*Official website:* [*https://www.cypress.io/*](https://www.cypress.io/)

[*https://docs.cypress.io/guides/overview/why-cypress*](https://docs.cypress.io/guides/overview/why-cypress)

*Note: cypress only supports `css selectors`*

*Syntax: #id, .class*

*To write css for parent to child – use space*

Cypress Step-by-Step Installation:

What is Node.js?

Node.js is an open-source, cross-platform, backend JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

Pre-requisites:

1. Download Node.js

2. Download Visual Studio Code

Step1: Create a new project with “Package.json”

1. A “package.json” is a JSON file at the root of a JavaScript/Node project.
2. It holds metadata relevant to the project and is used for managing project dependencies.
3. From <root\_folder> where package.json is present from there hit "npm install" to install all dependencies

Note:

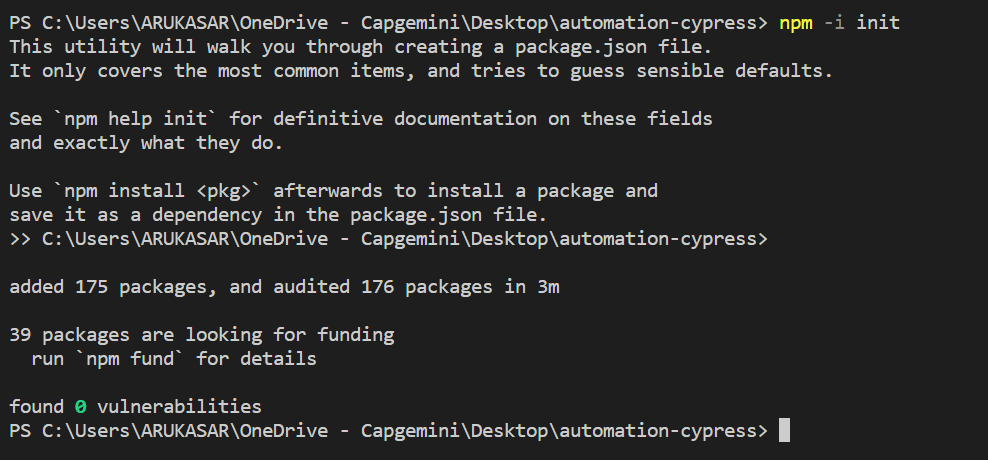
* Make a New Directory -- mkdir automation-cypress.
* Change Directory -- cd..
* Create a “package.json” file using the "npm -i init" command. And do not forget to hit enter.
* To update npm use the command "npm install -g npm"

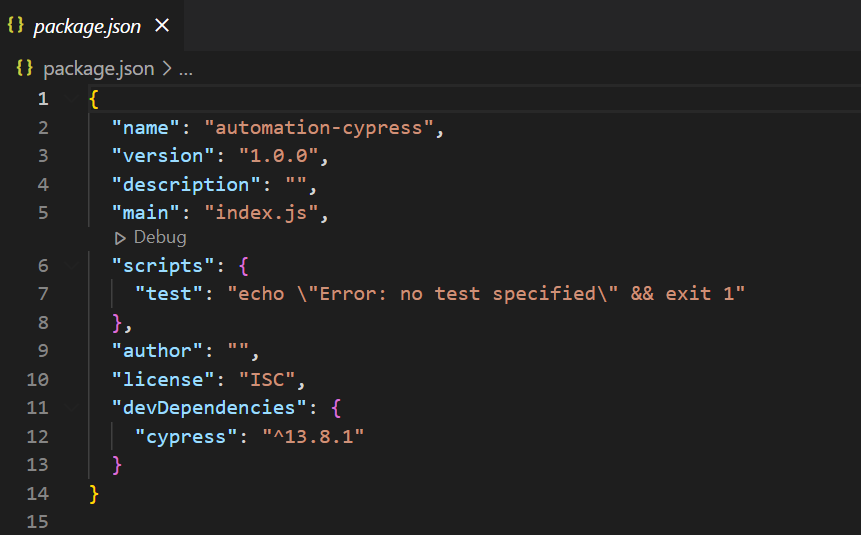
Install Cypress:

Installing: "npm install"

Install Cypress via npm [if cypress version is given in package.json]: “npm install cypress”

To save or create the entry in package.json use the command "npm install <package name> --save-dev": npm install cypress --save-dev

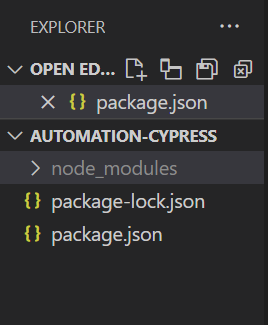




After Step1

Step2: “package-lock.json” will create automatically

Step3: The “node\_modules” folder also will be created automatically.

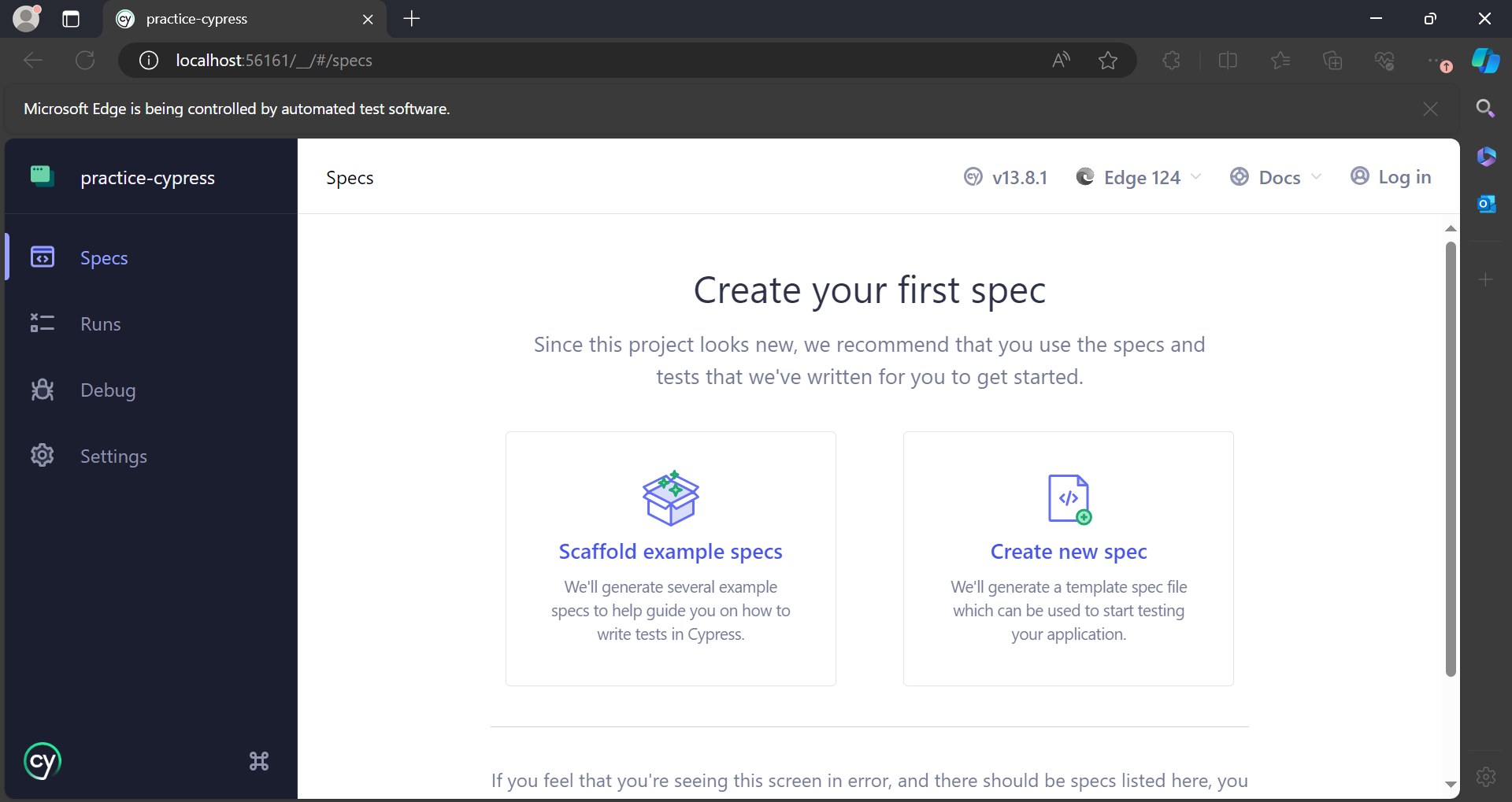


After Step2 and 3

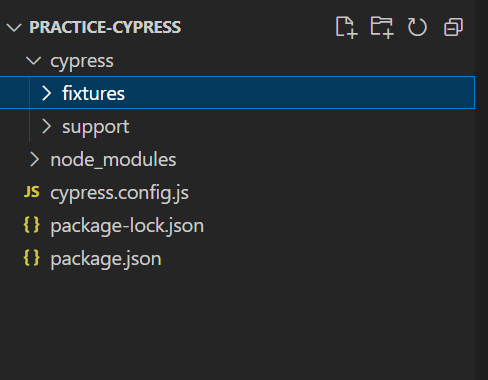
Step4: To run cypress use the command “npx cypress open” or refer: <https://docs.cypress.io/guides/getting-started/opening-the-app>



* *Click on E2E Testing and then continue.*



* Cypress folder is created automatically:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Write our first test case:

Write test cases inside cypress folder which was created automatically.

Save the test cases file using `.js` extension

Now it's time to write your first test. We're going to:

* Write your first passing test.
* Update it so it fails.
* Watch Cypress reload in real time.

/\*Describe take two arguments

1. About description

2. Function that wraps all its blocks

it

1. description

2. function

For example:

describe('My First Test', () => {

it('My first test case', () => {

//test step

})

it('My second test case', () =>{

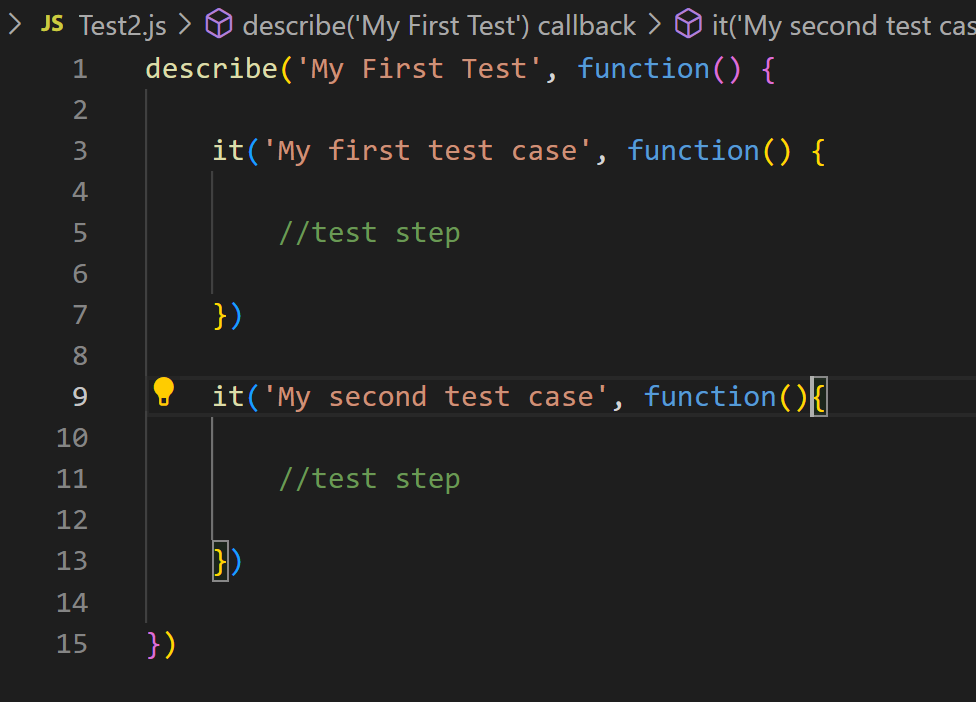
//test step

})

})

