
# Automation Testing with Protractor

# You Already Know

Before we begin, lets see what and all have you covered till now:

- Agile
- Git
- SQL
- HTML/CSS
- JavaScript
- Angular
- Cypress

# Recap

**Agile**

An iterative approach to manage the development of a Software  Project

**Git and GitHub**

A distributed version control system with which we can handle our software projects

**SQL/MySQL**

Relational Database Management System to store data in a structured way using tables

**HTML CSS**

Design interactive Web Pages

**JavaScript**

Programming language for the Web Pages.

**Angular**

A platform and framework by google to create single page web applications using HTML and TypeScript.

**Cypress**

An end-to-end testing framework that is not based on WebDriver to test any website.

# A Day in the Life of a Full Stack Developer

**PS: ID to Create SCENARIO Here with below data**

As an Automation Test Engineer, our key role is to test both client and server software with the latest  test automation tools.

We shall be testing food delivery application built in Angular, Node as the front end with Spring Boot, Java and MySQL/MongoDB as the backed.

Since we have configured cypress for the Angular Projects, now we will write scripts initially to perform automation testing on Angular Projects.

We will see how page object model can be used to perform automation testing in cypress

# Learning Objectives

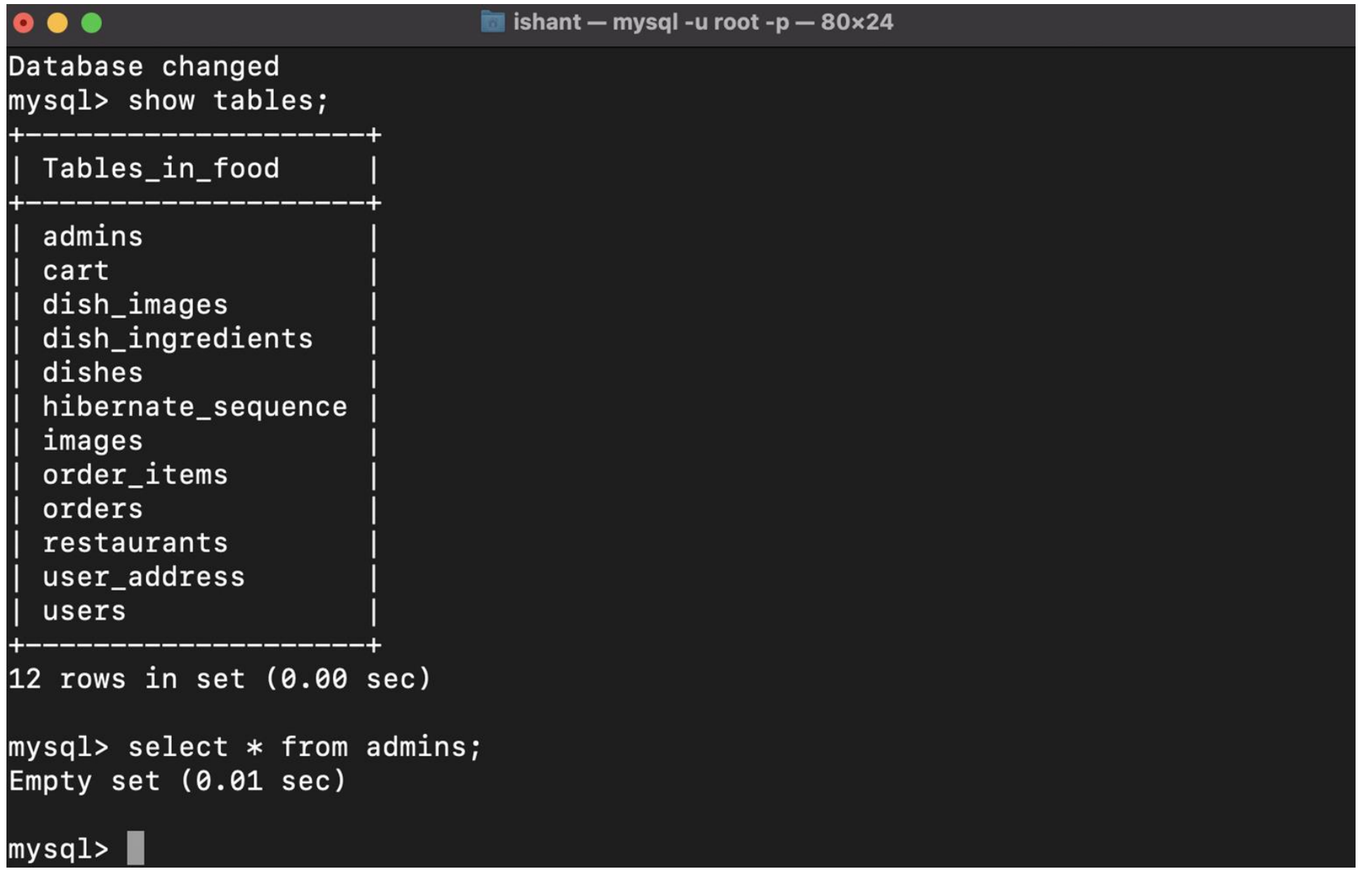
By the end of this lesson, you will be able to:

- Write Test Scripts for Angular Projects in Cypress
- Work with Locators to perform automation testing
- Work with Elements to perform automation testing
- Create Page Object Model in Cypress

# Task 1: Test the Angular Admin Web Pages with Cypress

# Create an Admin Credential in MySQL

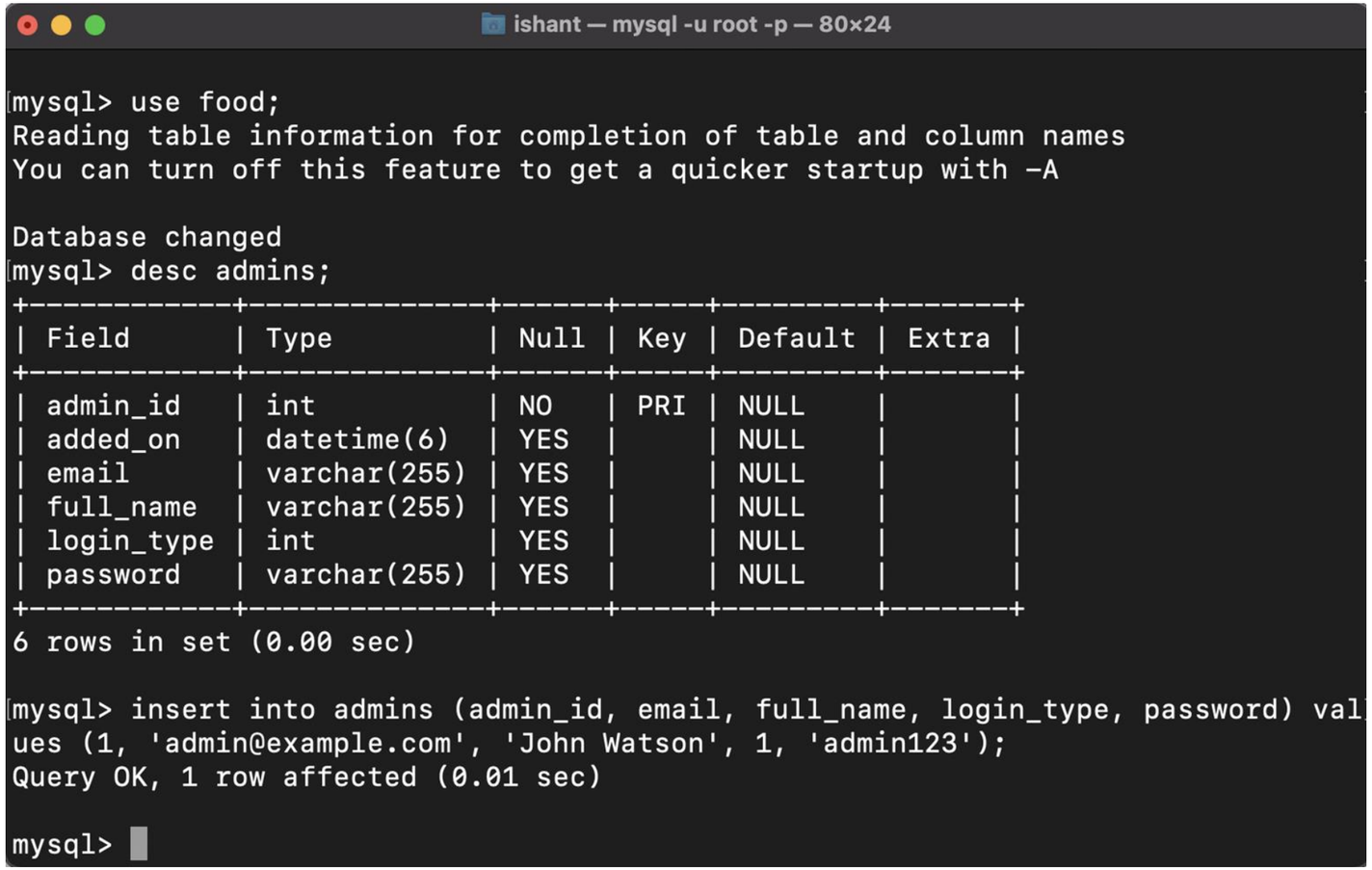
In order to begin our automation testing of Admin Project with protractor, we need to create a login credential for the admin in the database manually as no record is available in table admins.



# Create an Admin Credential in MySQL

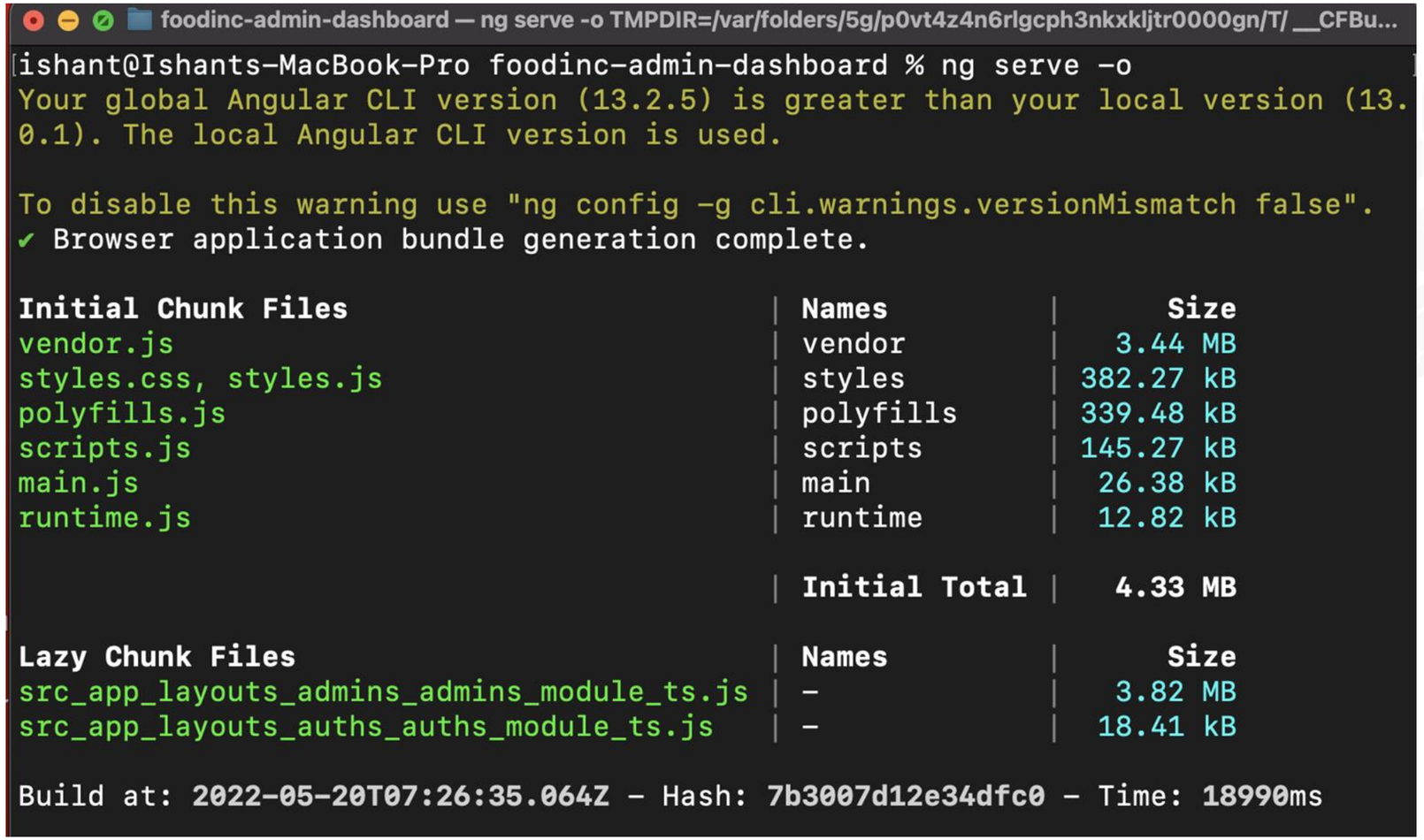
Use the command to create a new record in admins table.

