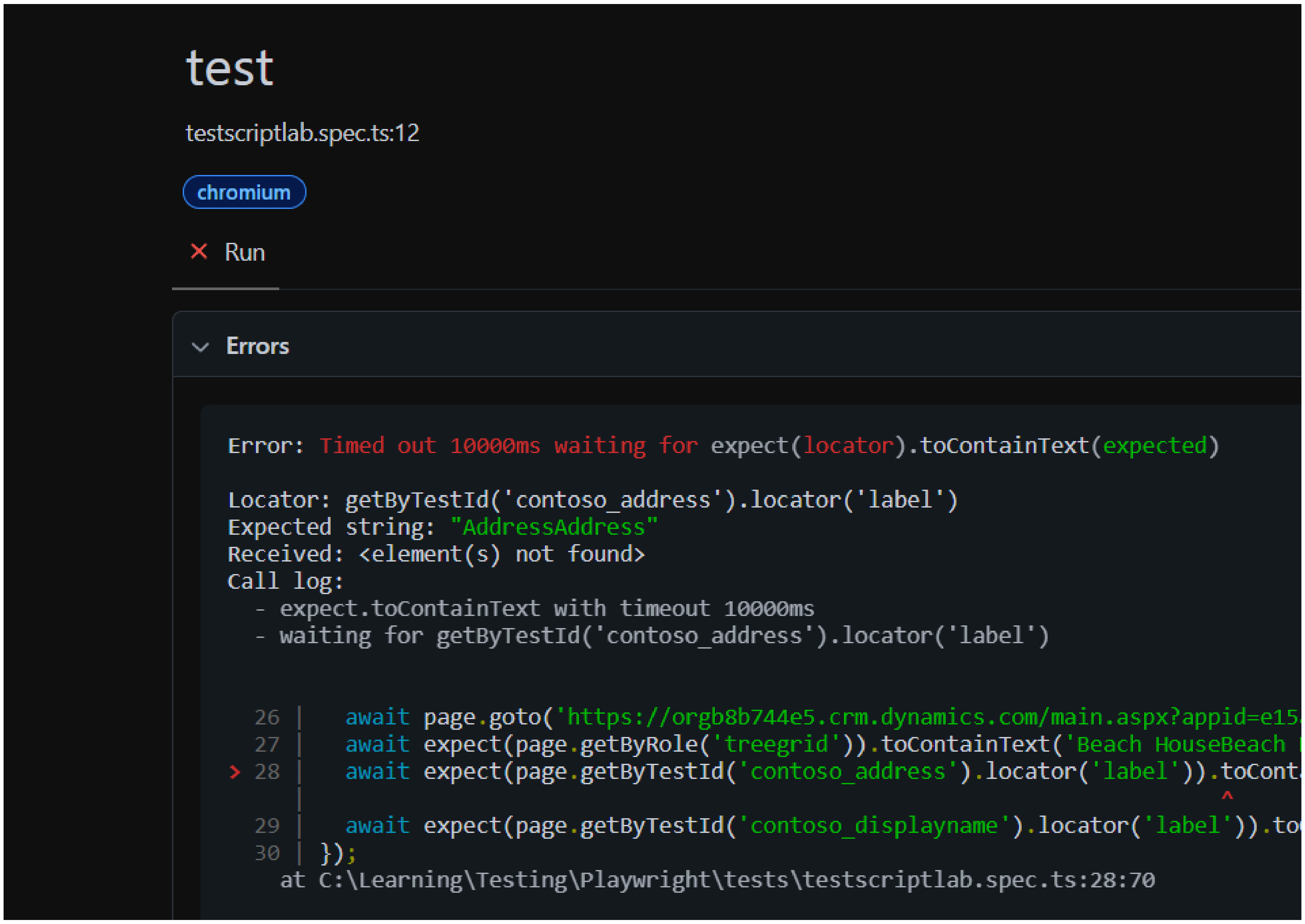
✗ test
testscriptlab.spec.ts:12

example.spec.ts

✓ has title
example.spec.ts:3

✓ get started link
example.spec.ts:10

Click on the Red item to see what failed, in our case, the Address field is not present.



test

testscriptlab.spec.ts:12

chromium

✕ Run

Errors

```
Error: Timed out 10000ms waiting for expect(locator).toContainText(expected)

Locator: getByTestId('contoso_address').locator('label')
Expected string: "AddressAddress"
Received: <element(s) not found>
Call log:
  - expect.toContainText with timeout 10000ms
  - waiting for getByTestId('contoso_address').locator('label')


    |   await page.goto('https://orgb8b744e5.crm.dynamics.com/main.aspx?appid=e15
    |   await expect(page.getByRole('treegrid')).toContainText('Beach HouseBeach
  > |   await expect(page.getByTestId('contoso_address').locator('label')).toCont
    |                                                                      ^
    |   await expect(page.getByTestId('contoso_displayname').locator('label')).to
    | });
    at C:\Learning\Testing\Playwright\tests\testscriptlab.spec.ts:28:70
```

5. Add the address field back and re-run your test to prove you have now fixed your test.

Configuring DevOps

The ideal scenario is for when a dev pushes their code into a test environment, a suite of tests are run against it, built up from what the testers and users deemed as being important. This suite will build over time to include aspects that are not changed over time but may be inadvertently changed (fields in view, fields on forms, calculations etc etc.)

Let us know look how to add our Playwright test script to ADO.