\*/

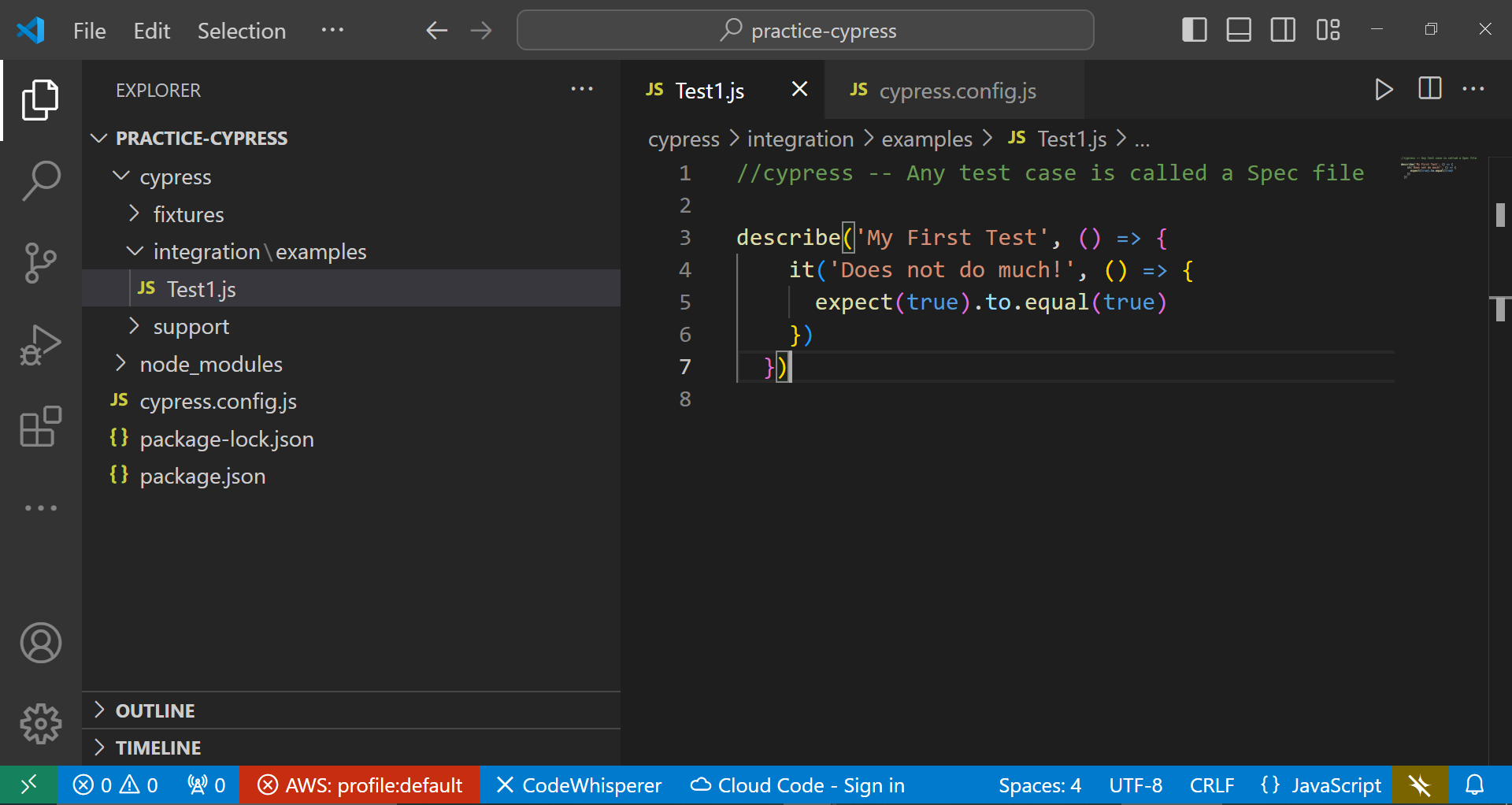
Syntax:



Open your favorite IDE and replace the contents of your spec with the code below.

//cypress -- Any test case is called a Spec file

In this dummy test case: Although it doesn't do anything useful, this is our first passing test! ✅

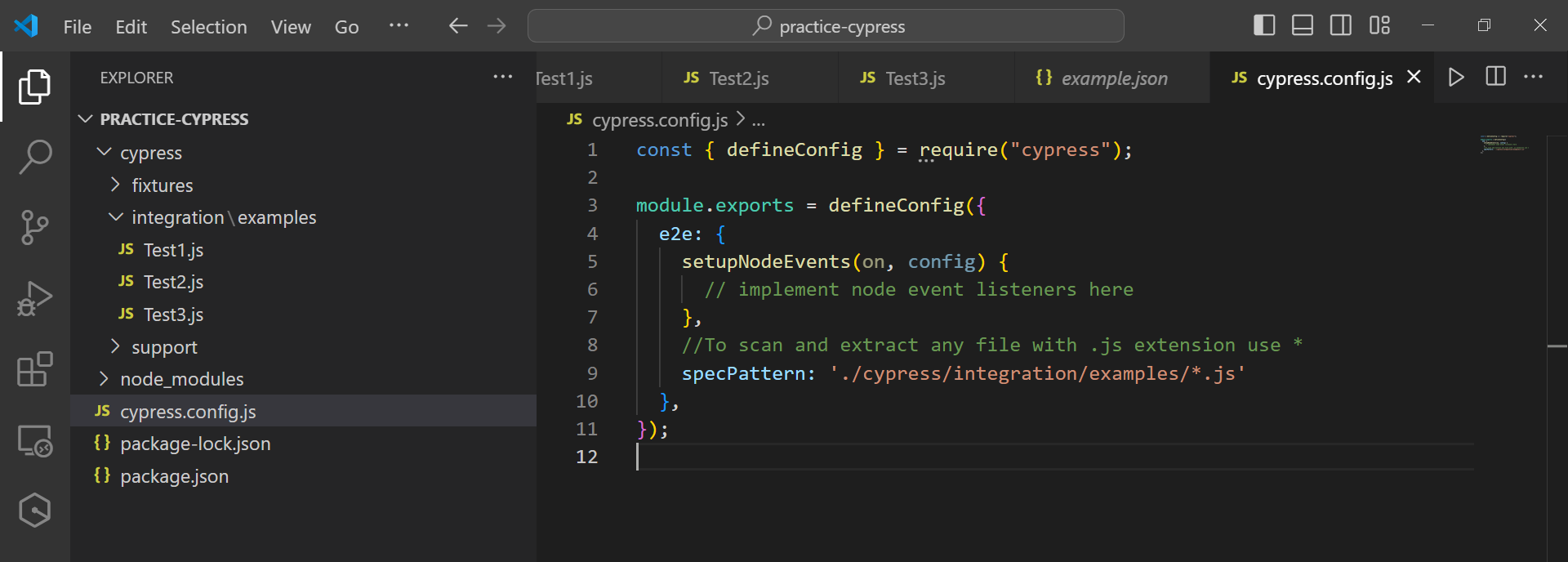


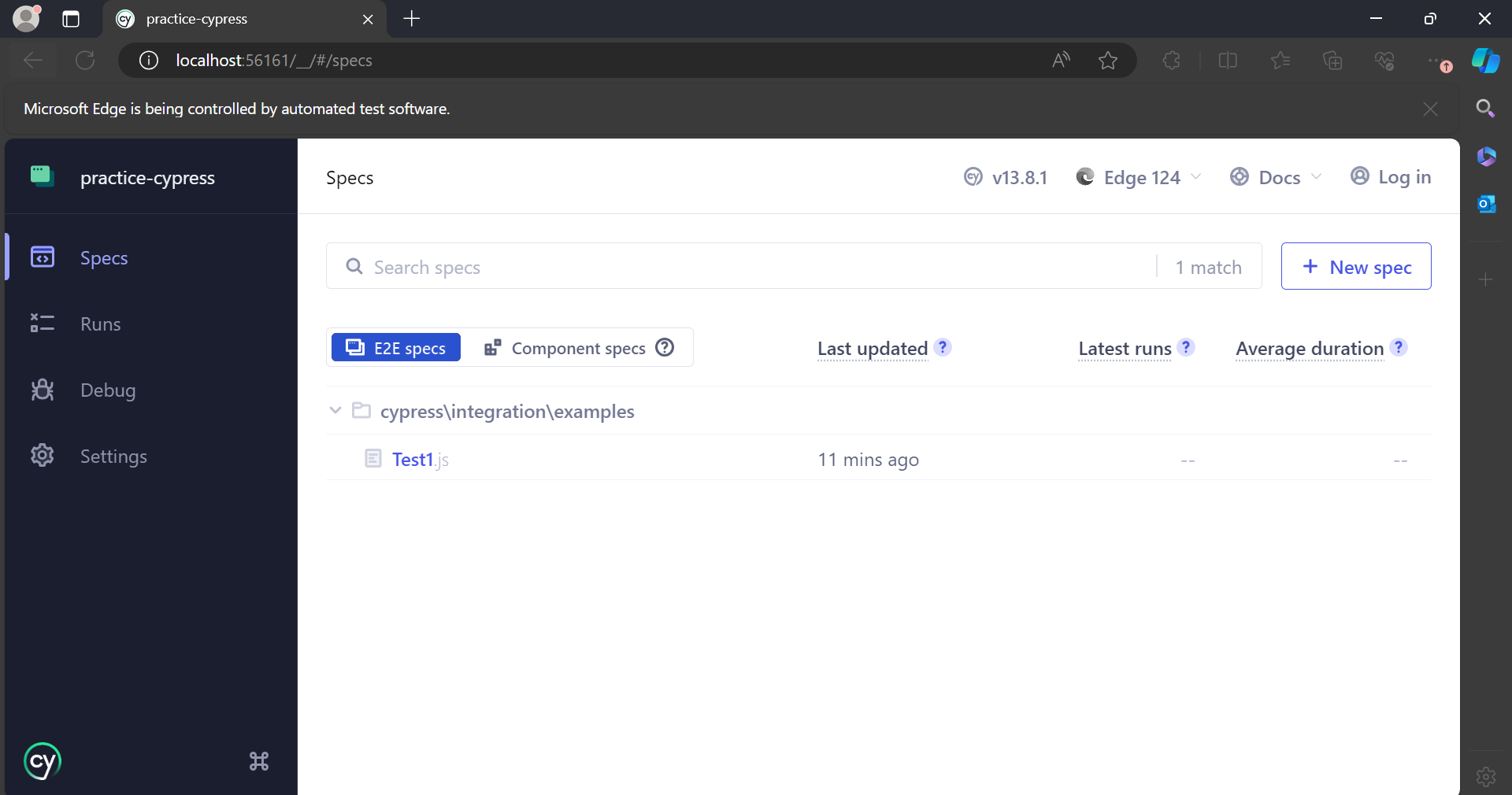
To tell the test runner where your test cases have been written we need to give path in “config.json file”

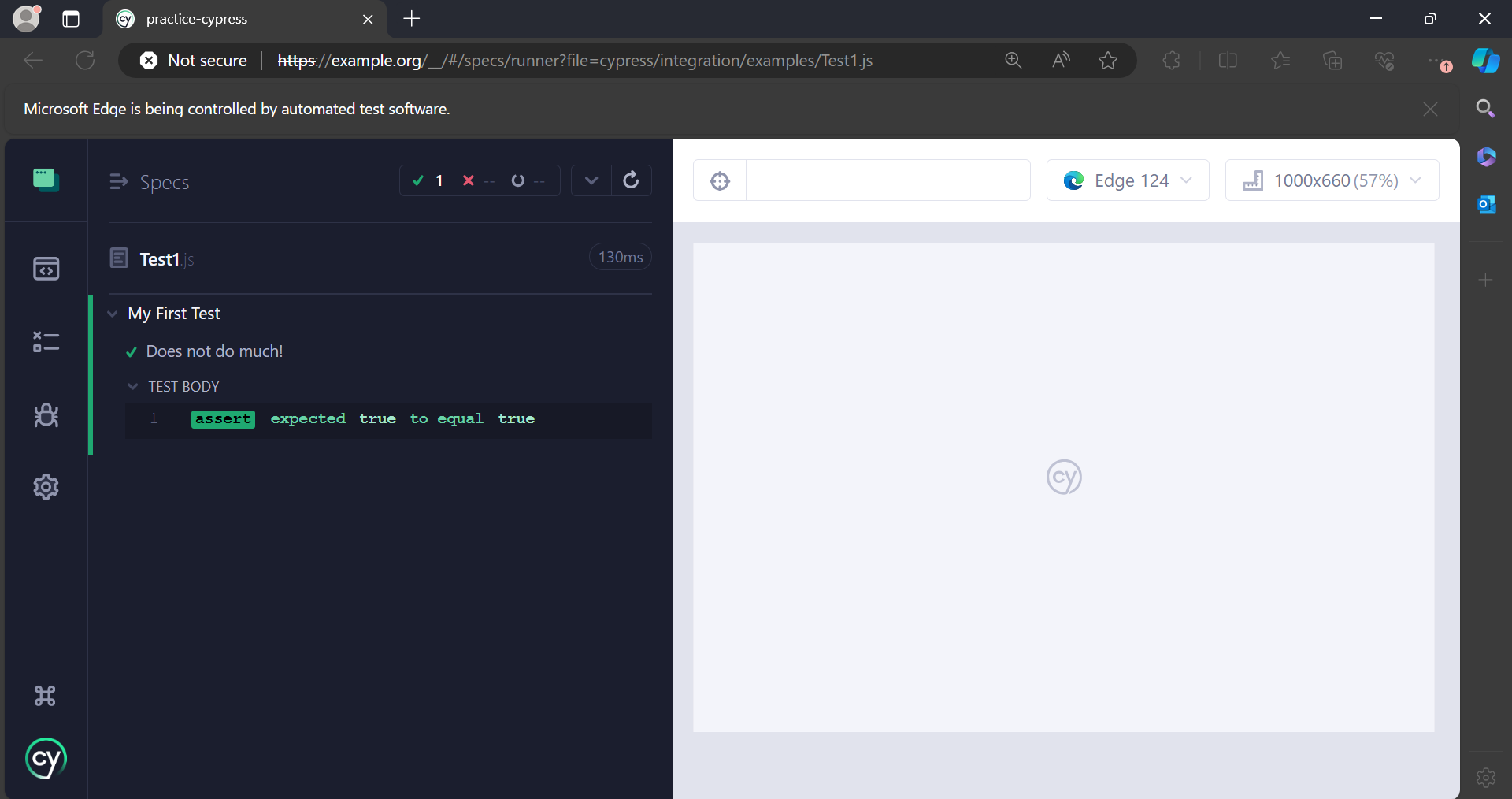
Add new field: //To scan and extract any file with .js extension use \*