insert into admins (admin_id, email, full_name, login_type, password) values (1, 'admin@example.com', 'John Watson', 1, 'admin123');



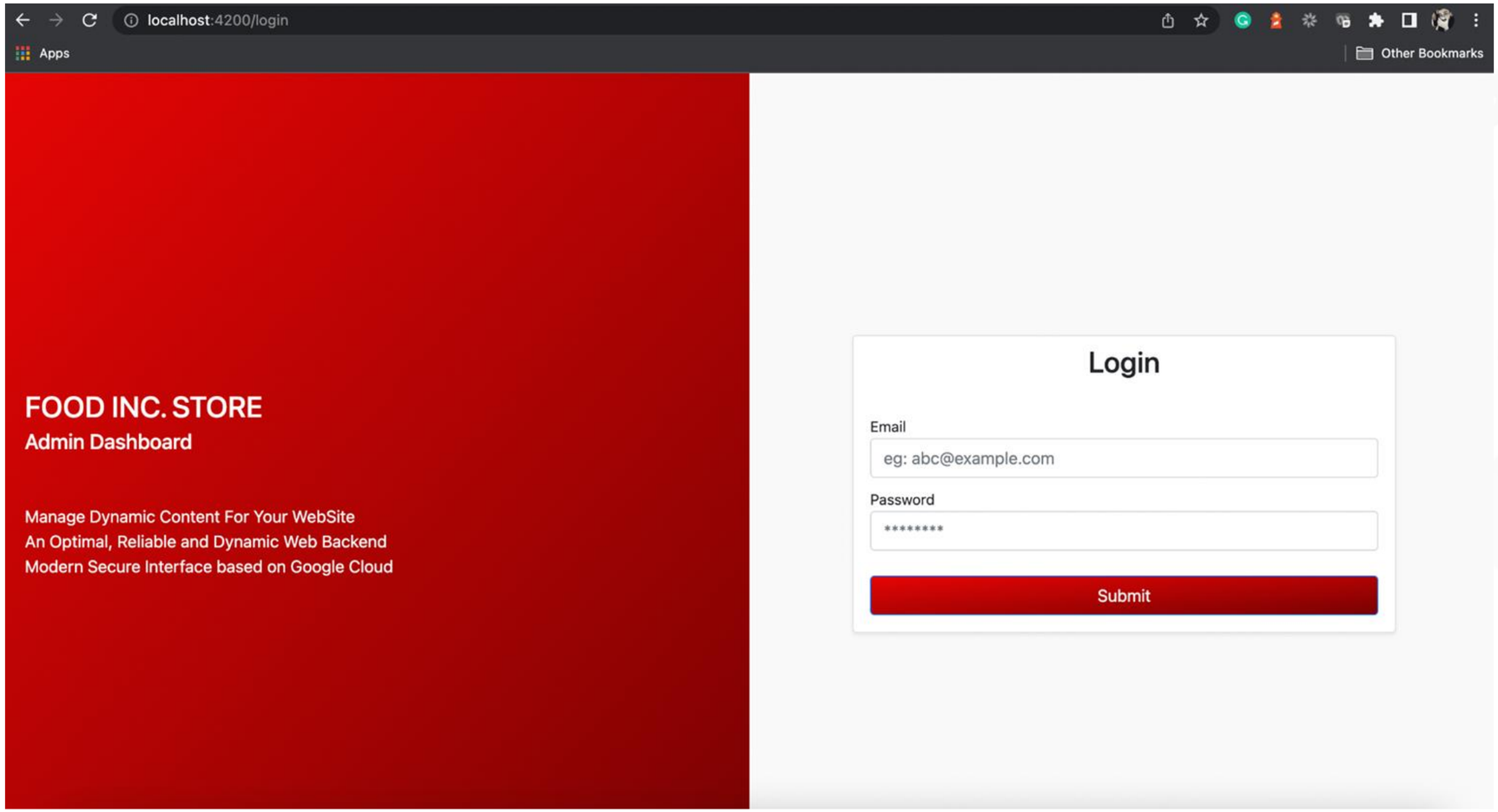
# Build and execute Angular Admin Project

Execute **ng serve -o** command to build and execute the project



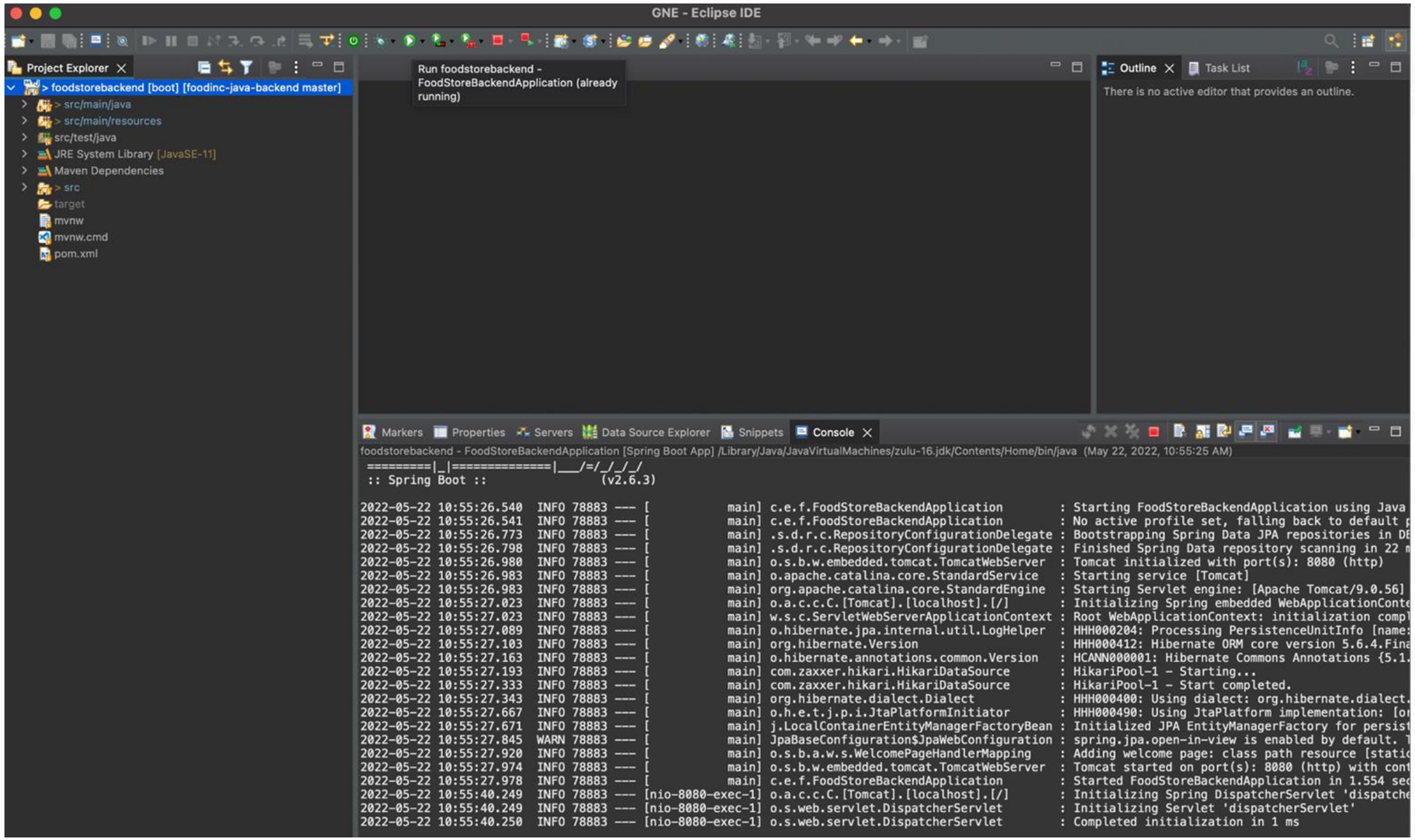
# Build and execute Angular Admin Project

Your project is up and running fine on localhost port 4200, with initial UI screen to login



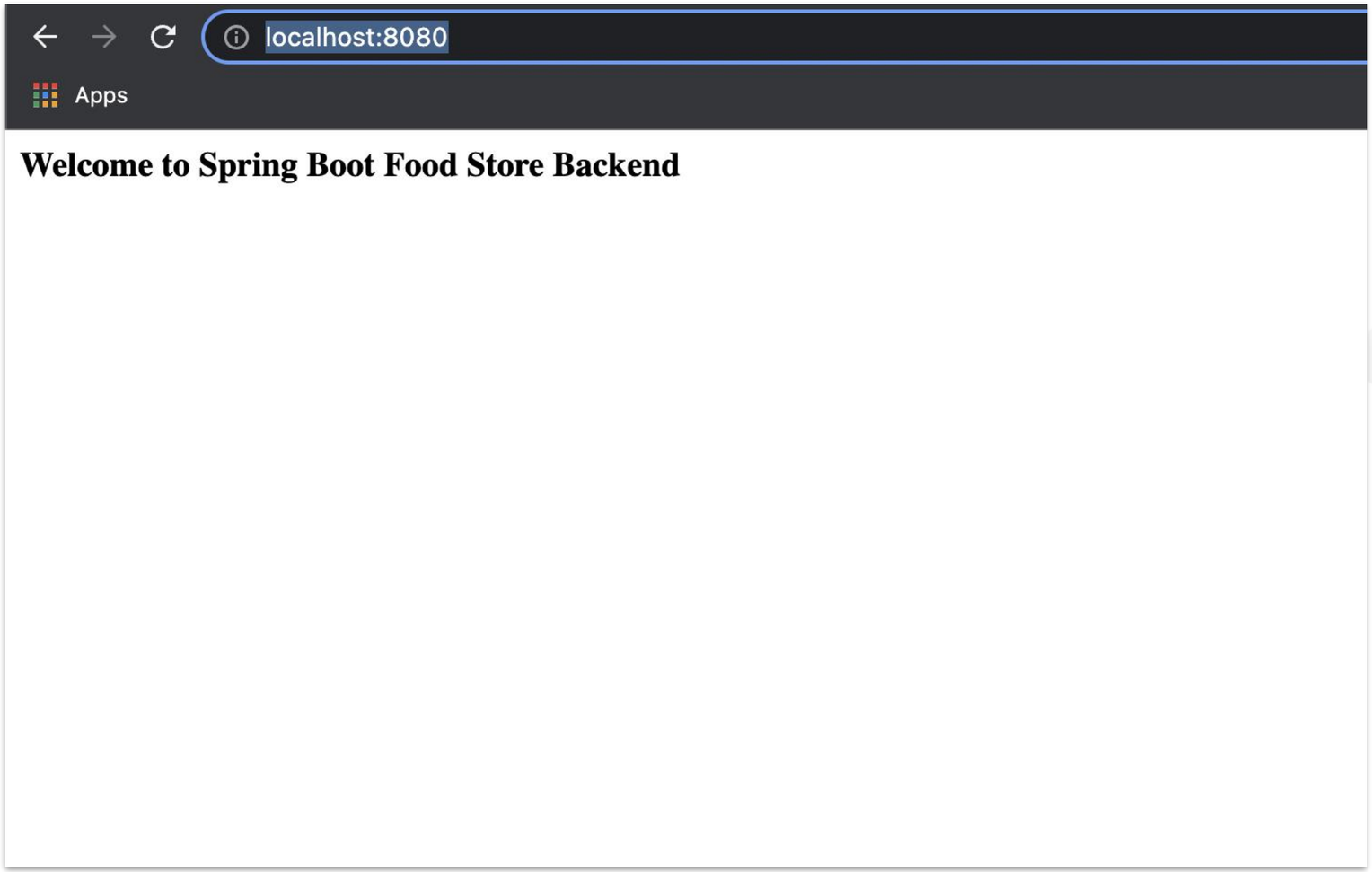
# Build and execute Java Backend Project

For Angular Apps to function with database, java backend should be running. Run Your project from Eclipse EE or You can create jar file by executing a maven build with goal as **clean package** and run it on your terminal by executing command **java -jar foodStoreBackend-0.0.1-SNAPSHOT.jar**



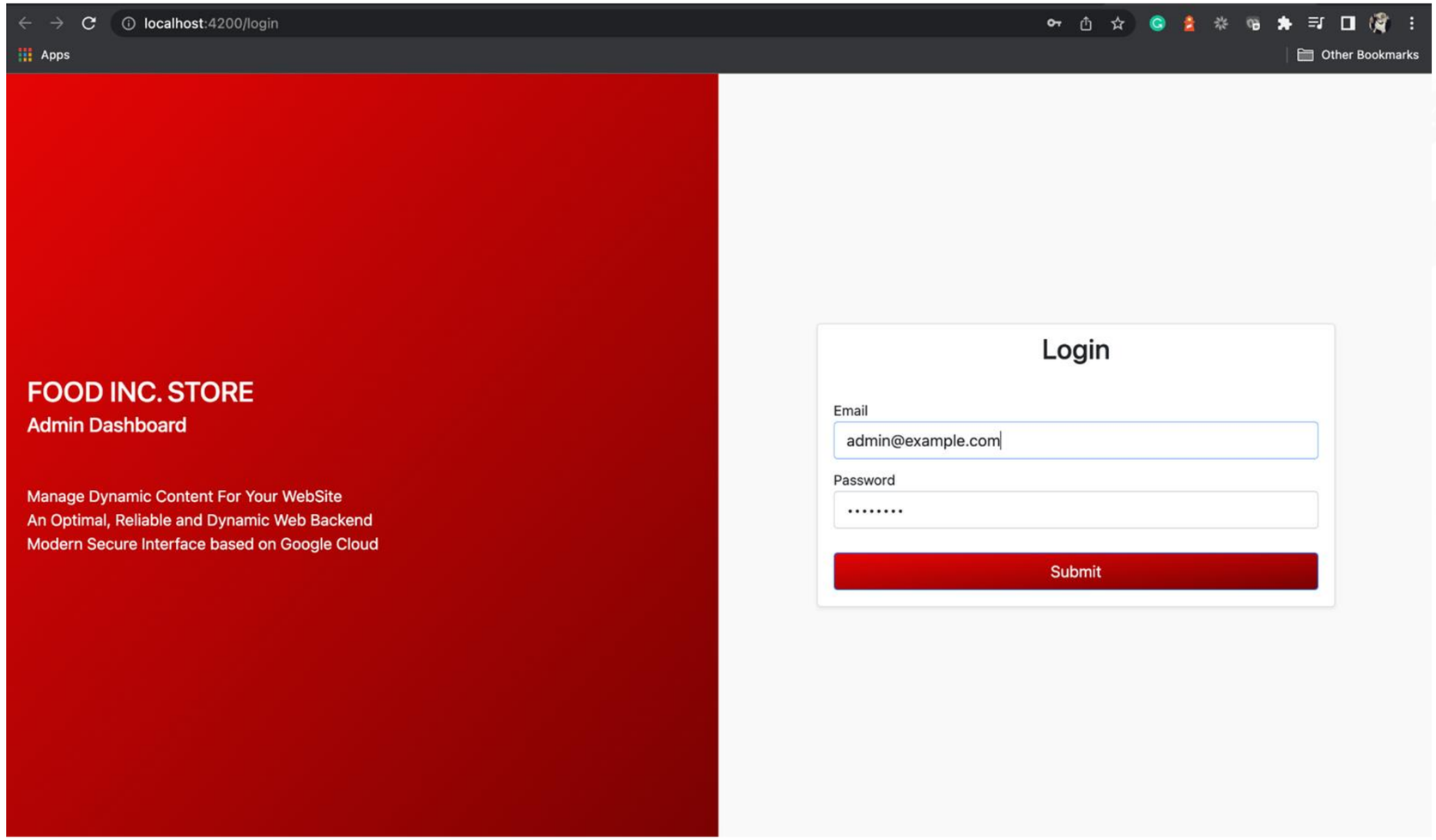
# Check Java Backend Project Status as Running

Open the Web Browser and type http://localhost:8080/ to check if your project is up and running fine



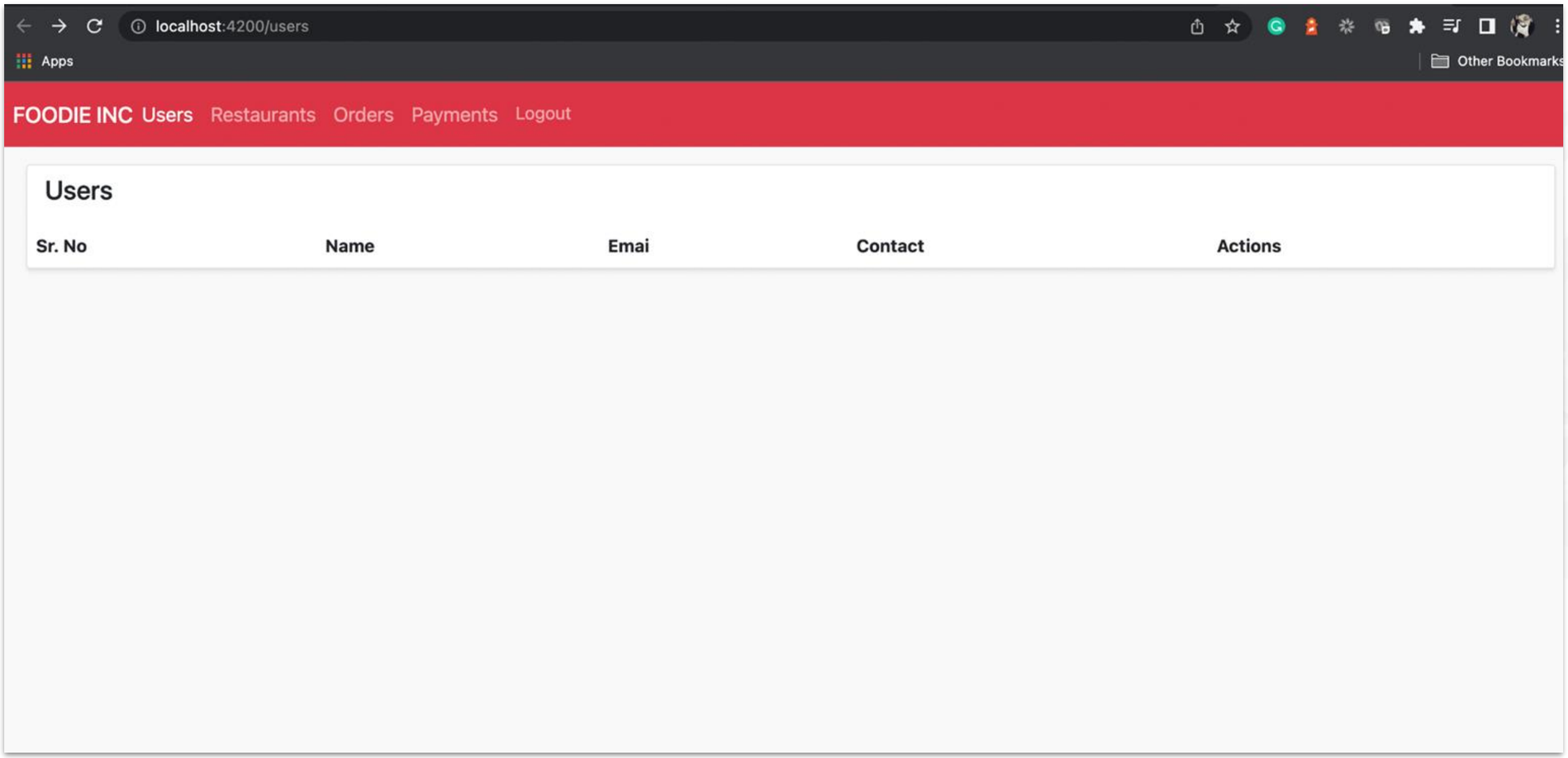
# Login with your Admin Credentials

Now, enter the credentials to login to the Admin Project and validate the login flow with newly generated email and password.



# Login with your Admin Credentials

Once, the login is successful, you will be navigated to home page.