specPattern: './cypress/integration/examples/\*.js

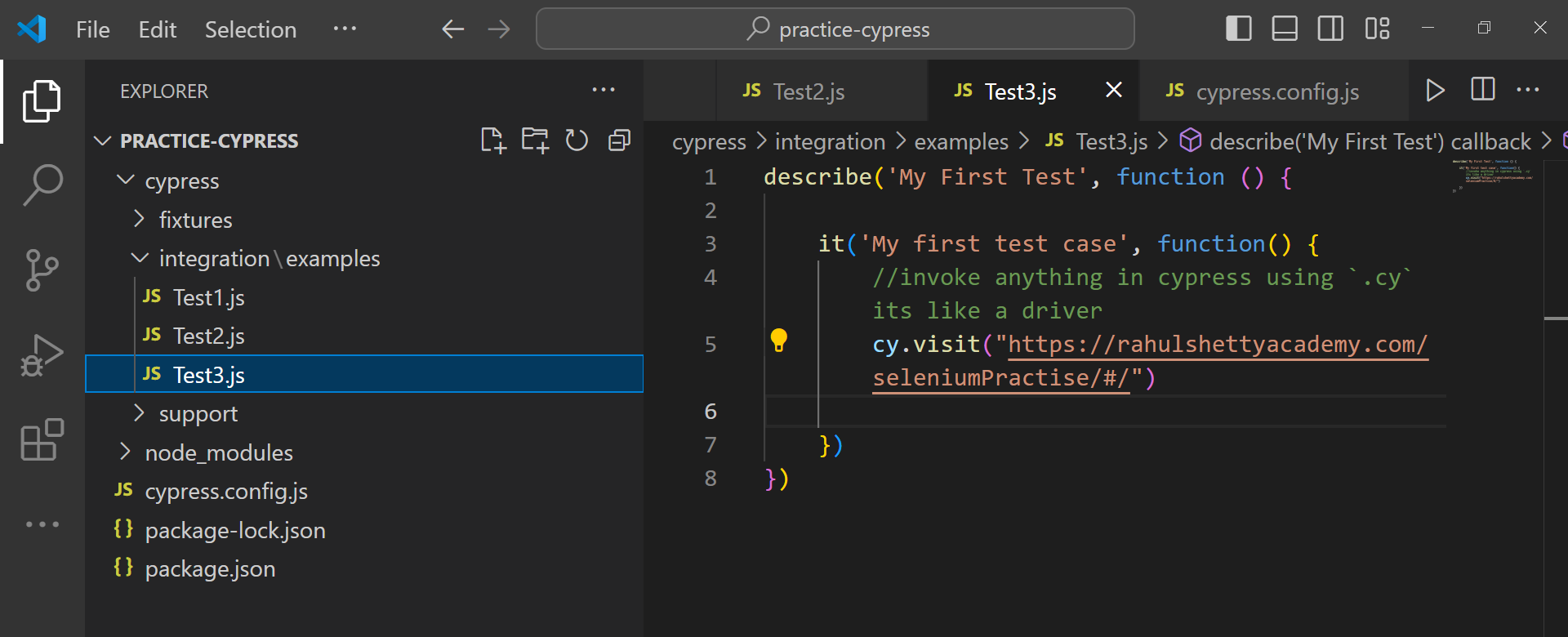
'

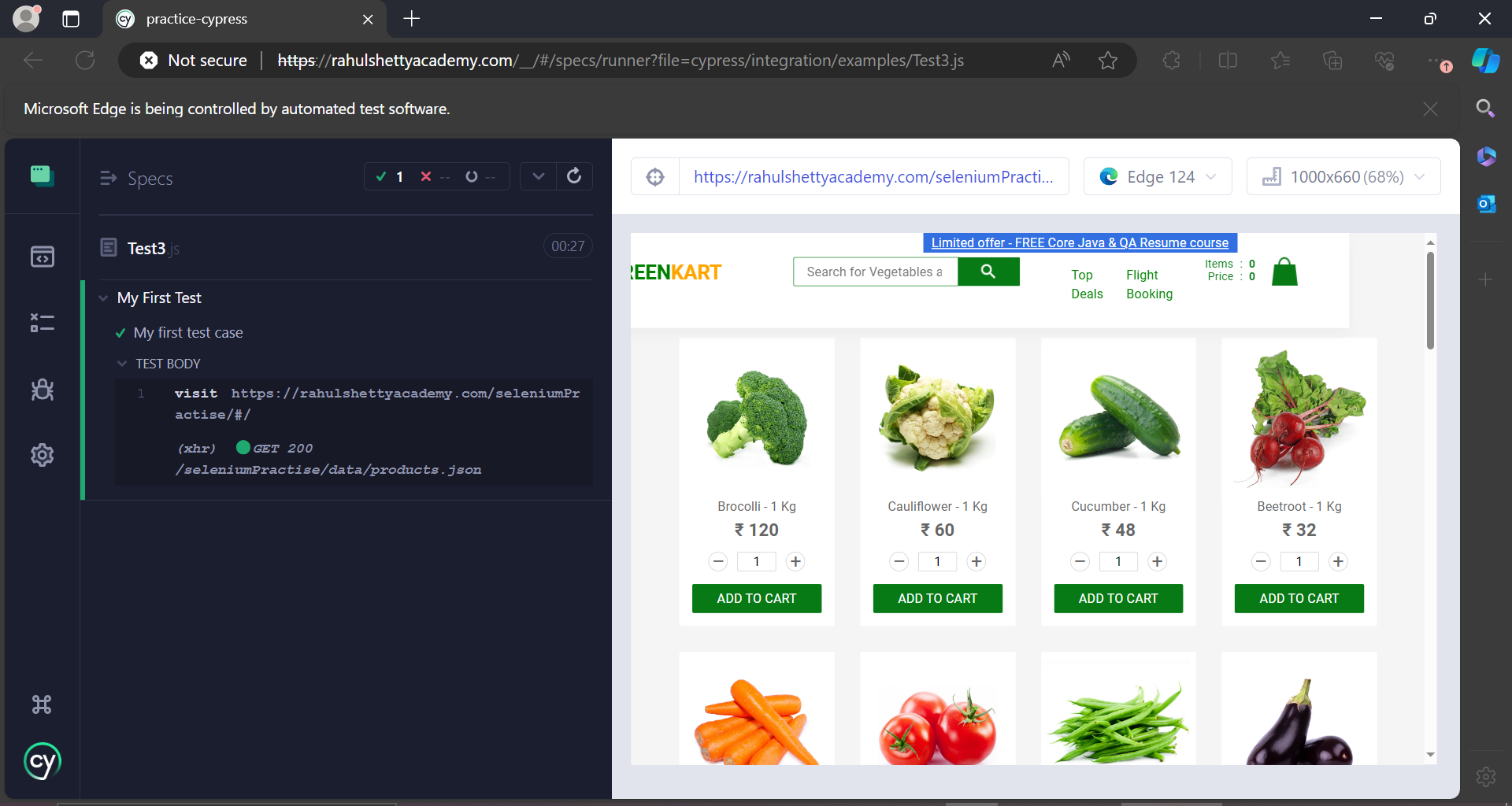


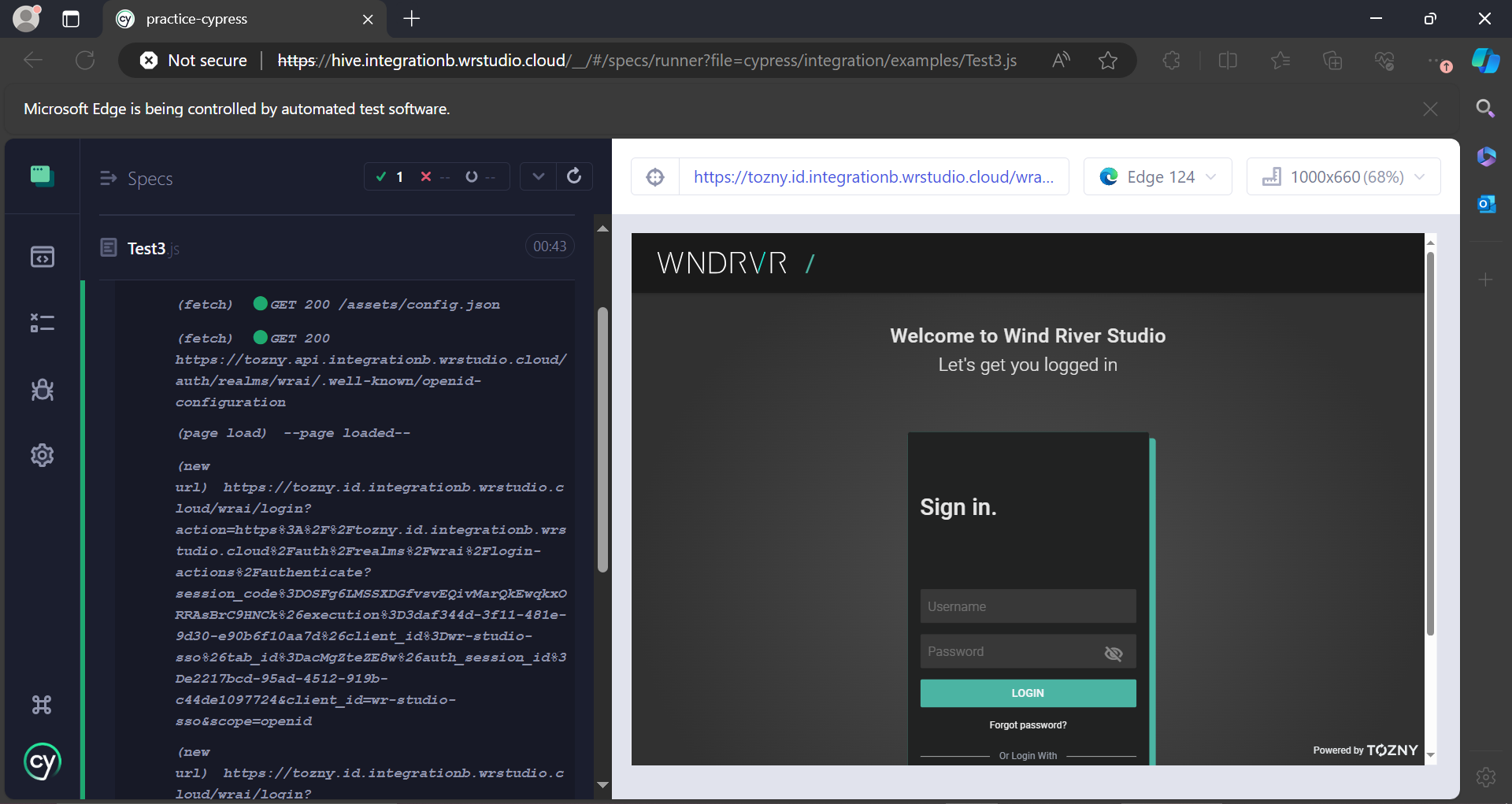




First test case:



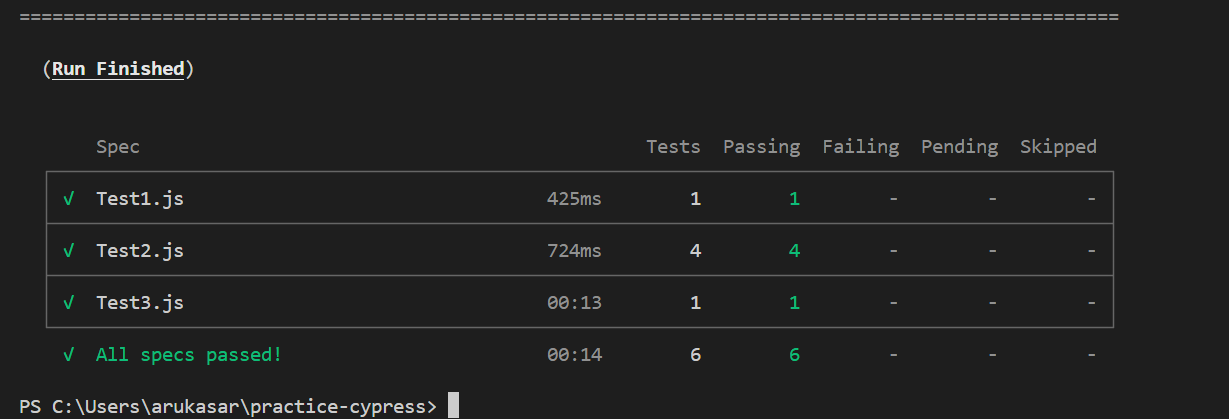
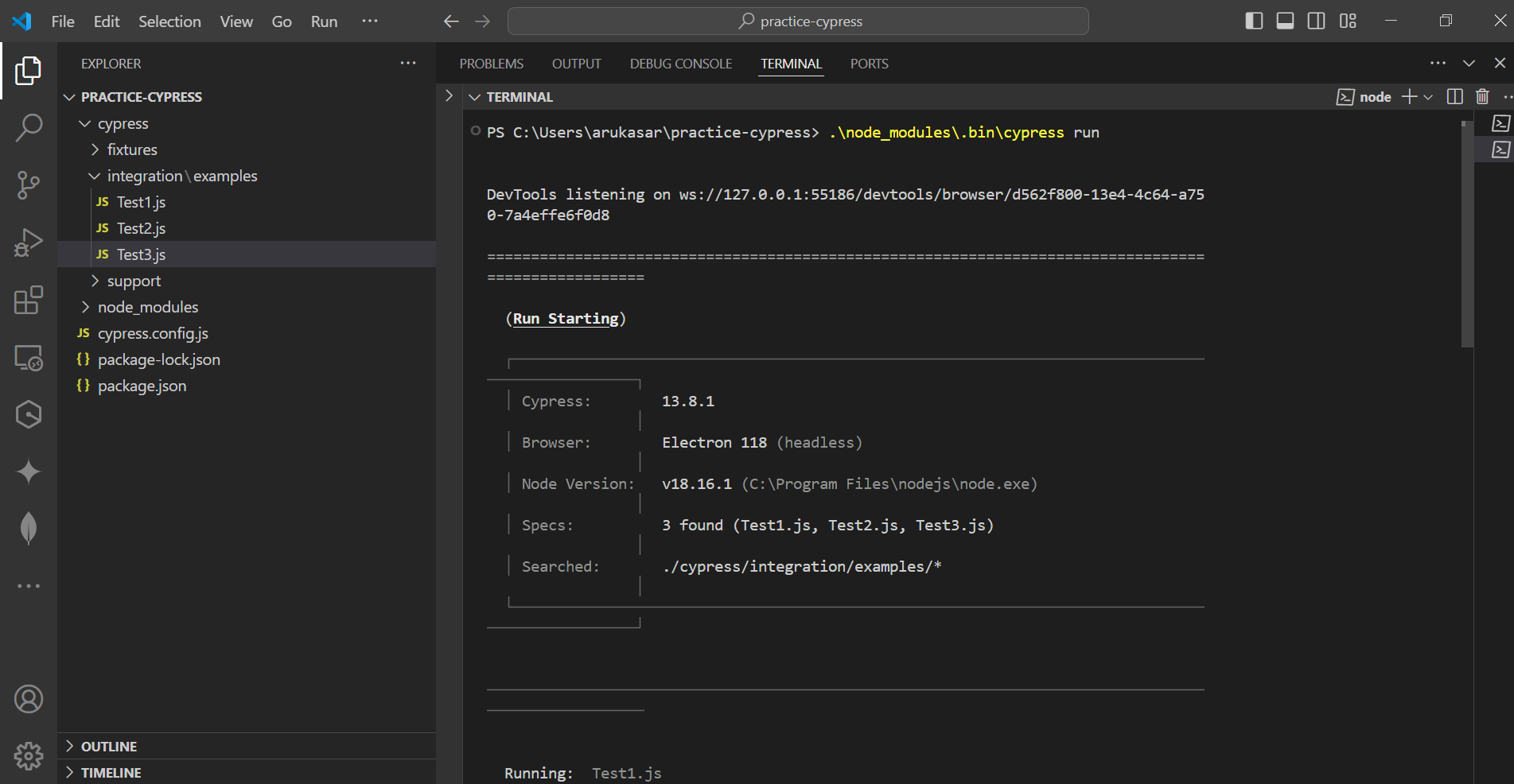


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How to run tests from cmd?