# Write Test Case for Login Page

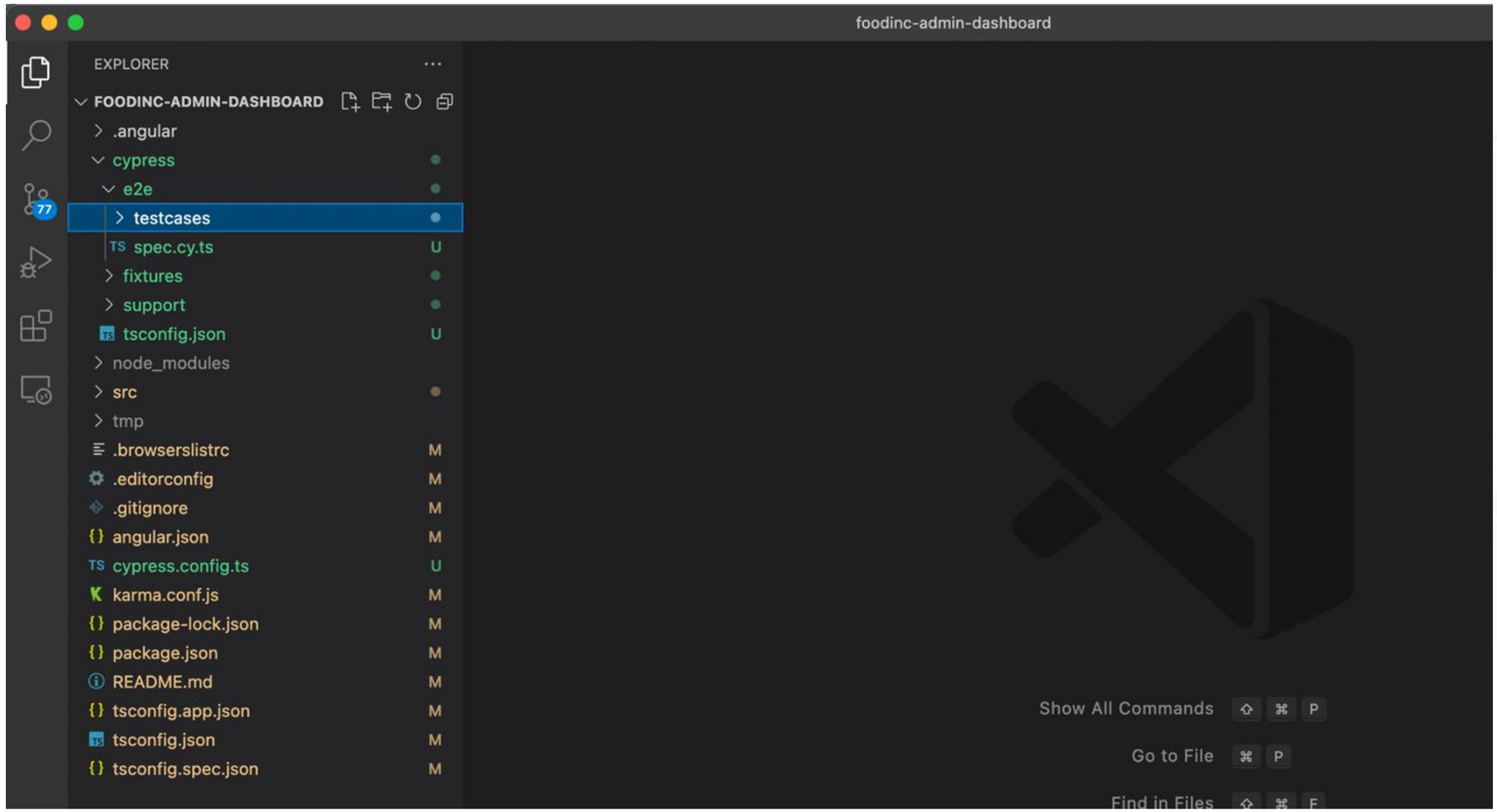
Let us start by writing a simple test for our admin web application. We will follow the below steps:

1. Create a directory testcases in your project structure.
2. Create a file login-spec.cy.ts and write the first scenario to test the login form for correct labels
3. Write the second scenario to test the login with email and password
4. Write the third scenario to test the login with email and password with redirection
5. Execute the Test Cases

# Write Test Case for Login Page

1. Create a directory testcases in your project structure in your cypress/e2e directory.

.

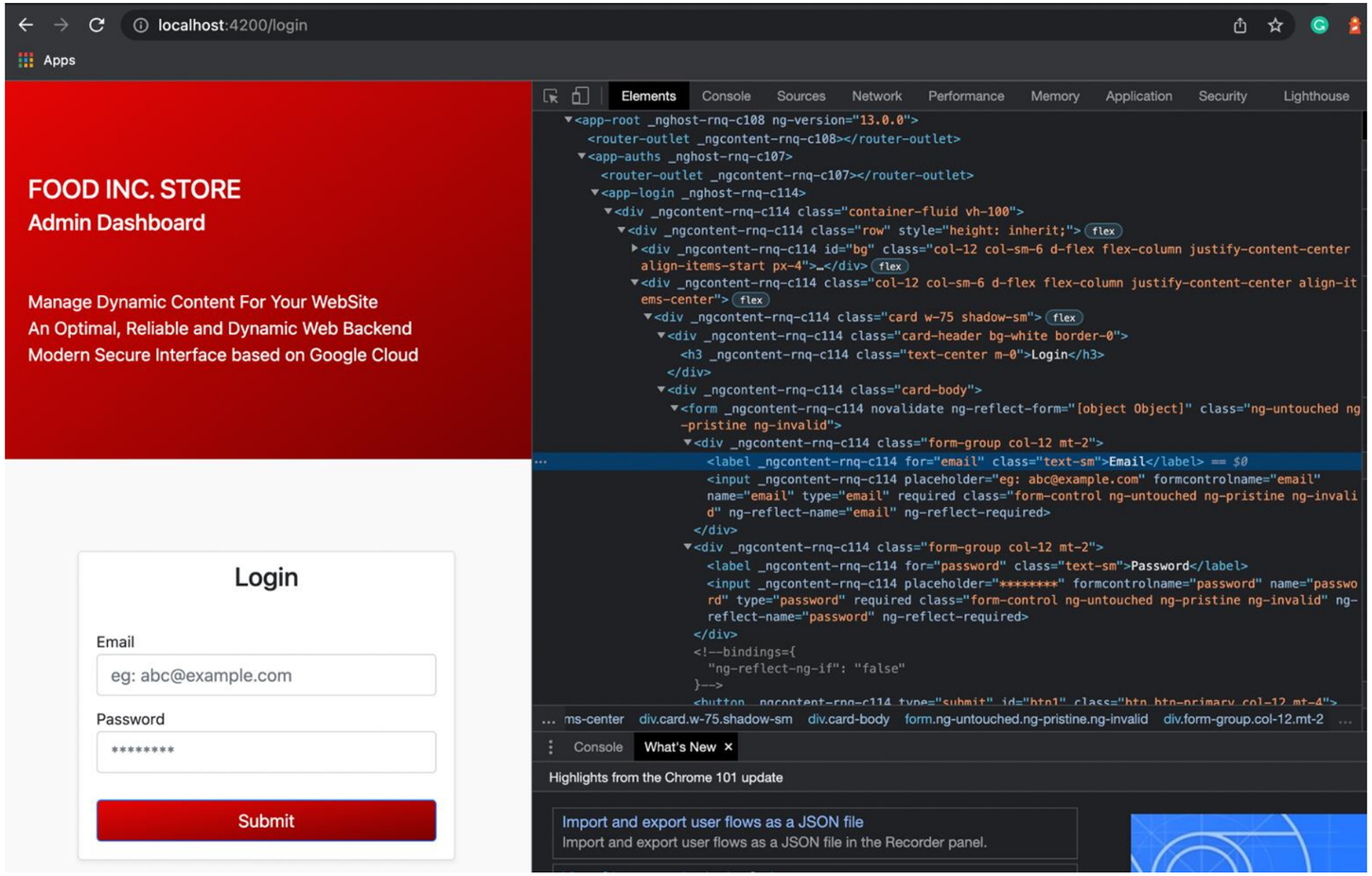


# Write Test Case for Login Page

2. Create a file login-spec.cy.ts and write the first scenario to test the login form for correct labels

We will create a scenario **should have correct labels**

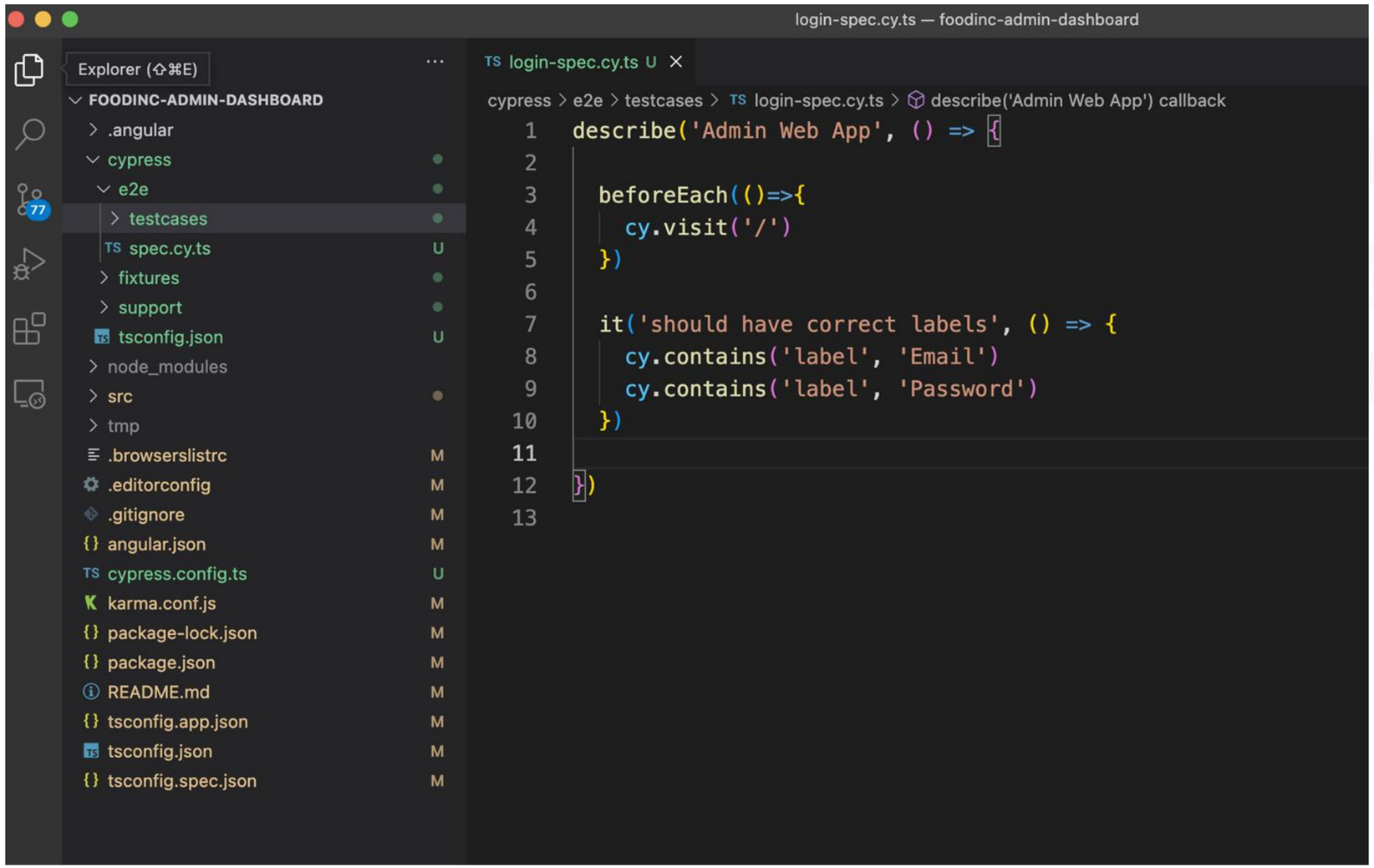
In order to check the labels, we will use CSS Locator. Right click on your web app and open elements to find the css class for the labels.



# Write Test Case for Login Page

2. Create a file login-spec.cy.ts and write the first scenario to test the login form for correct labels

Scenario **should have correct labels**

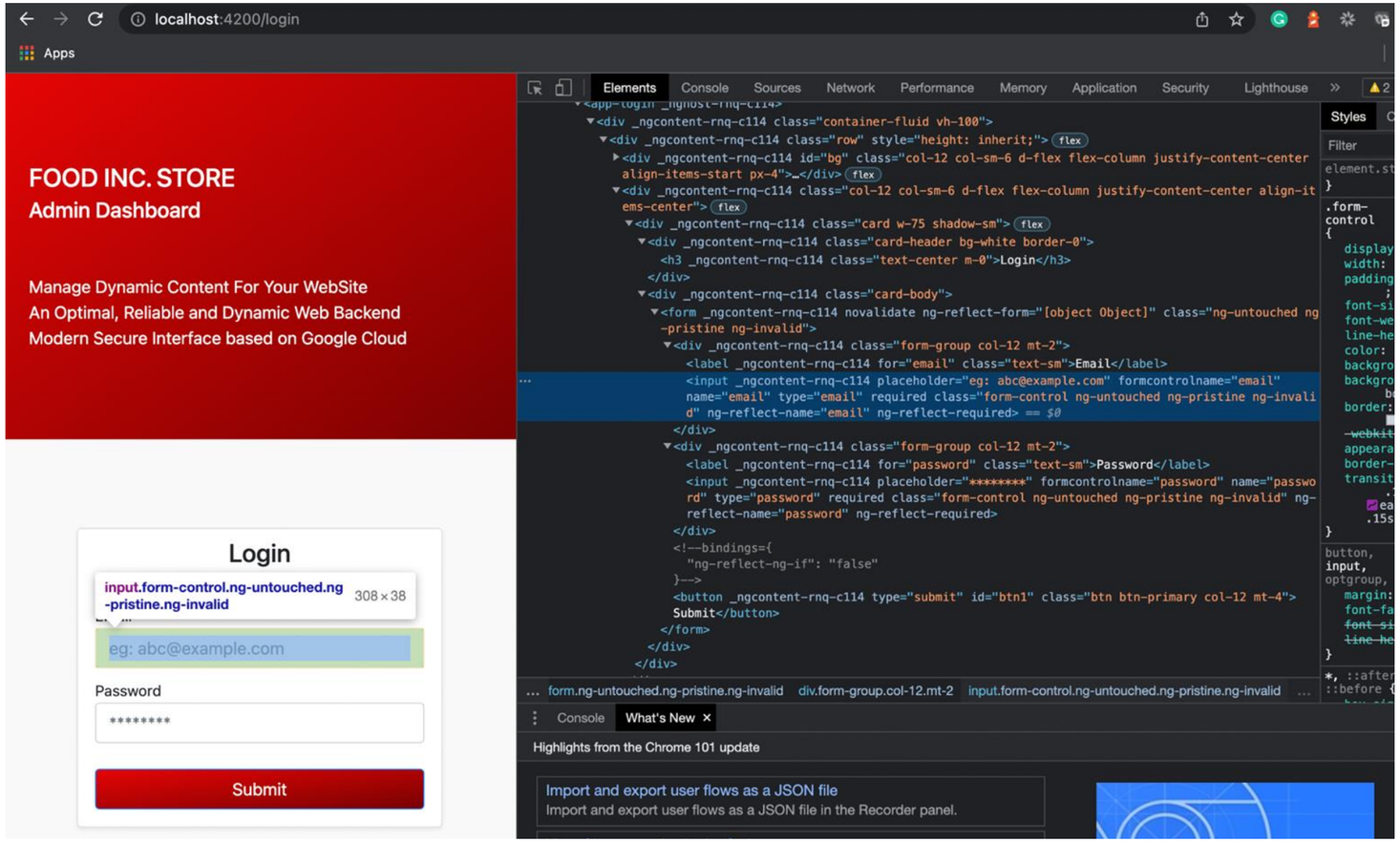


# Write Test Case for Login Page

3. Write the second scenario to test the login with email and password

We will now create a scenario **should login with email and password**

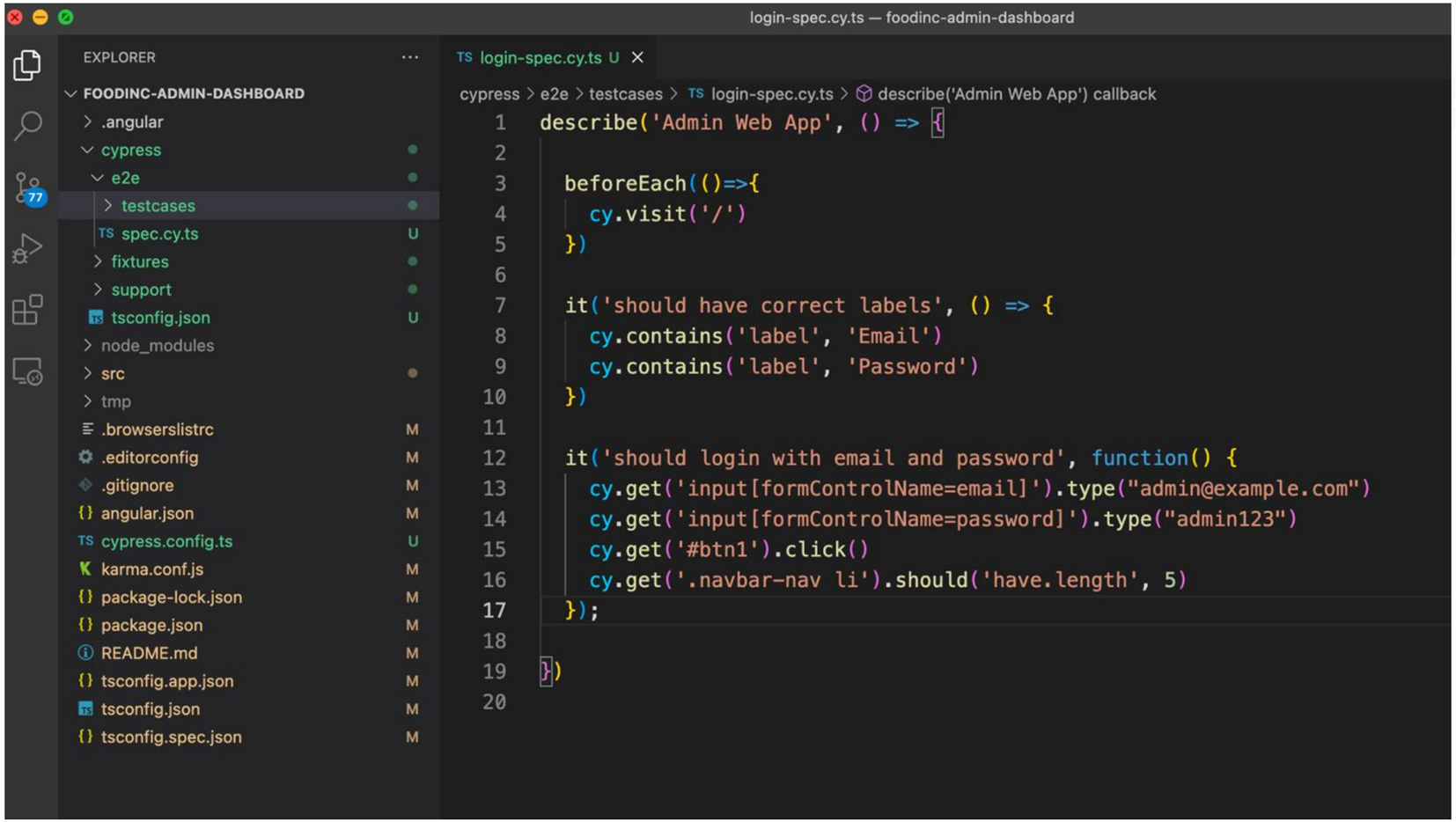
In order to send data to input fields, we will use formControlName Locator. Right click on your web app and open elements to find the formControlName and The Button Id.



# Write Test Case for Login Page

3. Write the second scenario to test the login with email and password

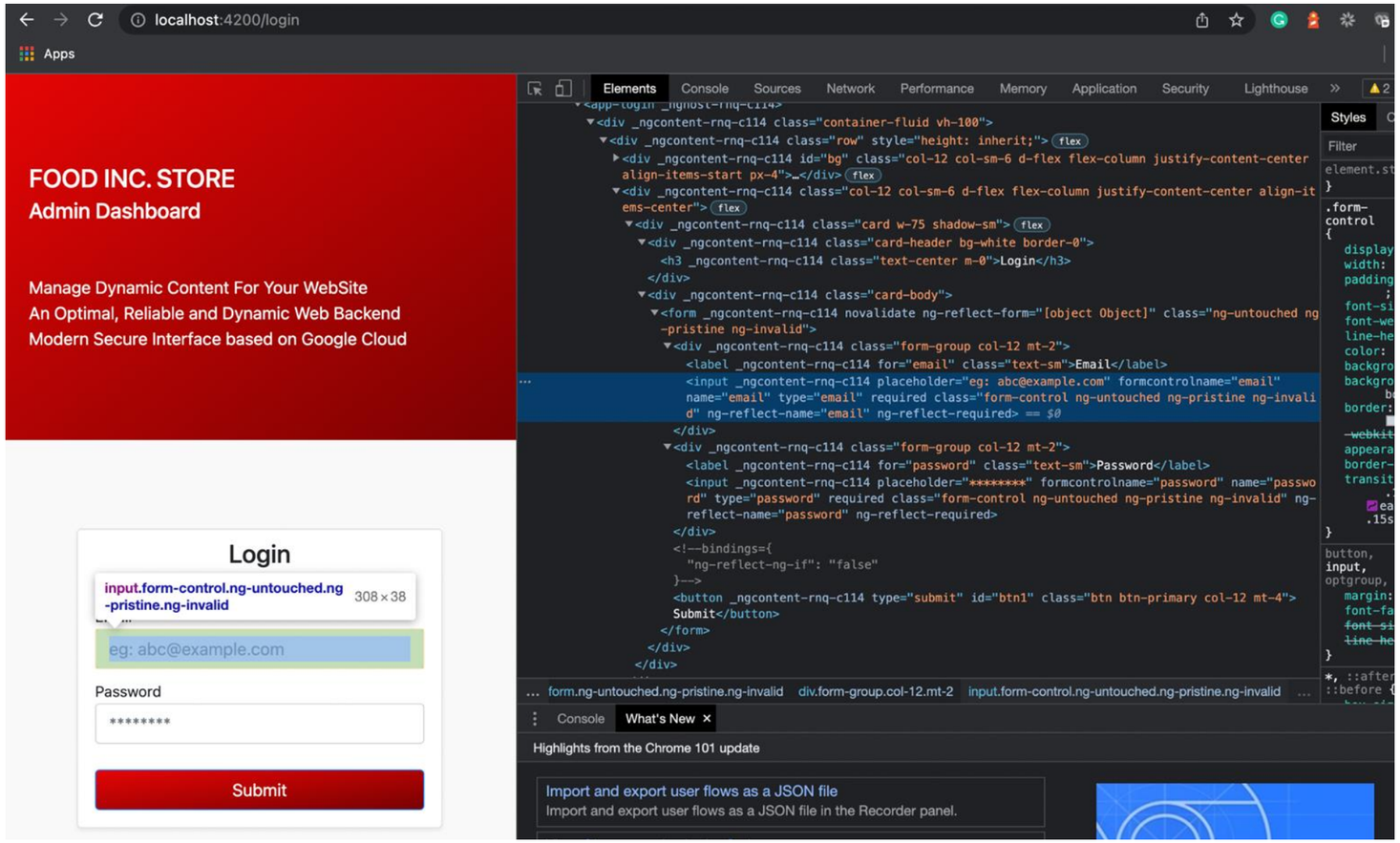
Scenario **should login with email and password**