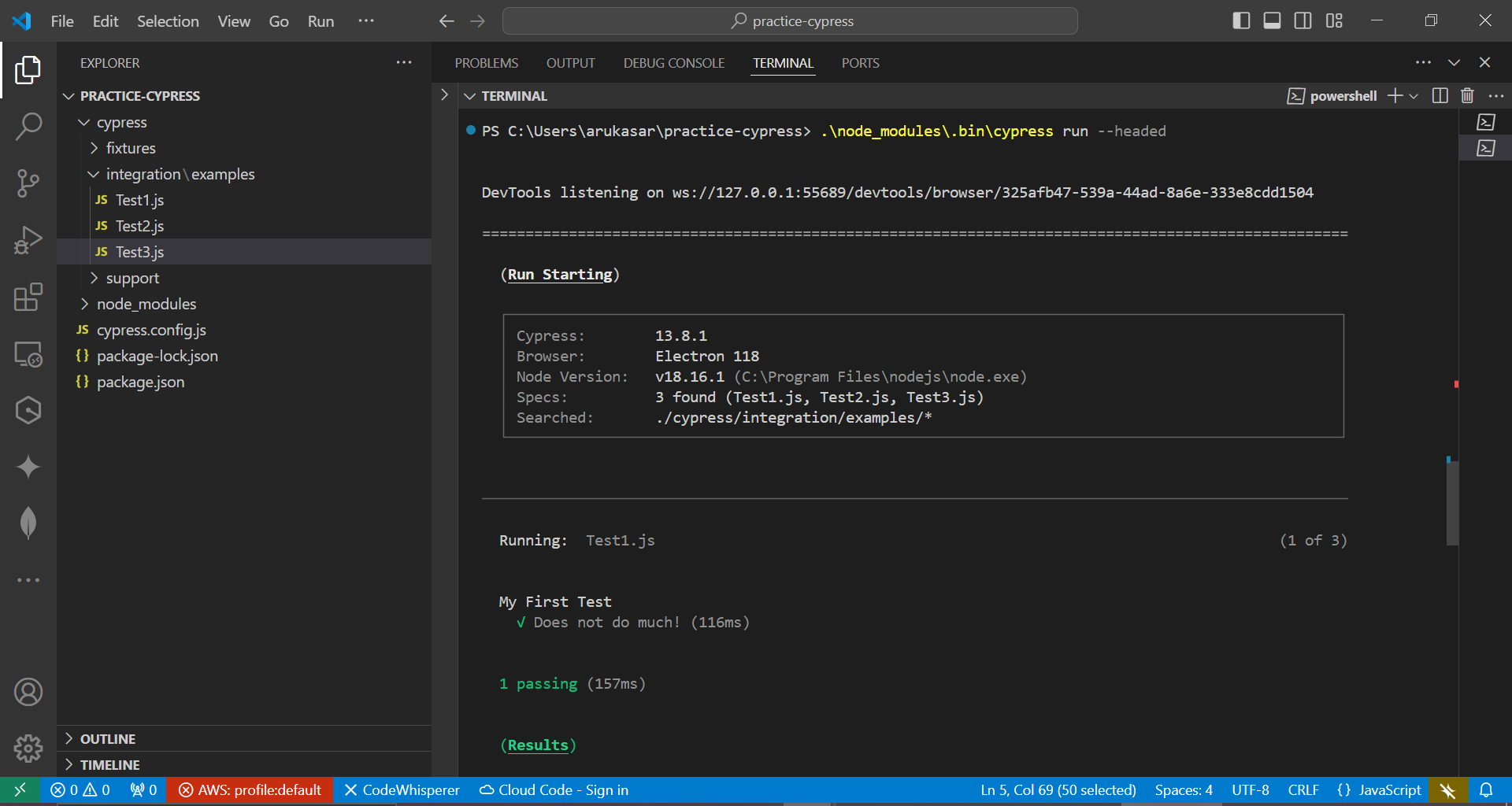
.\node\_modules\.bin\cypress run

It will execute all the test cases without opening any browser automatically.



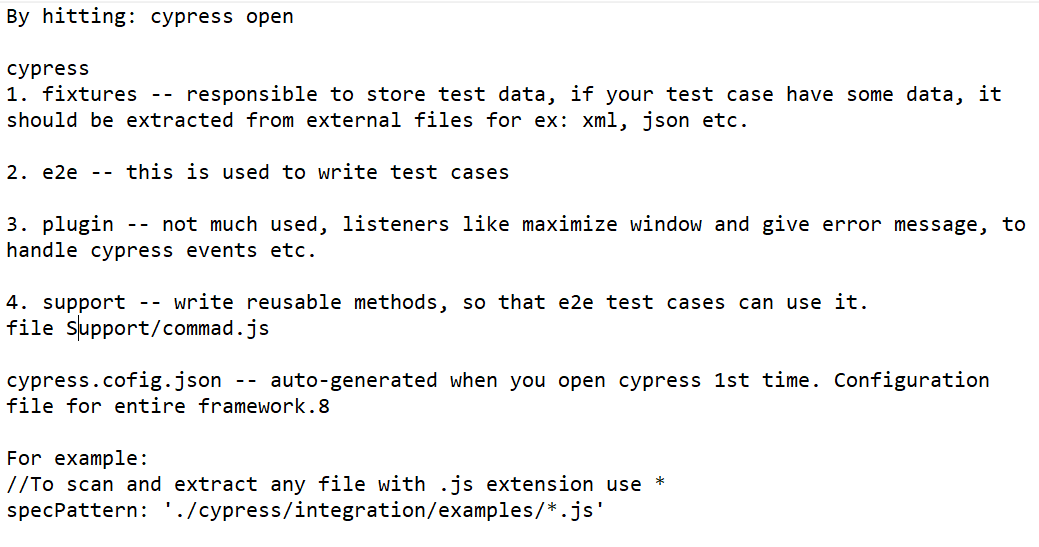
.\node\_modules\.bin\cypress run –headed

It will automatically invoke electron browser and run all the test cases after completion it will close itself.



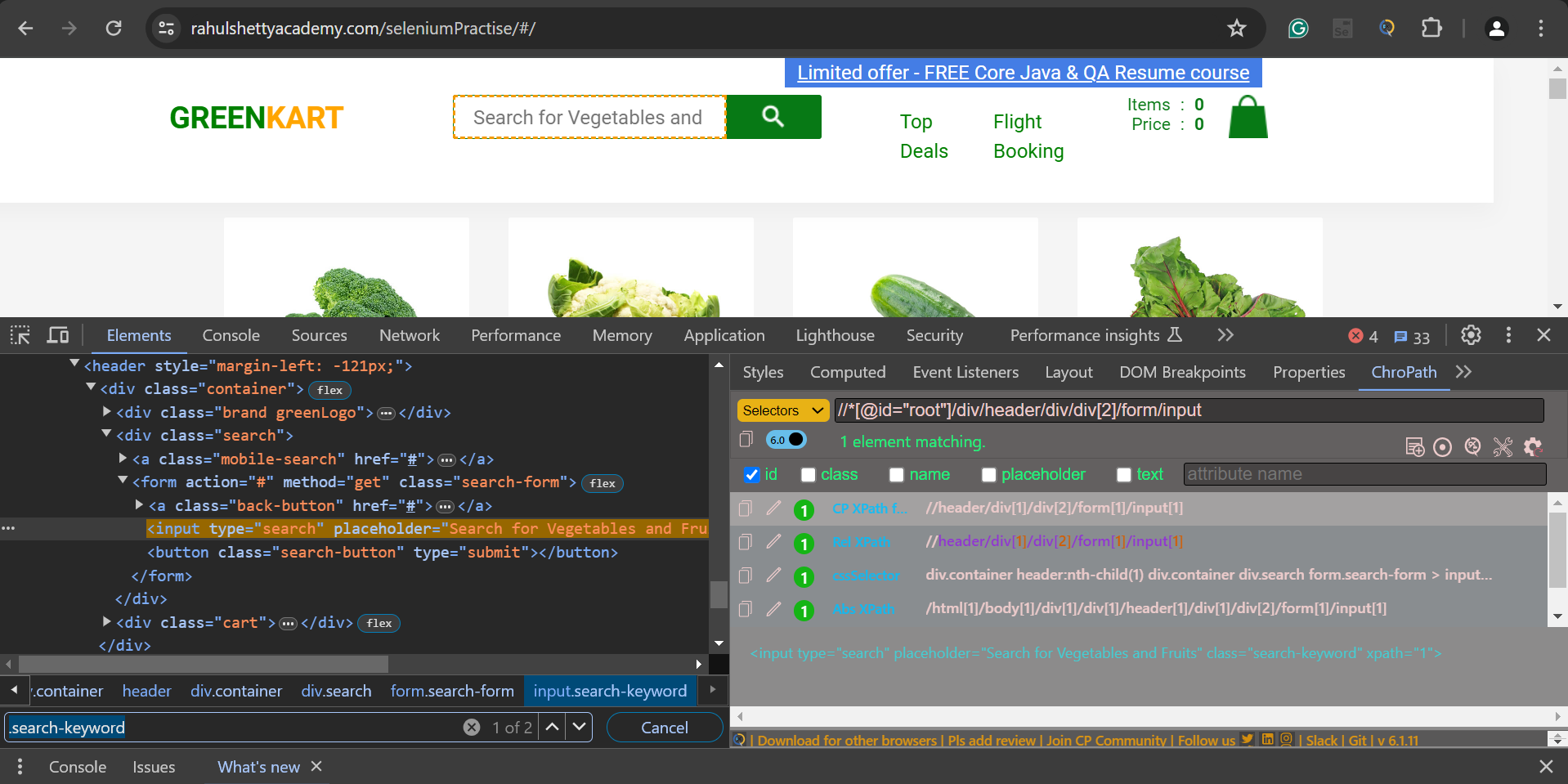
.\node\_modules\.bin\cypress run --browser <name>

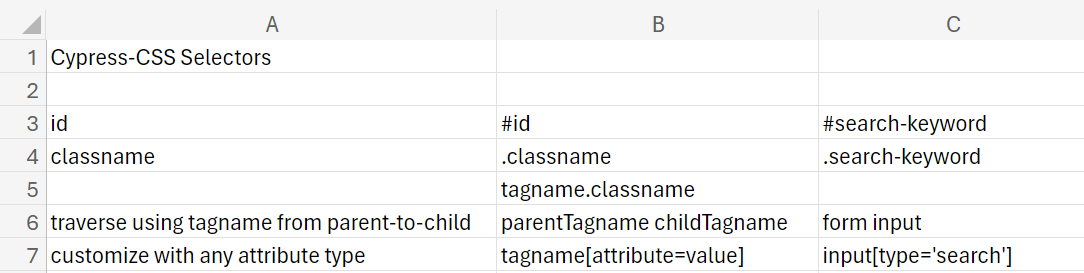
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Framework:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Locators [CSS Selector only]