# Write Test Case for Login Page

4. Write the third scenario to test the login with email and password with redirection

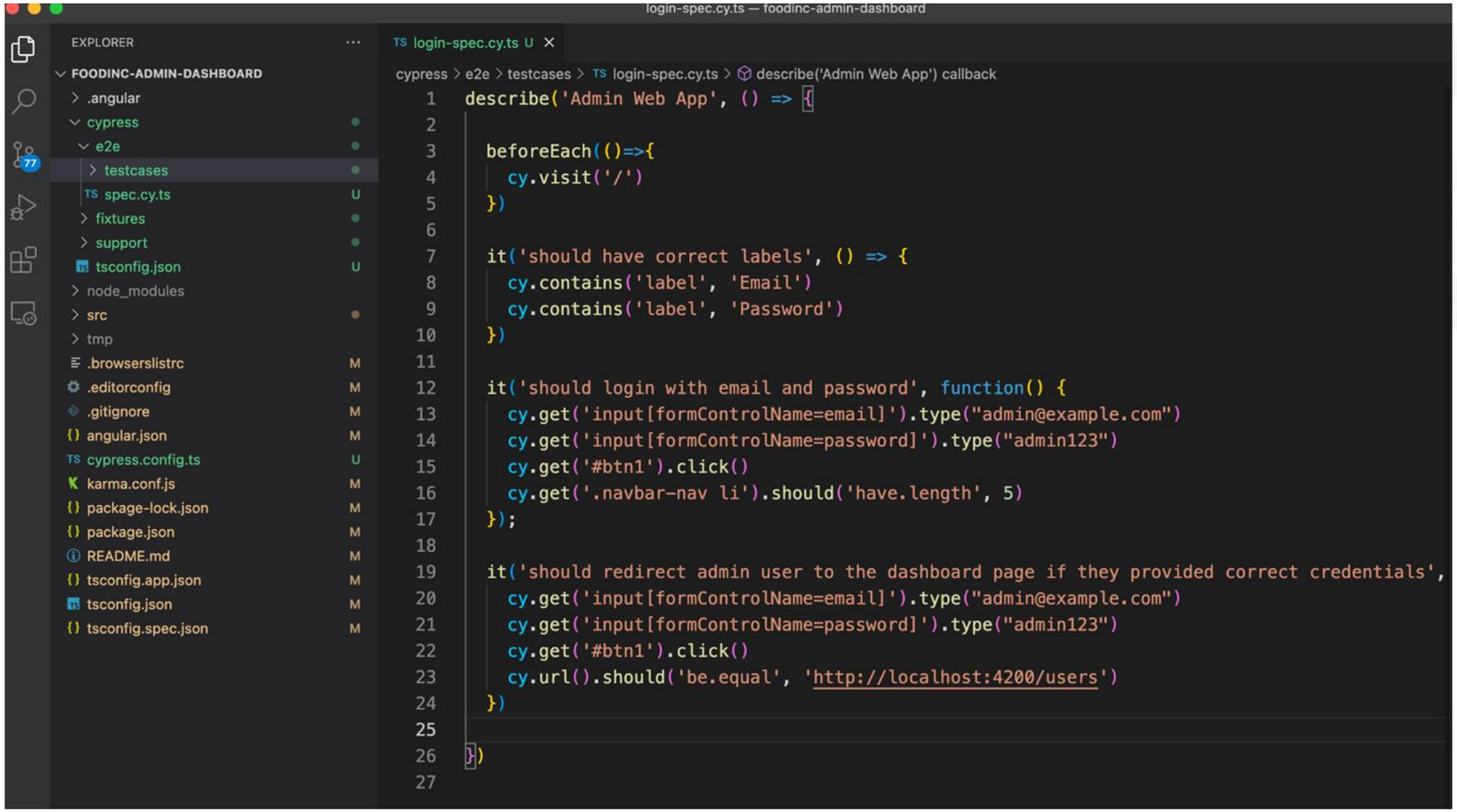
We will now create a scenario **should redirect admin user to dashboard page if they provide correct credentials**. In order to send data to input fields, we will use formControlName Locator. Right click on your web app and open elements to find the formControlName and The Button Id.



# Write Test Case for Login Page

4. Write the third scenario to test the login with email and password with redirection

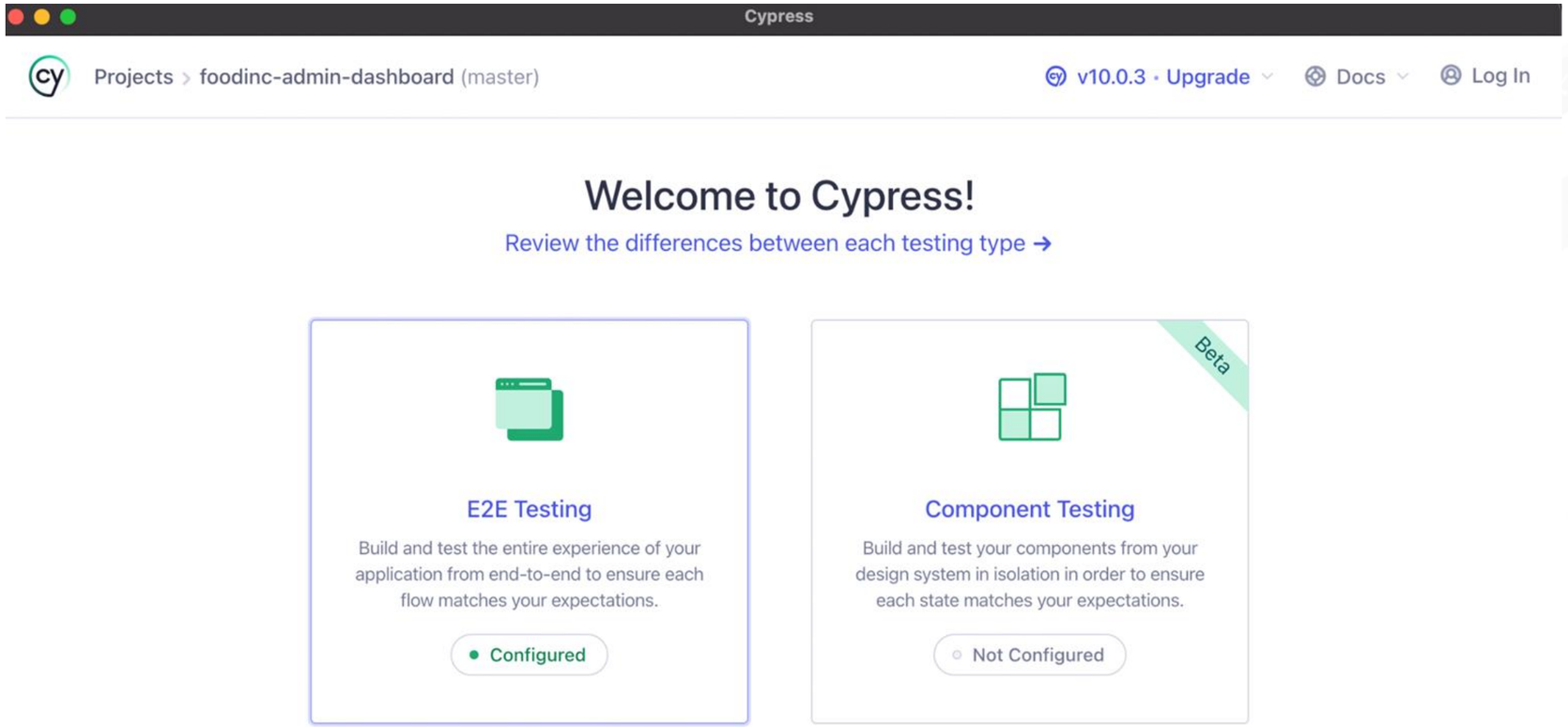
Scenario **should redirect admin user to dashboard page if they provide correct credentials.** Here, we will check if url contains the endpoint or not



# Write Test Case for Login Page

5. Execute the Test Cases

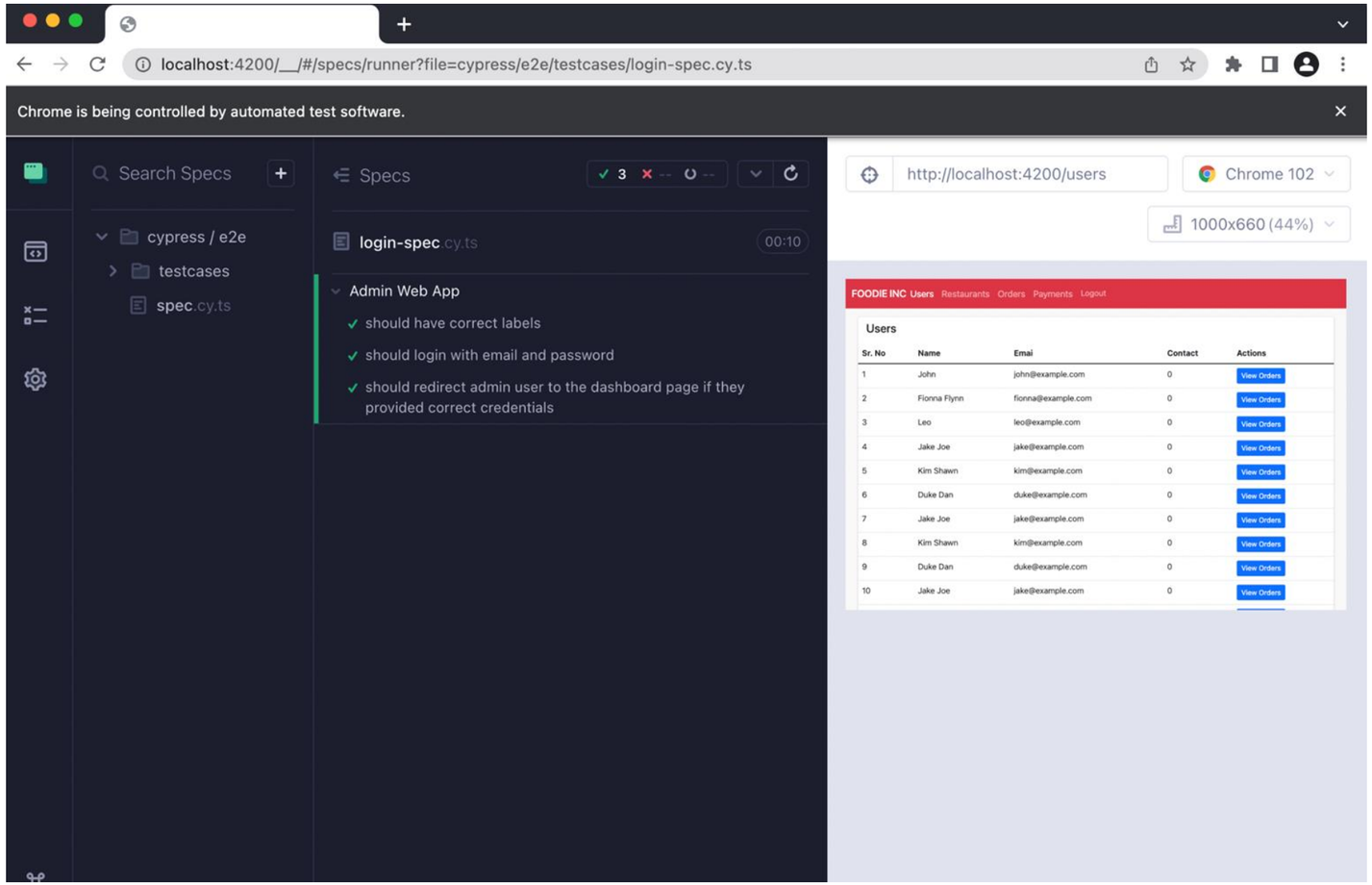
Now, run the command **ng e2e** to launch the cypress launchpad. And choose E2E on the Cypress Launchpad thereafter choose Chrome as browser



# Write Test Case for Login Page

5. Execute the Test Cases

Now, finally let us run the file **login-spec.cy.ts** on the specs section to validate if the testing works fine.



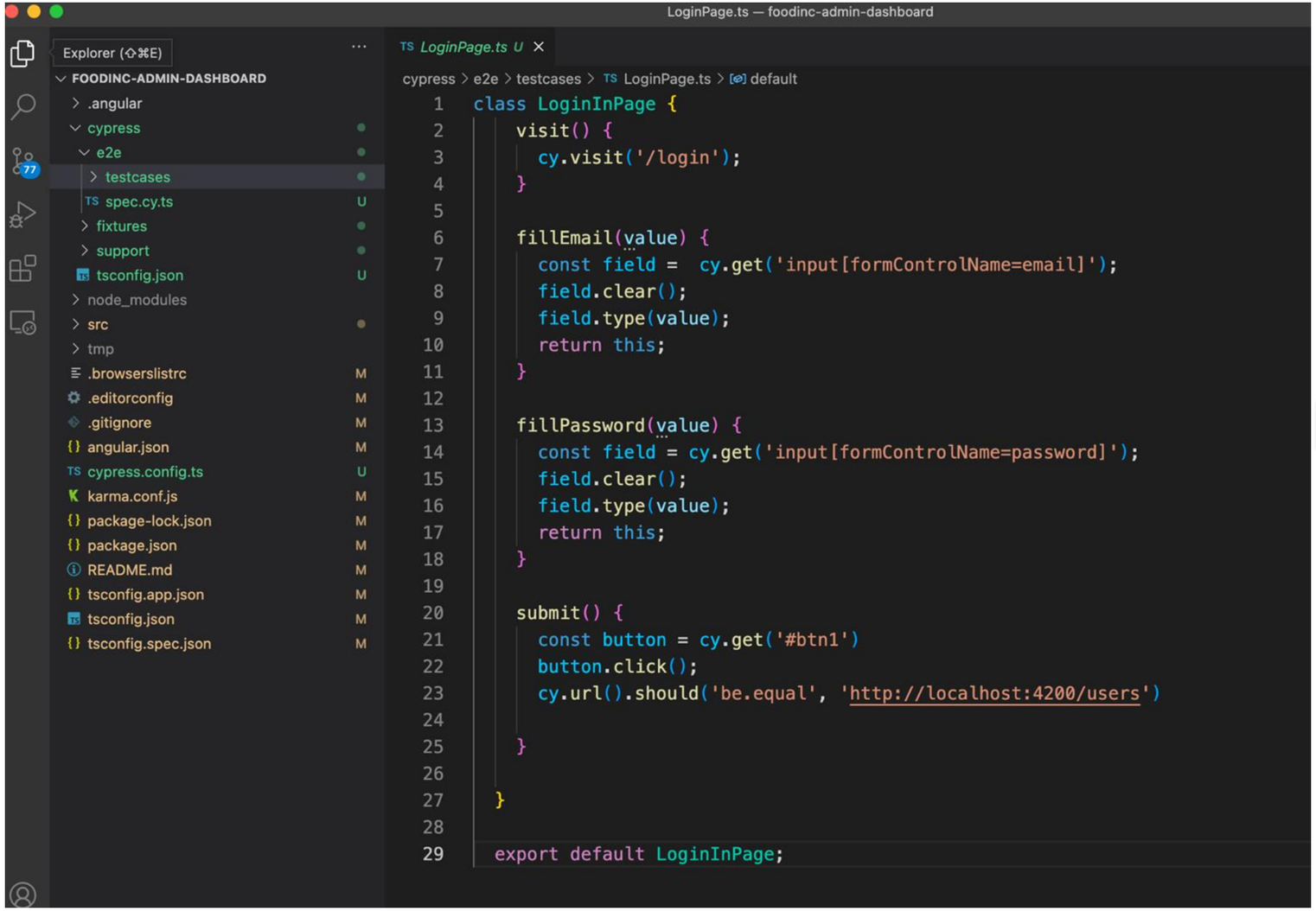
# Page Object Model for Login Page

Now, we will see Page Object Model in order to write test cases to test the login with email and password scenario. For this, we will follow below steps:

1. Create the LoginPage.ts file to define the Page Object Model
2. Create the login-pageobject-spec.cy.ts to define the test case using Page Object Model
3. Execute the Test Case

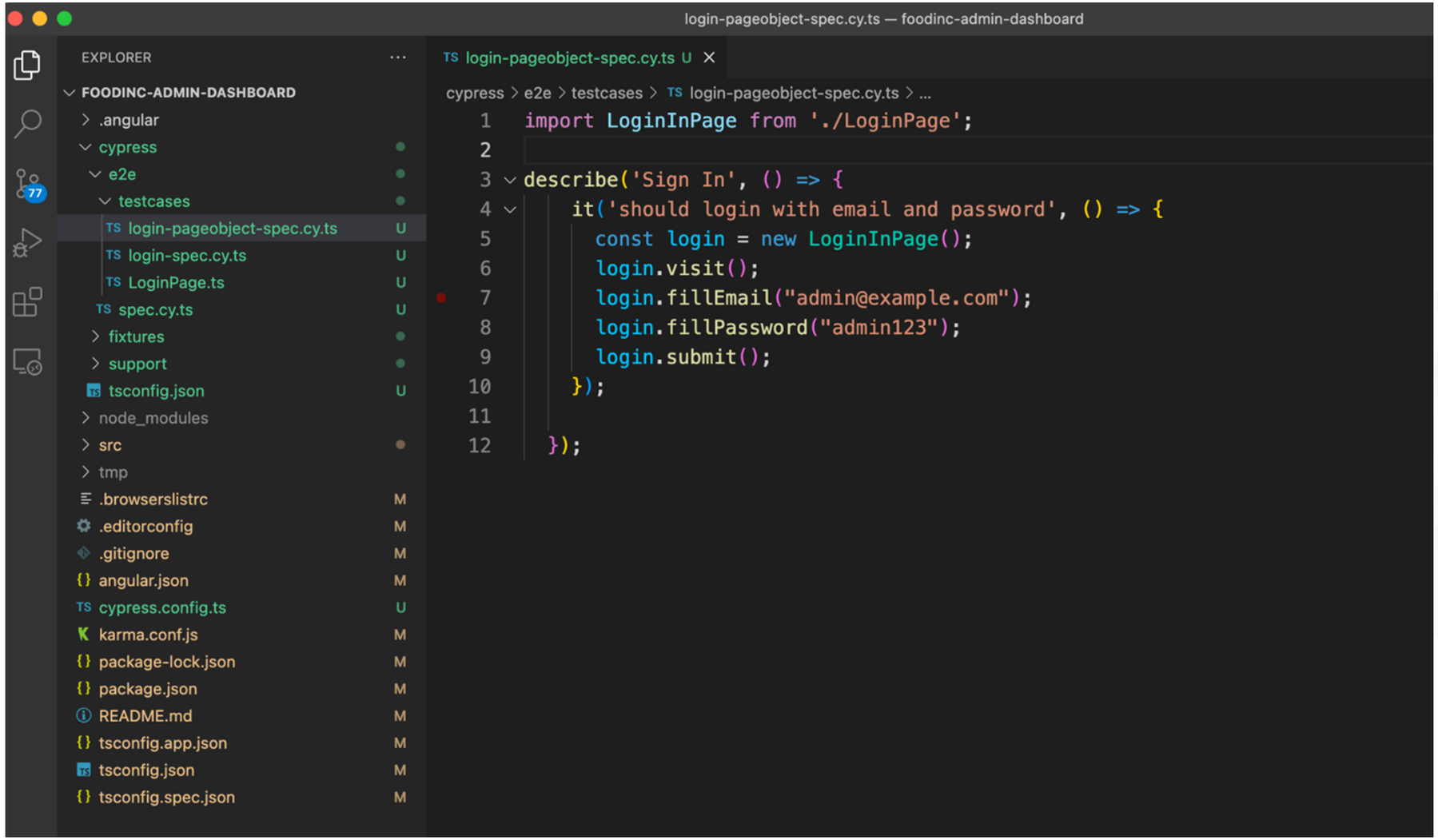
# Page Object Model for Login Page

1. Create the LoginPage.ts file to define the Page Object Model in testcases directory



# Page Object Model for Login Page

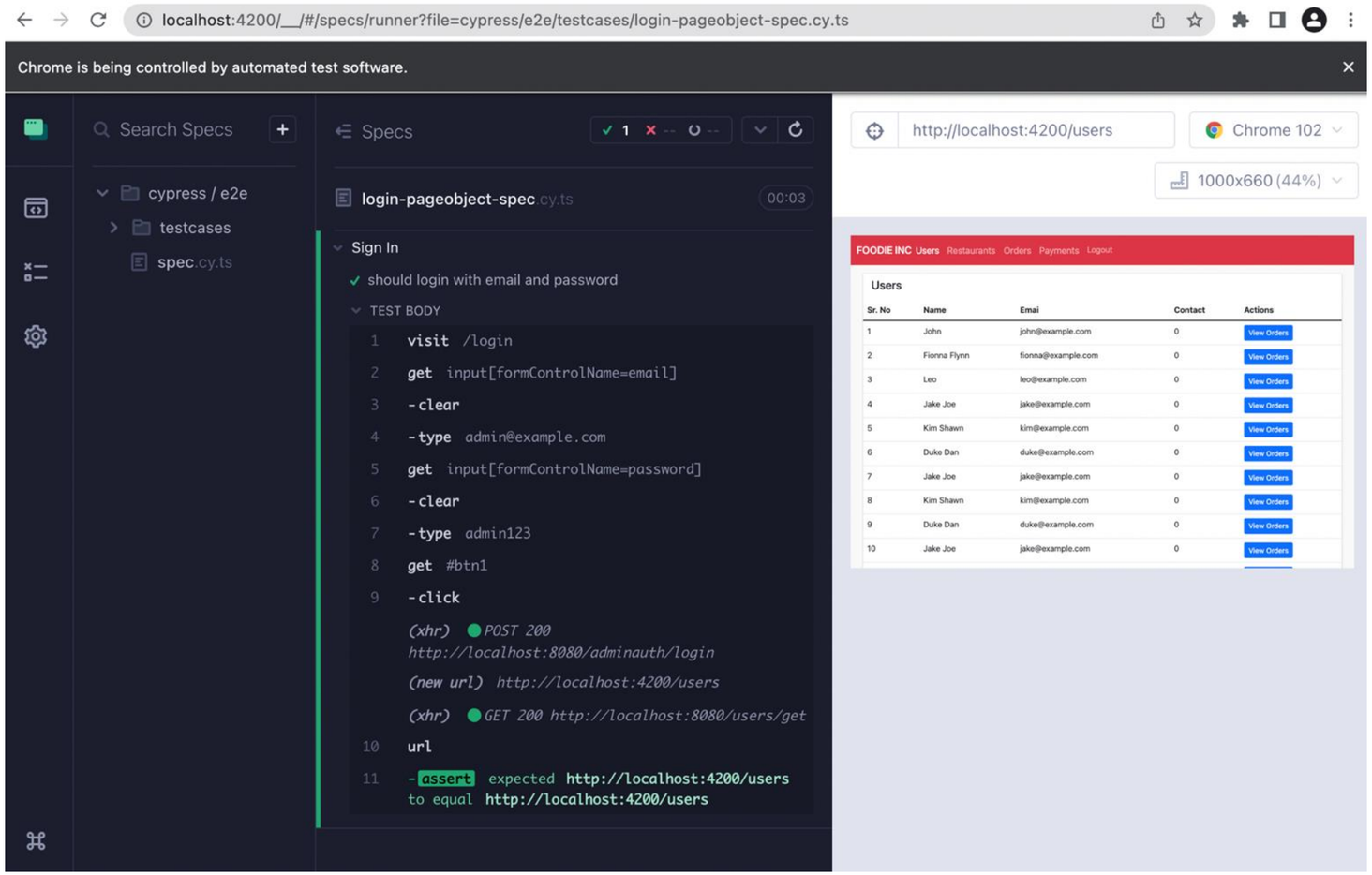
2. Create the login-pageobject-spec.cy.ts to define the test case using Page Object Model



# Page Object Model for Login Page

3. Execute the Test Case

Now, finally let us run the file **login-pageobject-spec.cy.ts** on the specs section to validate if the testing works fine.



# Cypress Mochawesome Reporter

You can work with several reporting tools available in the market for cypress. We will configure cypress-mochawesome-reporter

You need to initially install the library.