Extension--chropath



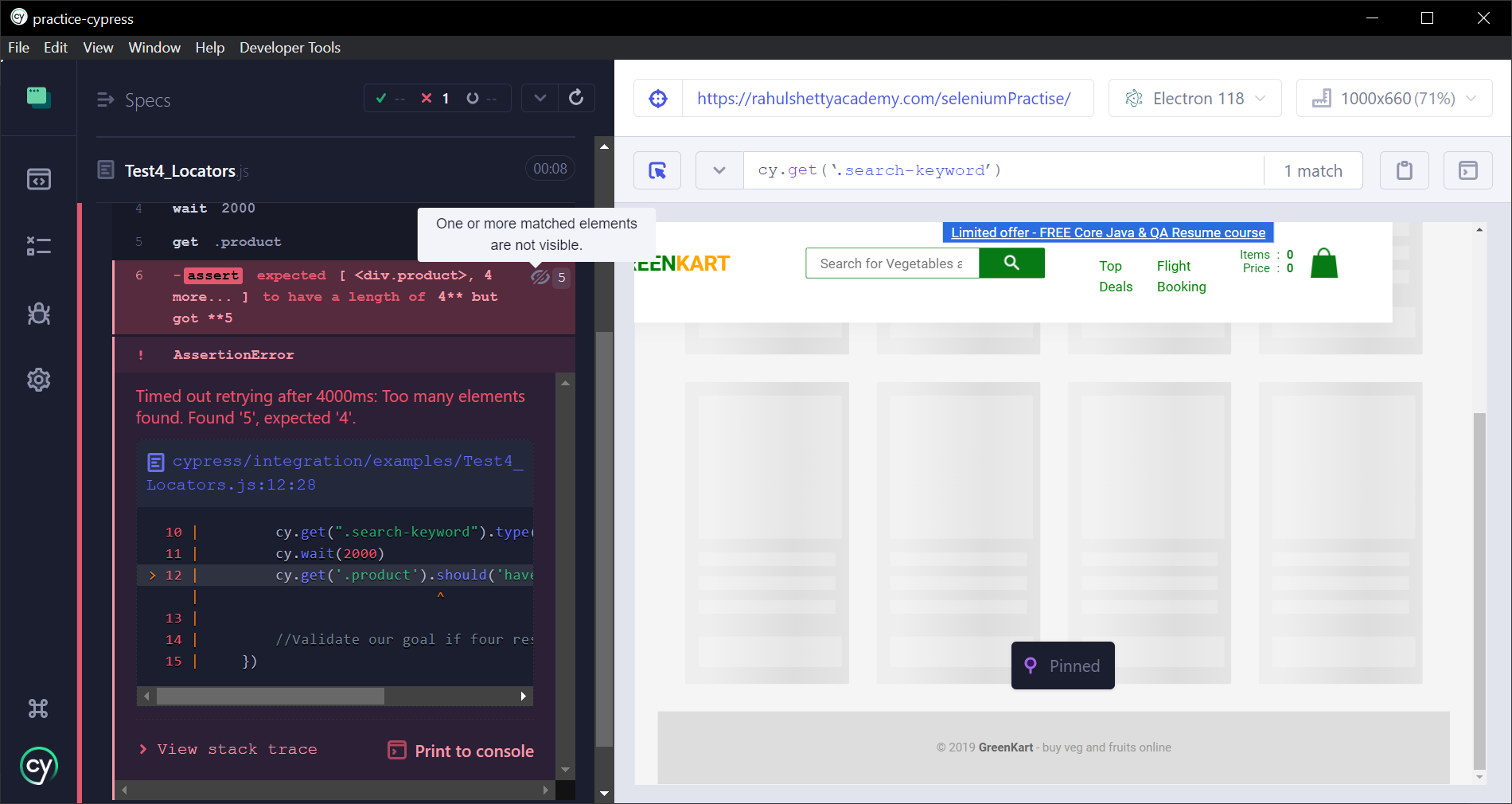
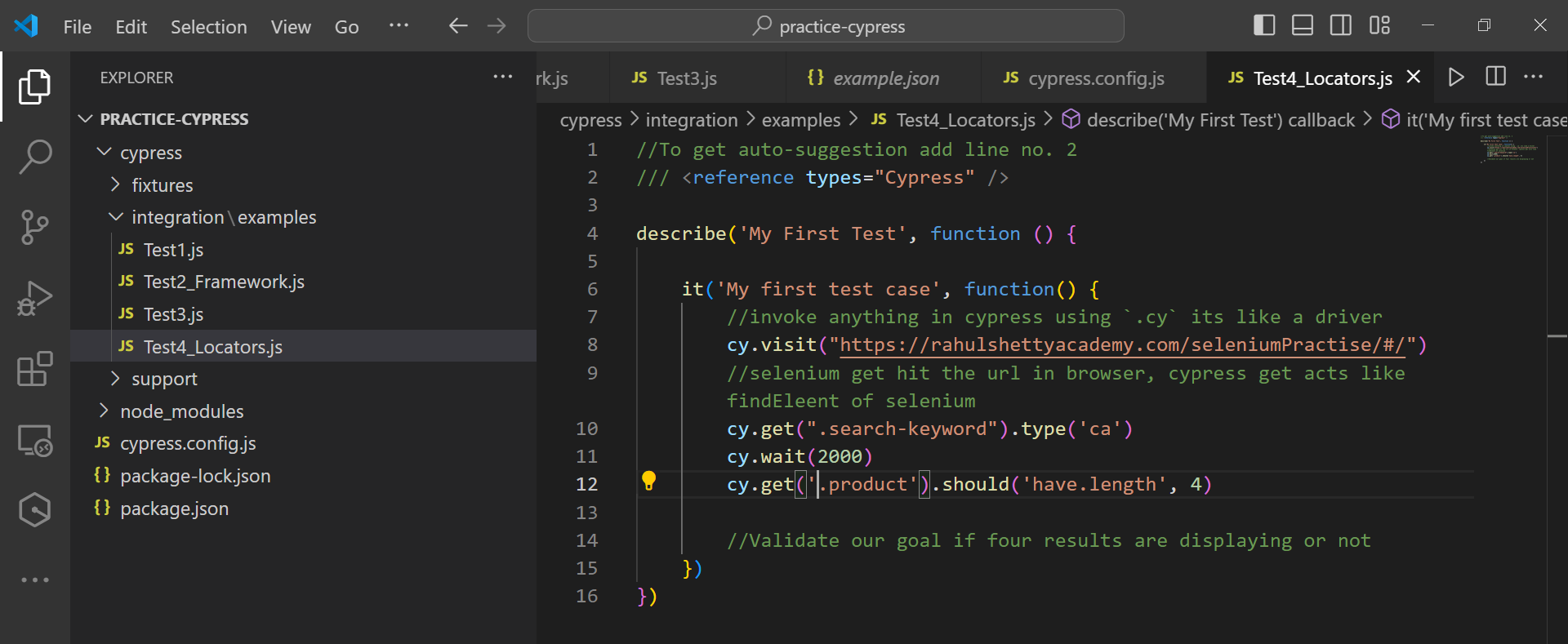
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

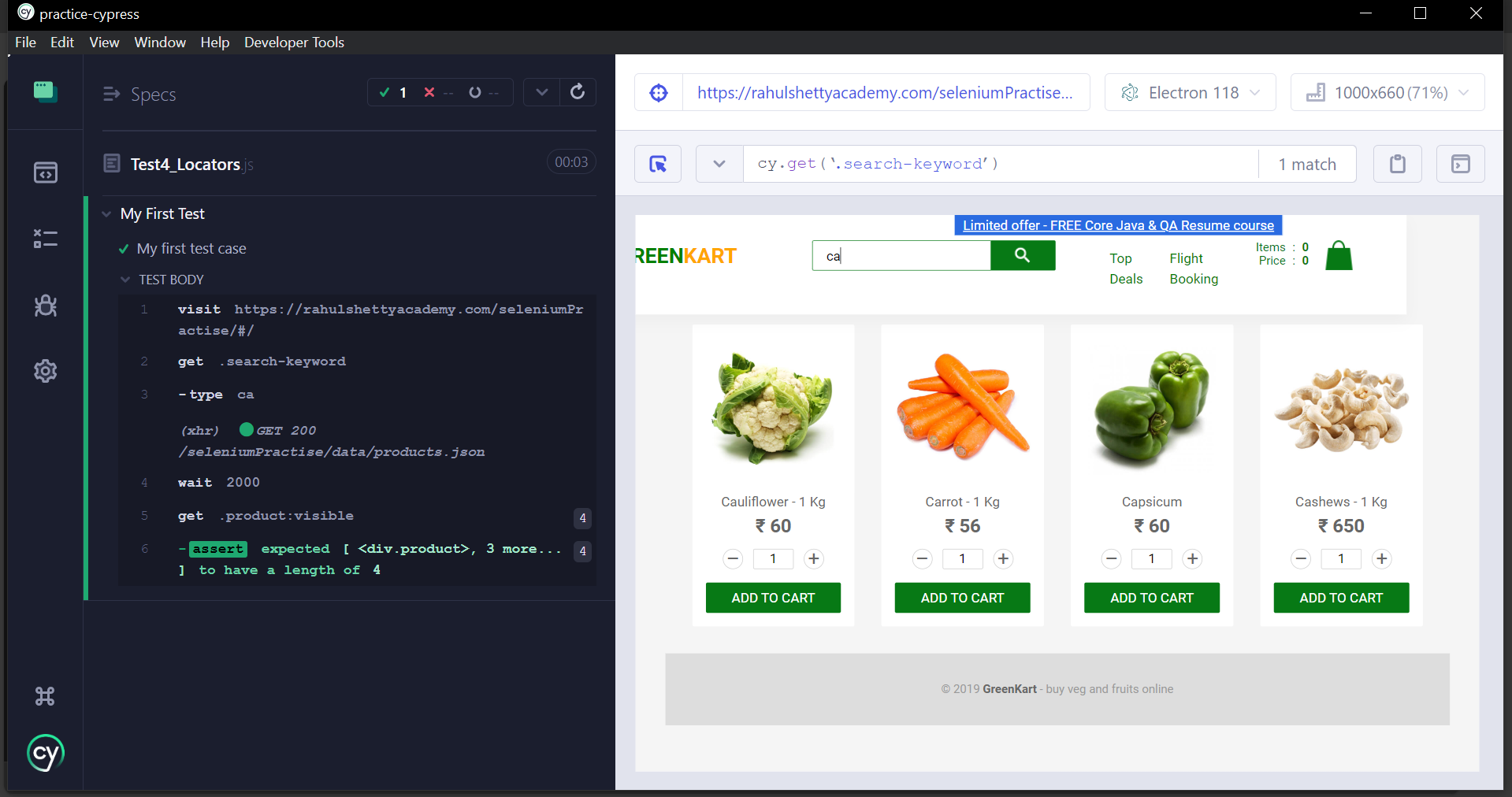
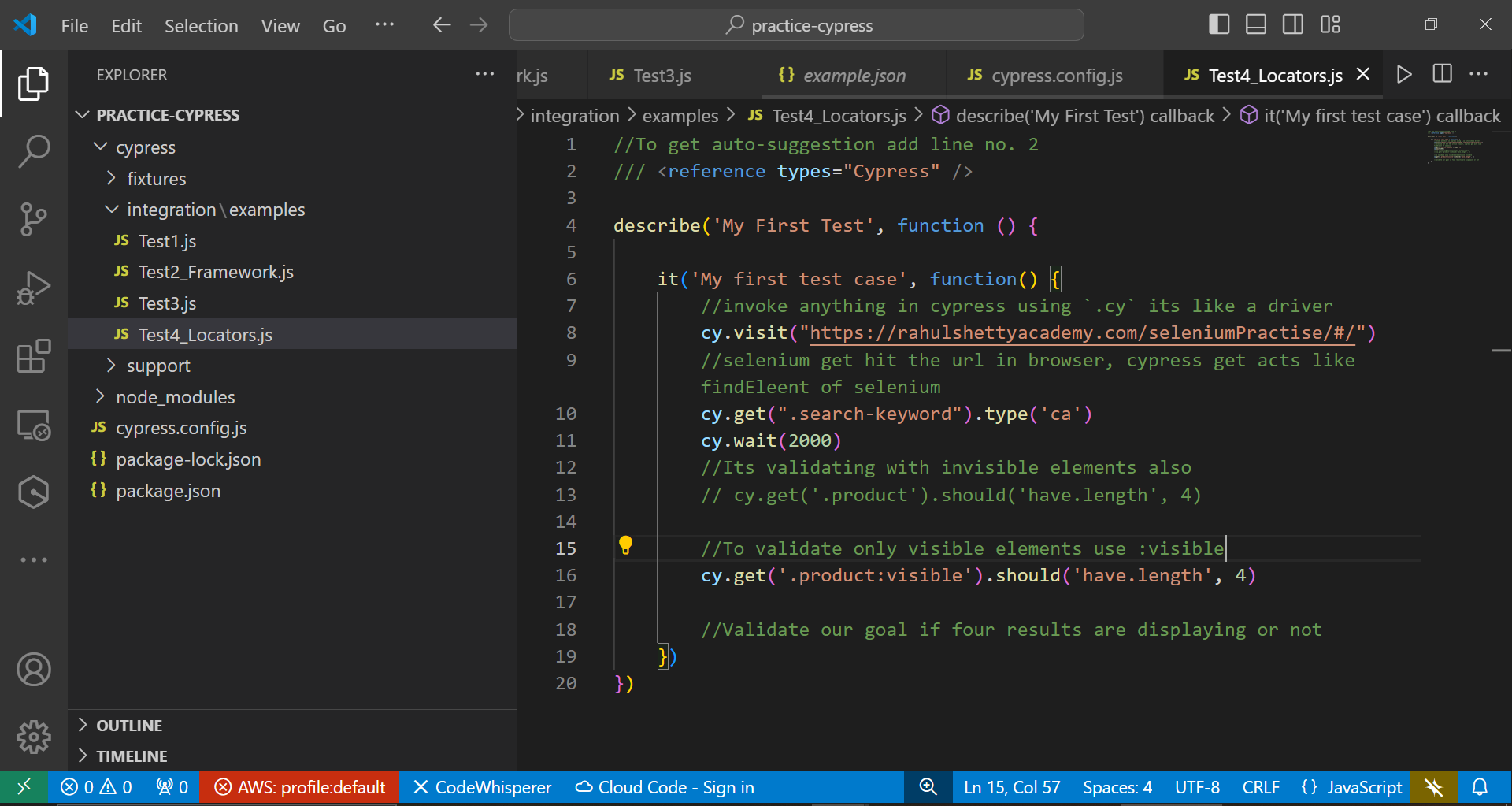
To get auto-suggestion under testcase file write:

//To get auto-suggestion add line no. 2

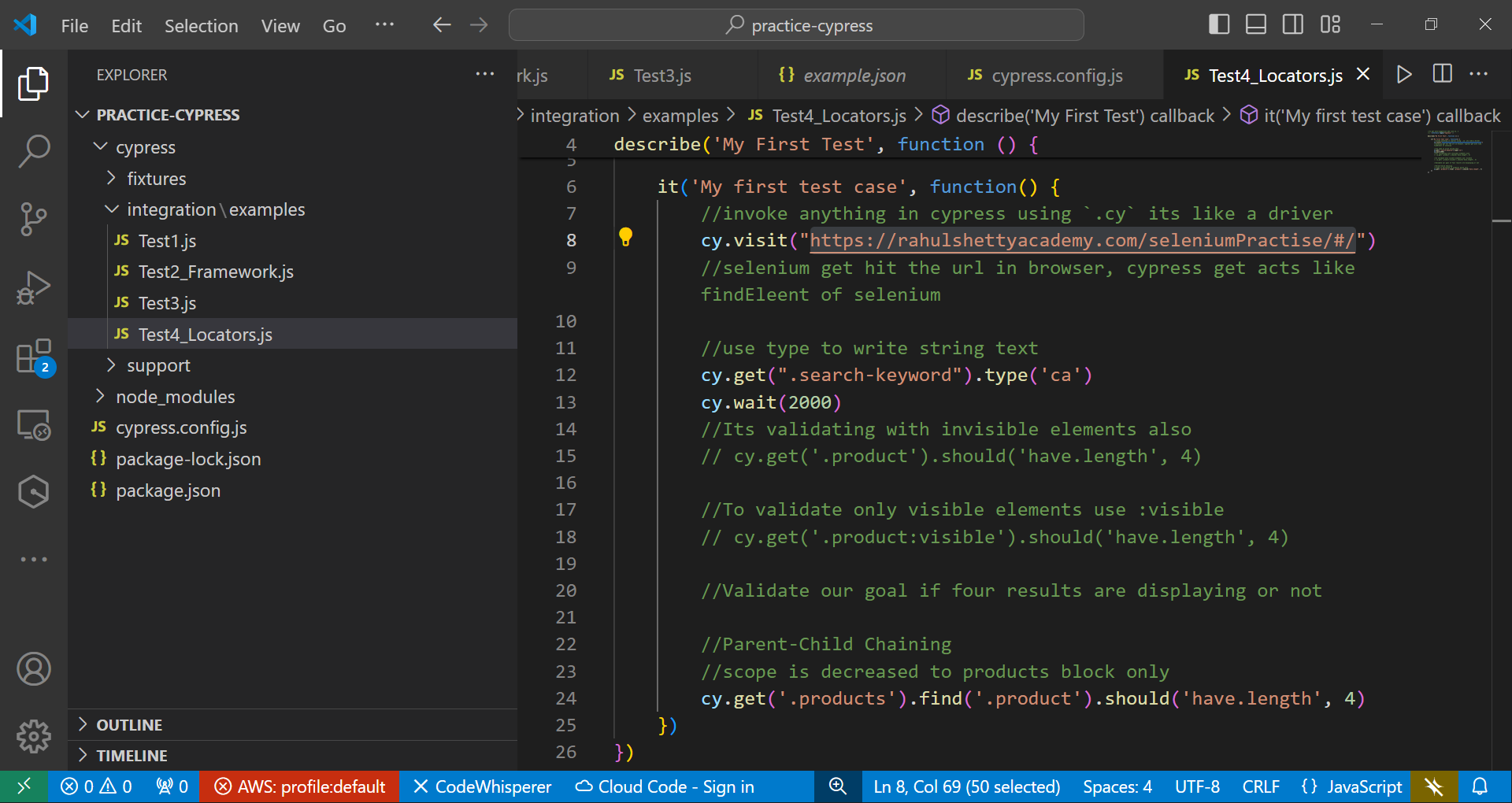
/// <reference types="Cypress" />

With Locator TestCase:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Parent-Child Chaining



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

First UI Test Case: Test4\_Locators.js

//To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My First Test', function () {

it('My first test case', function() {

//invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/seleniumPractise/#/")

//selenium get hit the url in browser, cypress get acts like findEleent of selenium

//use type to write string text

cy.get(".search-keyword").type('ca')

cy.wait(2000)

//Its validating with invisible elements also

// cy.get('.product').should('have.length', 4)

//To validate only visible elements use :visible

// cy.get('.product:visible').should('have.length', 4)

//Validate our goal if four results are displaying or not

/\*Parent-Child Chaining

scope is decreased to products block only\*/

// cy.get('.products').find('.product').should('have.length', 4)

/\*

methods: contains, equal, find and get

Click on Add to Cart

and resolving the promise

\*/

cy.get('.products').find('.product').eq(2).contains('ADD TO CART').click().then(function(){

console.log("Hello Cypress")

})

/\* Req: Grab all the product name and add only capsicum when its found

iterate where capsicum is present

Method: each\*/

/\*

To RE-USE LOCATORS EVERYTIME

// Aliases: part of optimization, to act as a variable

cy.get('.products').as('productsLocator')

//Before

cy.get('.products').find('.product').should('have.length', 4)

//After

[cy.get('@productsLocator').find('.product').should('have.length](mailto:cy.get('@productsLocator').find('.product').should('have.length)', 4)

\*/

// Aliases: part of optimization, to act as a variable

cy.get('.products').as('productsLocator')

[cy.get('@productsLocator').find('.product').each(($e1](mailto:cy.get('@productsLocator').find('.product').each(($e1), index, $list) => {

const textVEG = $e1.find('h4.product-name').text()

if(textVEG.includes('Cashew'))

{

//with .find click method is depreciated so wrap it using cy.wrap() method

cy.wrap($e1).find('button').click()

console.log('HelloWorld')

}

})

//DO NOT USE

/\*

const logo = cy.get('.brand')

//To print something we use method: cy.log()

cy.log(logo.text())

\*/

//this is to assert if logo text is correctly displayed

cy.get('.brand').should('have.text', 'GREENKART')

//this is to print in logs

//DO USE

cy.get('.brand').then(function(logoelement)

{

//To print something we use method: cy.log()

cy.log(logoelement.text())

})

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Test5.js

//To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My Second Test', function () {

it('My second test case', function() {

//invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/seleniumPractise/#/")

cy.get(".search-keyword").type('ca')

cy.wait(2000)

cy.get('.products').as('productsLocator')

[cy.get('@productsLocator').find('.product').each(($e1](mailto:cy.get('@productsLocator').find('.product').each(($e1), index, $list) => {

const textVEG = $e1.find('h4.product-name').text()

if(textVEG.includes('Cashews'))

{

cy.wrap($e1).find('button').click()

}

})

cy.get('.cart-icon').click()

// Locator of PROCEED TO CHECKOUT--('button[css="1"]')

cy.contains('PROCEED TO CHECKOUT').click()

cy.contains('Place Order').click()

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

My Understanding:

Here's how you can structure your Cypress test suites and API endpoints for the mentioned modules:

\*\*API Endpoints:\*\*

- \*\*architectures.js\*\*: Defines API endpoints related to architectures.

- \*\*bsp.js\*\*: Contains API endpoints for BSP (Board Support Packages).

- \*\*networkInterface.js\*\*: Includes API endpoints related to network interfaces.

\*\*Alternative URLs:\*\*

- \*\*routes.js\*\*: Contains alternative URLs for the vLab module.

\*\*Page Objects:\*\*

- \*\*Common Components\*\*:

- \*\*navBar.js\*\*: Defines navigation bar elements with locators and keywords.

- \*\*sideNav.js\*\*: Contains locators and keywords for the side navigation.

- \*\*Pages\*\*:

- \*\*common\*\*: Page elements common across different modules.

- \*\*vLab\*\*: Page elements specific to the vLab module.

- \*\*vLabAdministration\*\*: Page elements related to vLab administration.

\*\*Test Cases:\*\*

- \*\*testSuites\*\*: Contains test suites covering various scenarios.

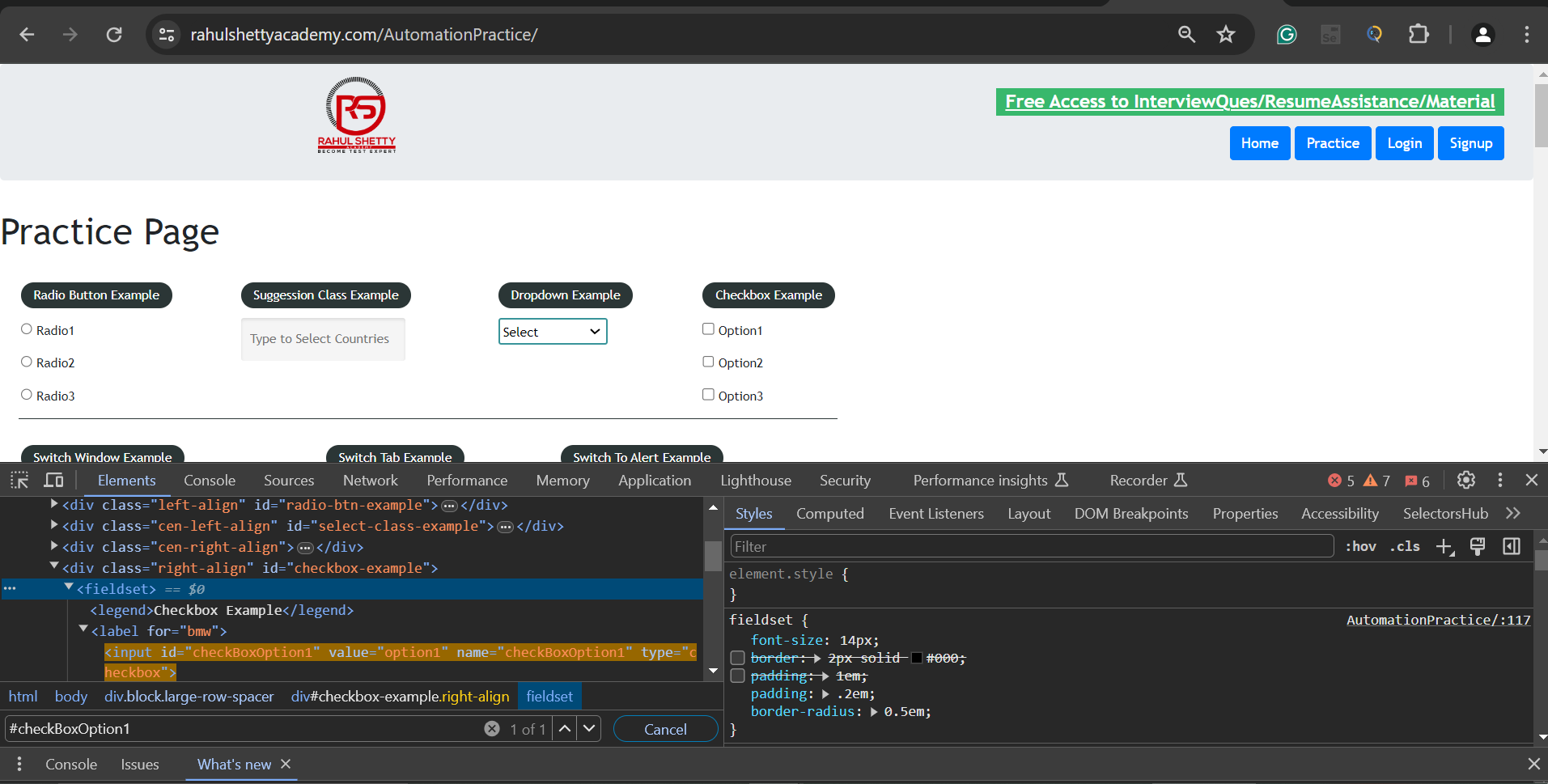
- \*\*vLabAdministration\*\*: Test cases specifically for vLab administration.

- \*\*vLabMenu\*\*: Test cases related to the vLab menu functionality.

In your Cypress setup, you can organize your code according to these modules and folders for easier navigation and maintenance.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How to write CSS for checkboxes





//To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My Third Test', function () {

it('My Third test case', function() {

//invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

/\*verify if the checkbox is check or not

validate if option1 is clicked or not\*/

//First: It will check option1

cy.get('#checkBoxOption1').check().should('be.checked').and('have.value','option1')

//Second: It will uncheck option1

cy.get('#checkBoxOption1').uncheck().should('not.be.checked')

//Third: It will check option2 and 3

cy.get('input[type="checkbox"]').check(['option2','option3'])

//Four: It will uncheck option2 and 3

cy.get('input[type="checkbox"]').uncheck().should('not.be.checked')

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

When dealing with dynamic dropdowns, if the options disappear after typing an individual text into a search box, you can use the inspect option in your browser's developer tools to identify the CSS path of the elements.

//To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My Fourth Test', function () {

it('My Fourth test case', function() {

//invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

/\*

dynamic dropdowns -- means options will be displayed based on inputs provides

[tagname: select] static dropdowns -- means we can select from the given suggestions

\*/

//Static Dropdowns

cy.get('select').select('option2').should('have.value', 'option2')

//Dynamic dropdowns

cy.get('#autocomplete').type('ind')

cy.get('.ui-menu-item div').each(($el, index, $list) => {

if($el.text()==="India")

{

$el.click()

}

})

//verifying if its have value as india or not

cy.get('#autocomplete').should('have.value', 'India')

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hide and Show

//To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My Third Test', function () {

it('My Third test case', function() {

//invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

cy.get("#displayed-text").should('be.visible')

cy.get("#hide-textbox").click()

cy.get("#displayed-text").should('not.be.visible')

cy.get("#show-textbox").click()

cy.get("#displayed-text").should('be.visible')

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Alert:

// To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My Fifth Test', function () {

it('My Fifth test case', function() {

// invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

// cypress auto accepts alerts

cy.get('#alertbtn').click()

cy.get('#confirmbtn').click()

// window:alert

cy.on('window:alert', (str)=>{

// how to compare two str

expect(str).to.equal('Hello , share this practice page and share your knowledge')

})

// window:confirm

cy.on('window:confirm', (str)=>{

// how to compare two str

expect(str).to.equal('Hello , Are you sure you want to confirm?')

})

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Navigate to another domain

// To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My six Test', function () {

it('My Six test case', function() {

// invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

// Because every child url have target attribute in they html

cy.get('#opentab').invoke('removeAttr','target').click()

/\*

to further automate things on the next page do follow the thing below:

cy.origin("<next page url>", ()=>

{

//code

})

\*/

cy.origin("https://www.qaclickacademy.com", ()=>

{

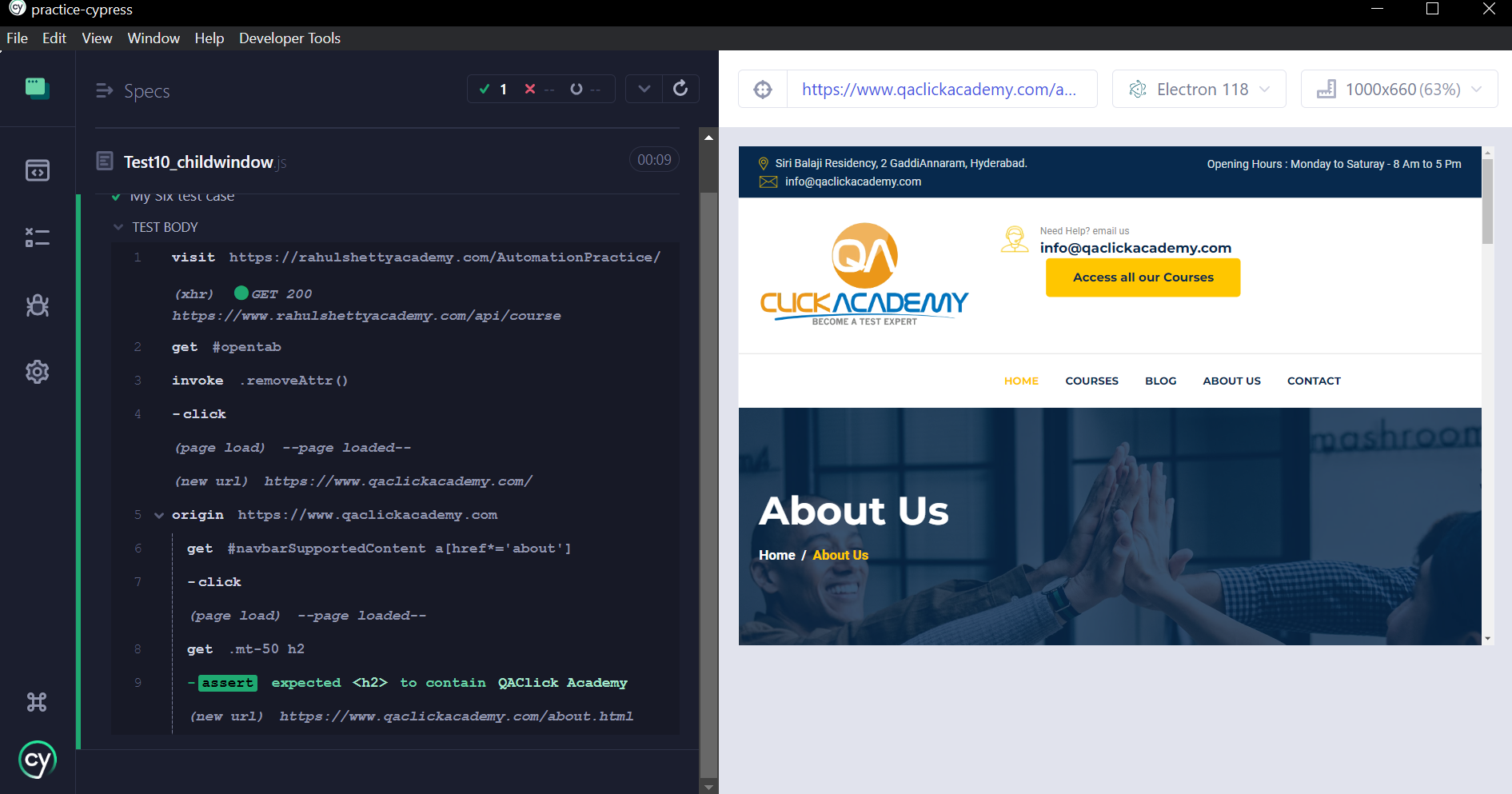
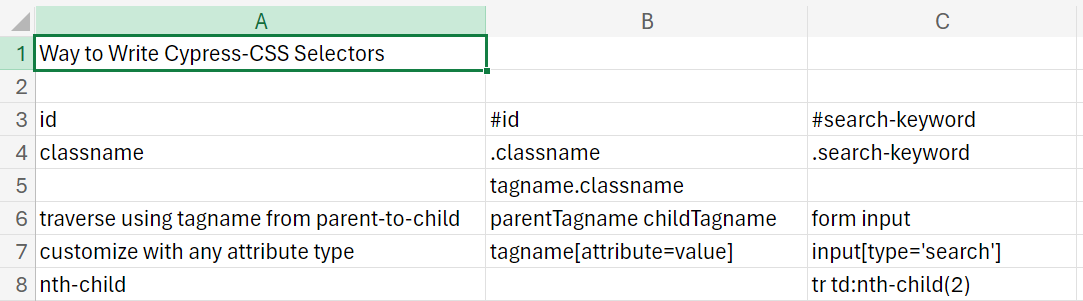
cy.get("#navbarSupportedContent a[href\*='about']").click()

cy.get(".mt-50 h2").should('contain','QAClick Academy')

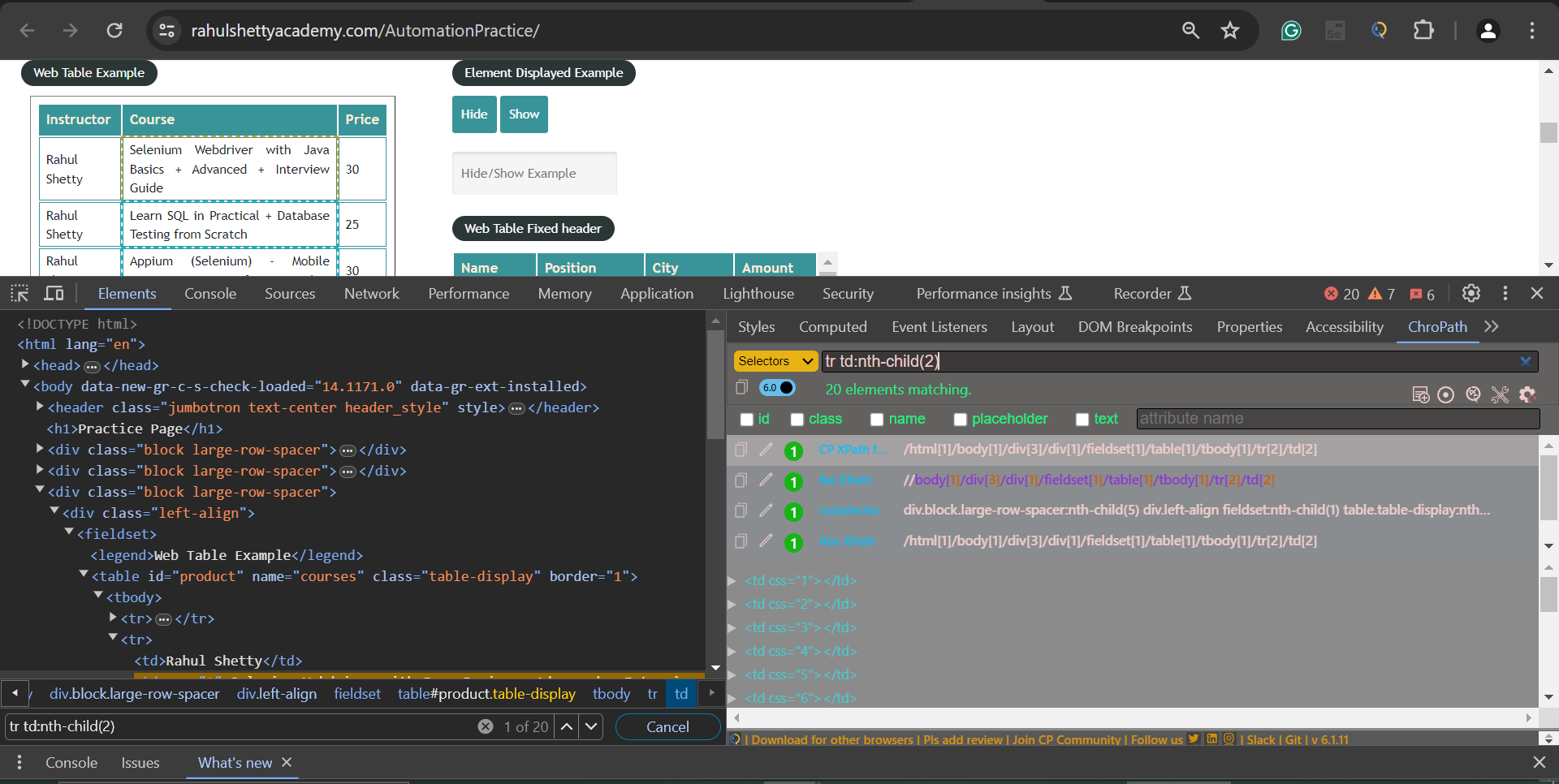
})

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tables:



// To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My seventh Test', function () {

it('My Seventh test case', function() {

// invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

//get the specific column data

cy.get('tr td:nth-child(2)').each(($e1, index, $list) => {

const text = $e1.text()

if(text.includes('Python'))

{

// how to find next sibling using method `.next()`

cy.get('tr td:nth-child(2)').eq(index).next().then(function(price)

{

const priceText= price.text()

expect(priceText).to.equal('25')

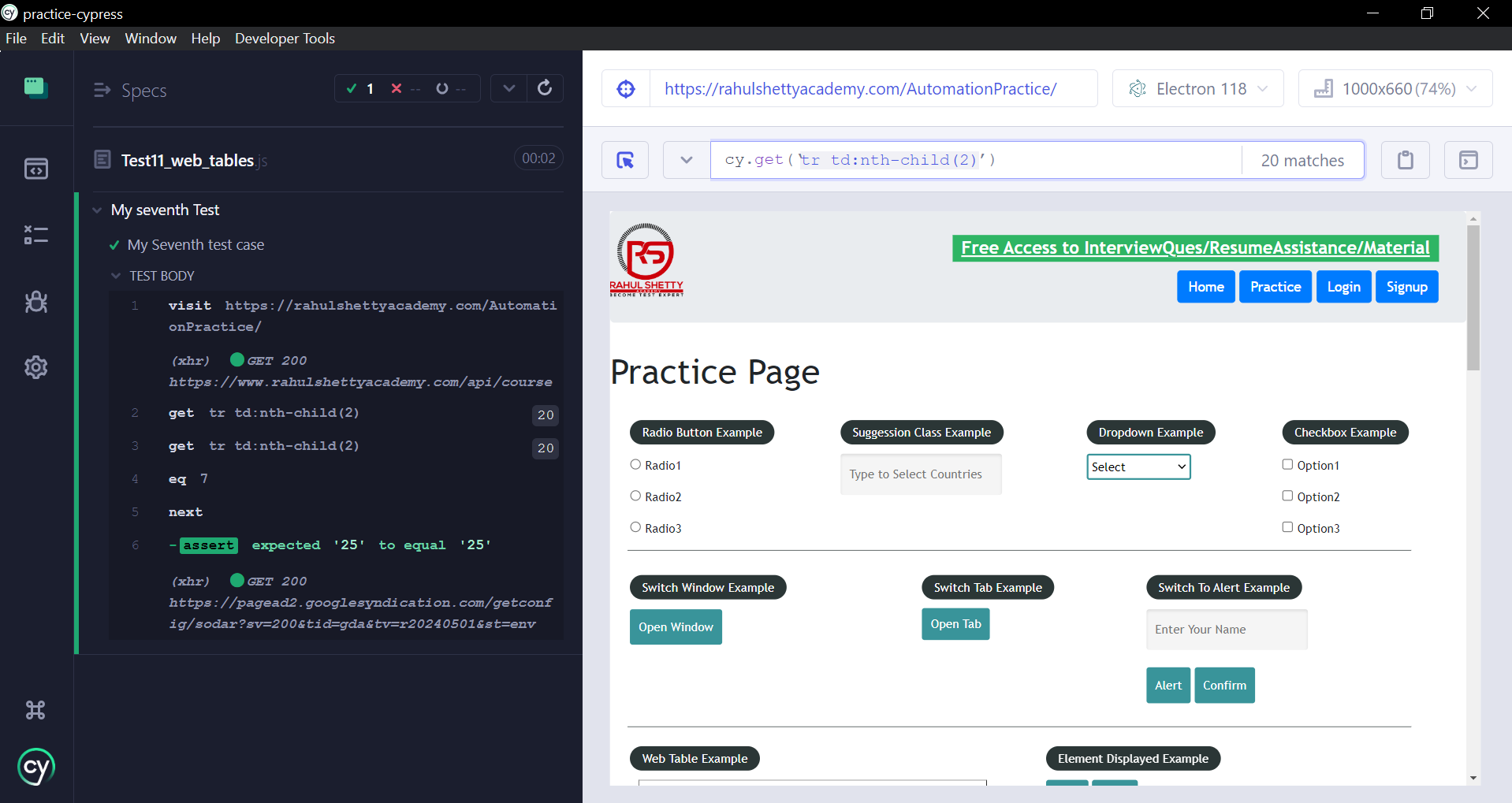
})

}

})

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mouse-Hover

// To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My eight Test', function () {

it('My eight test case', function() {

// invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

/\*

//give the parent near element css

cy.get('div.mouse-hover-content').invoke('show')

cy.contains('Top').click()

cy.url().should('include', 'top')

\*/

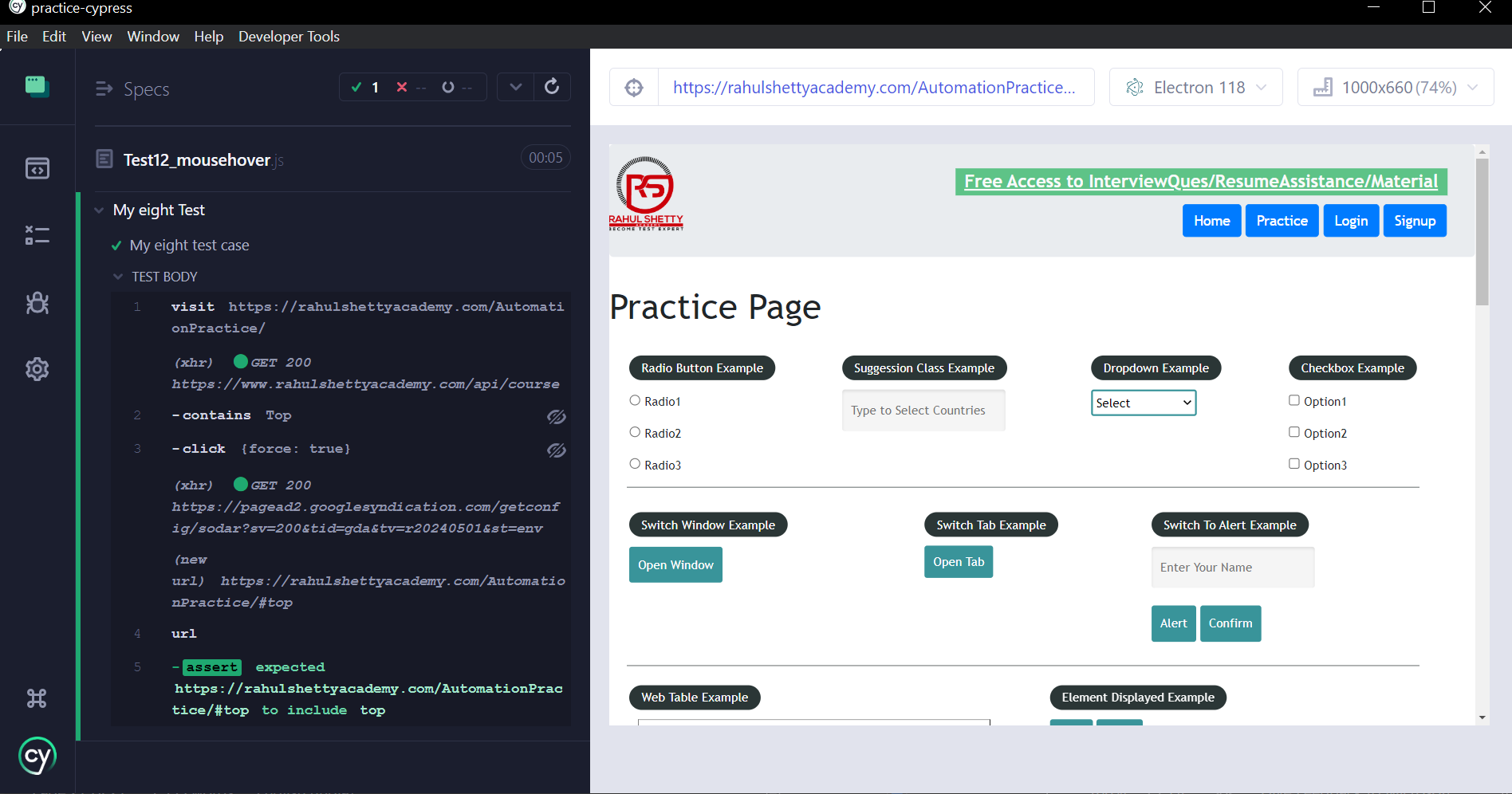
// forcefully clickinh hidden pop-ups

cy.contains('Top').click({force:true})

cy.url().should('include', 'top')

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Child Window:

// To get auto-suggestion add line no. 2

/// <reference types="Cypress" />

describe('My eight Test', function () {

it('My eight test case', function() {

// invoke anything in cypress using `.cy` its like a driver

cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

/\*

directly hit the url without clicking on any button

or user href tag to visit that url

1. To click url

2. To visit href link

\*/

/\*

open window

Here we are concatenating cypress method with prop method prop is not a cypress method, so first we are resolving this promise

\*/

cy.get('#opentab').then(function(e1){

const url =e1.prop('href')

//using .visit can redirect you to the particular url, you cant perform automation

cy.visit(url)

//inside func only: new url page can be automated

cy.origin(url, ()=>

{

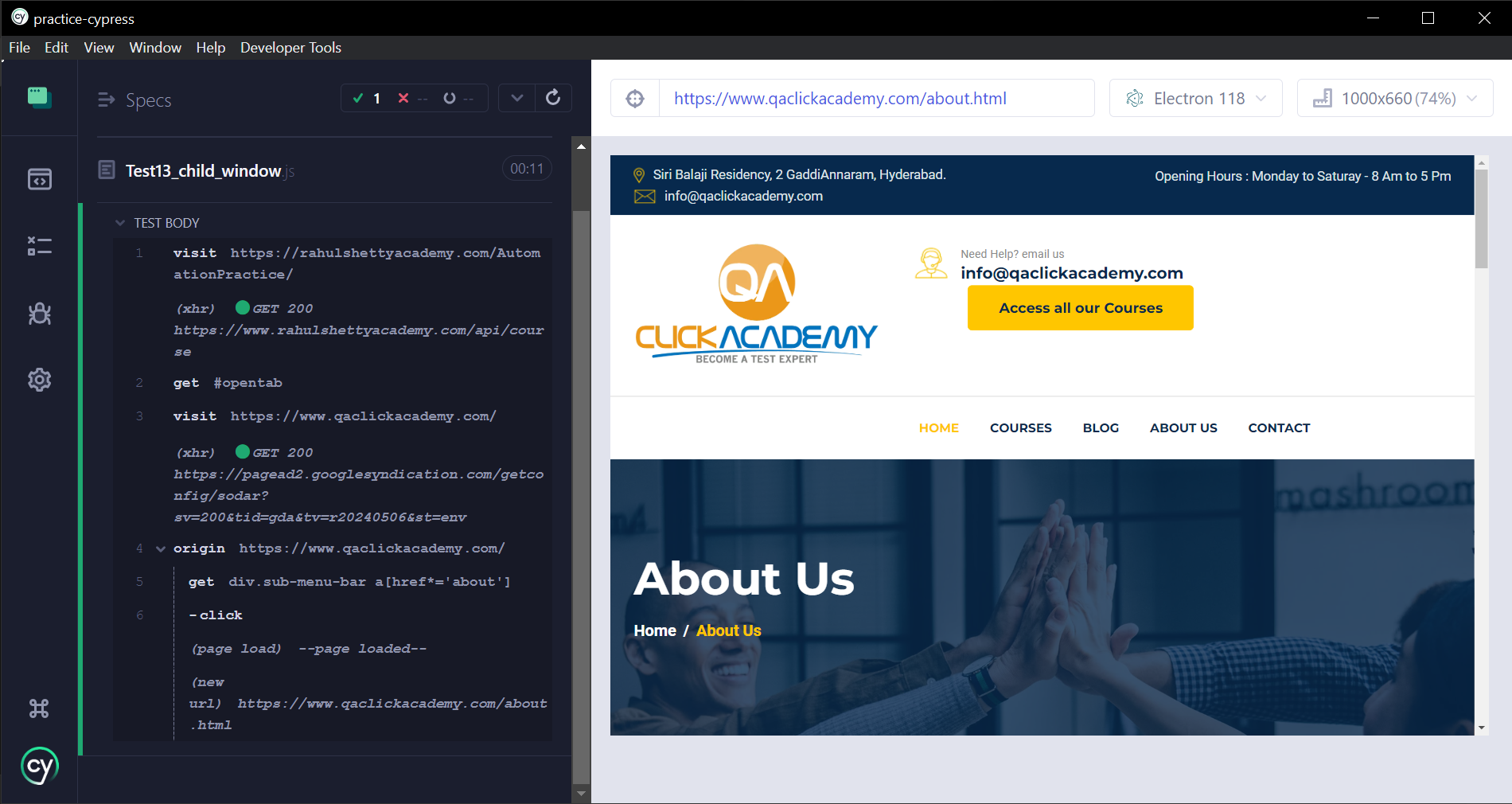
cy.get("div.sub-menu-bar a[href\*='about']").click()

})

})

})

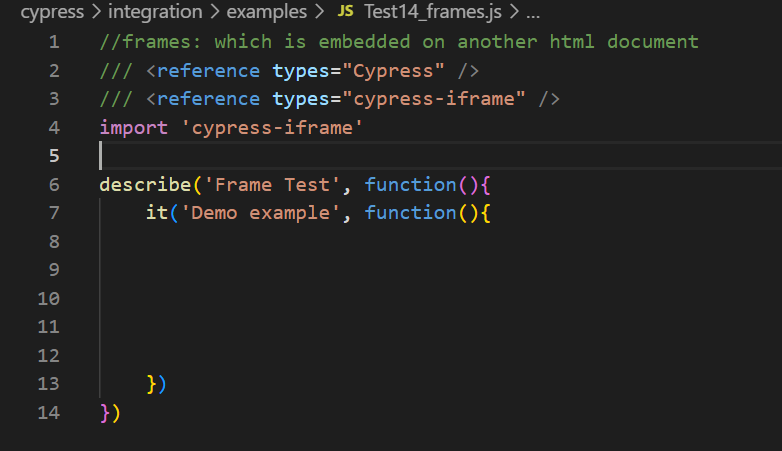
})

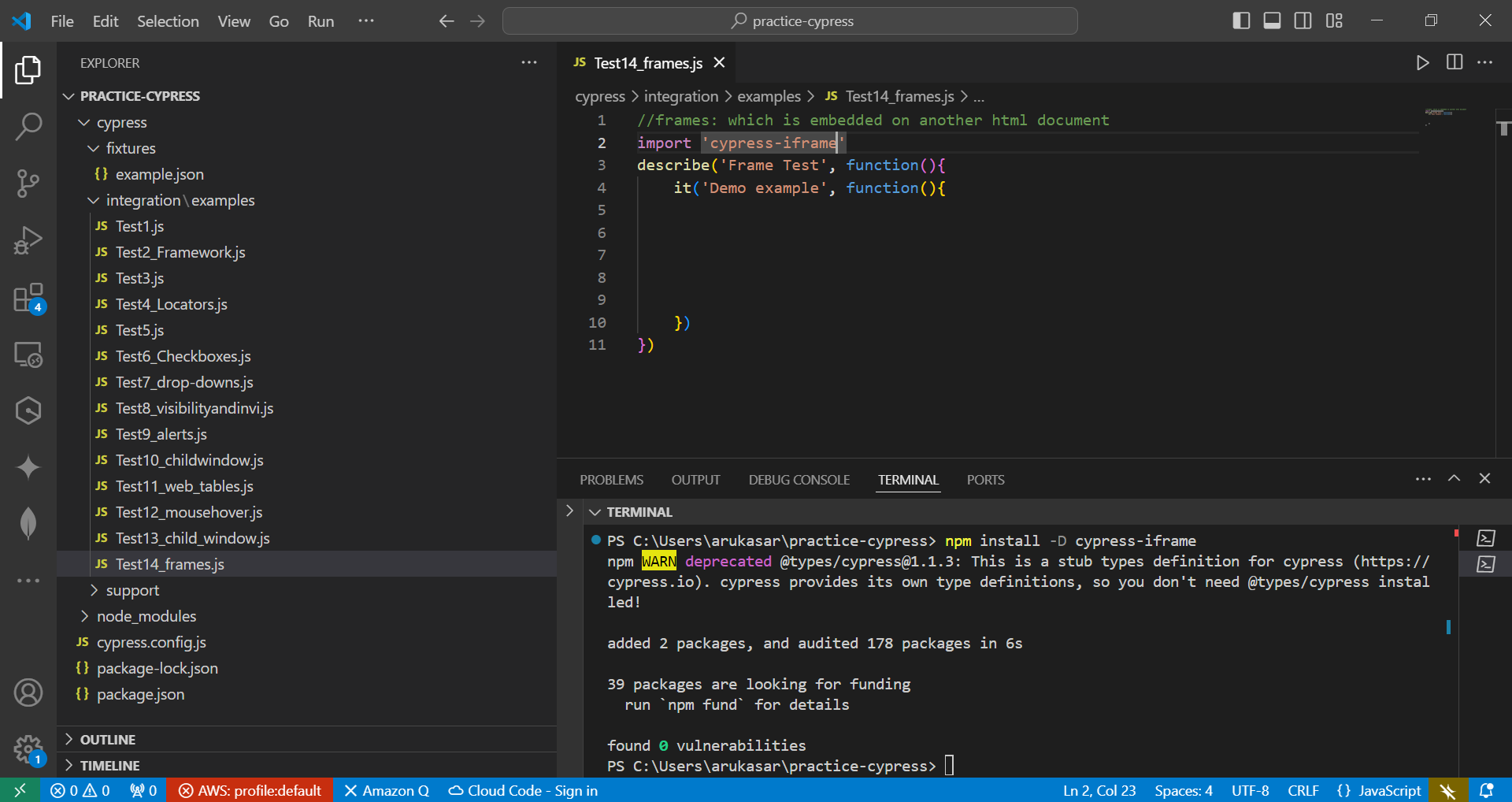
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Frame

Command to install iframe: npm install -D cypress-iframe

Note: Do not forget to import





//frames: which is embedded on another html document

/// <reference types="Cypress" />

/// <reference types="cypress-iframe" />

import 'cypress-iframe'

describe('Frame Test', function(){

it('Demo example', function(){

cy.visit('https://rahulshettyacademy.com/AutomationPractice/')

cy.frameLoaded('#courses-iframe')

cy.iframe().find("a[href\*='mentorship']").eq(0).click()

// cy.iframe().find("h1[class\*='pricing-title']").should('have.length',2)

})

})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_