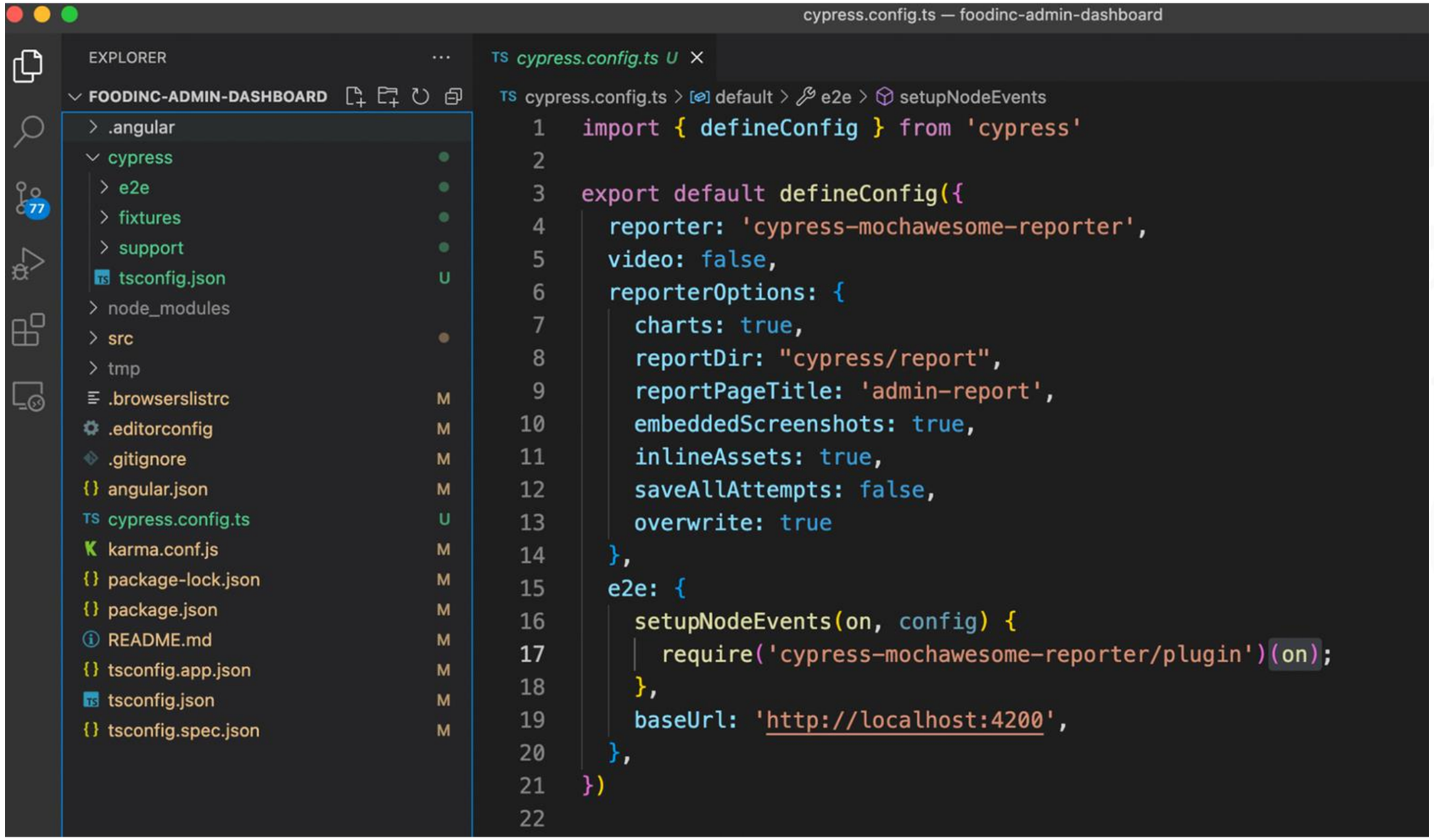
Use the below command:

**npm i --save-dev cypress-mochawesome-reporter**

You can also add the flag --legacy-peer-deps in case you find some error

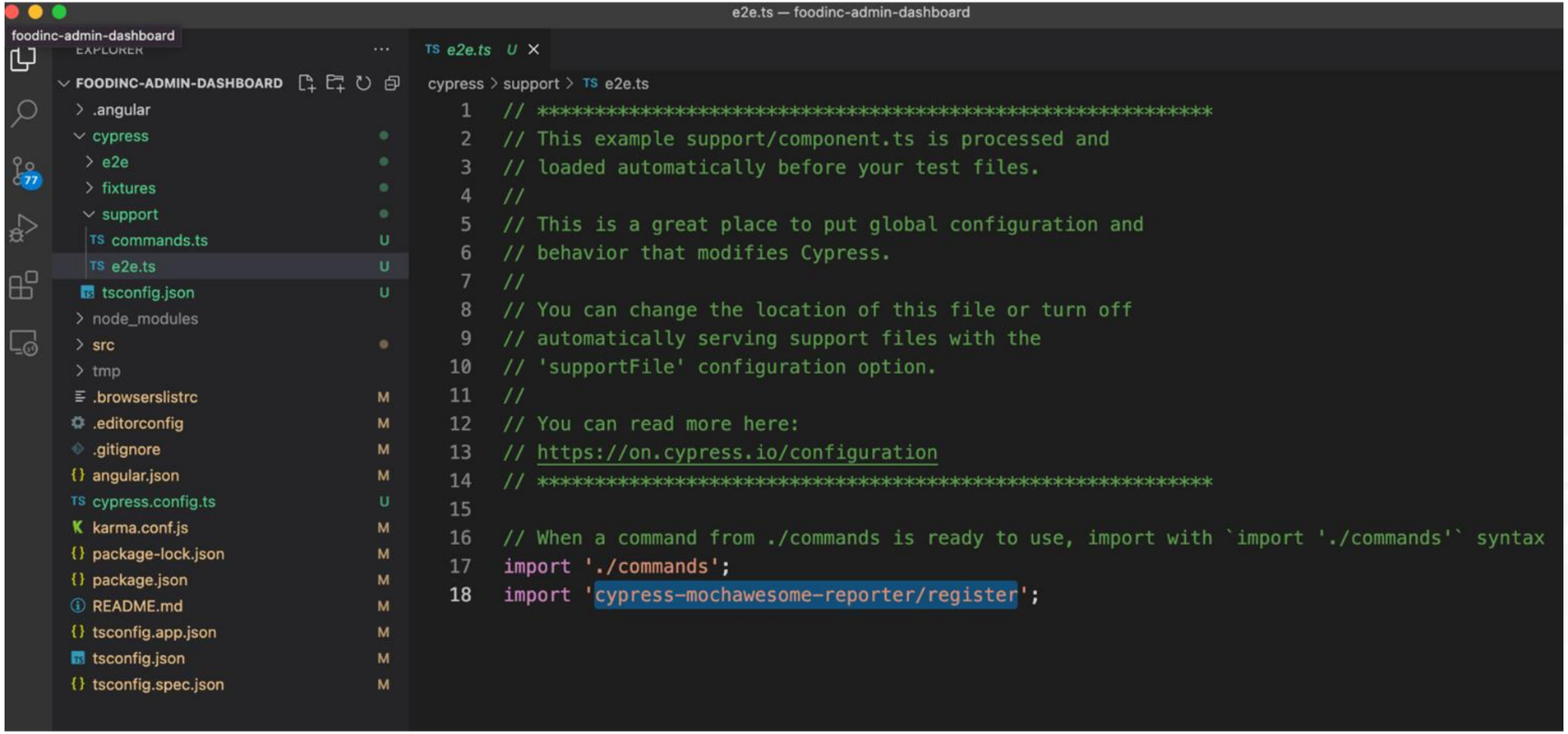
# Cypress Mochawesome Reporter

Now, configure reporter tool in your cypress.config.ts file. Add reporter options in the configuration as per your need.



# Cypress Mochawesome Reporter

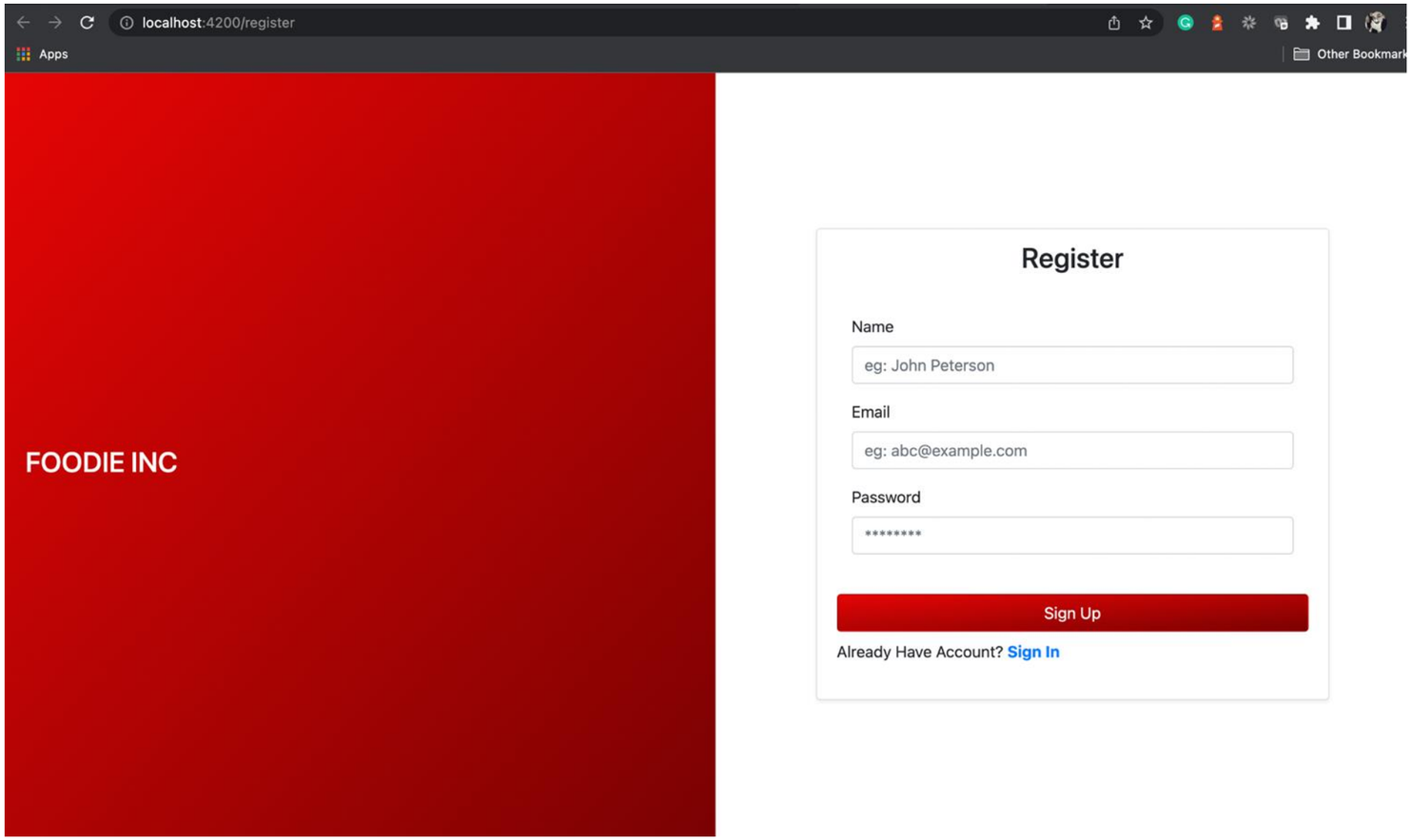
Navigate to the cypress/support directory and import **cypress-mochawesome-reporter/register**



# Task 2: Test the Angular End User Web Pages with Protractor

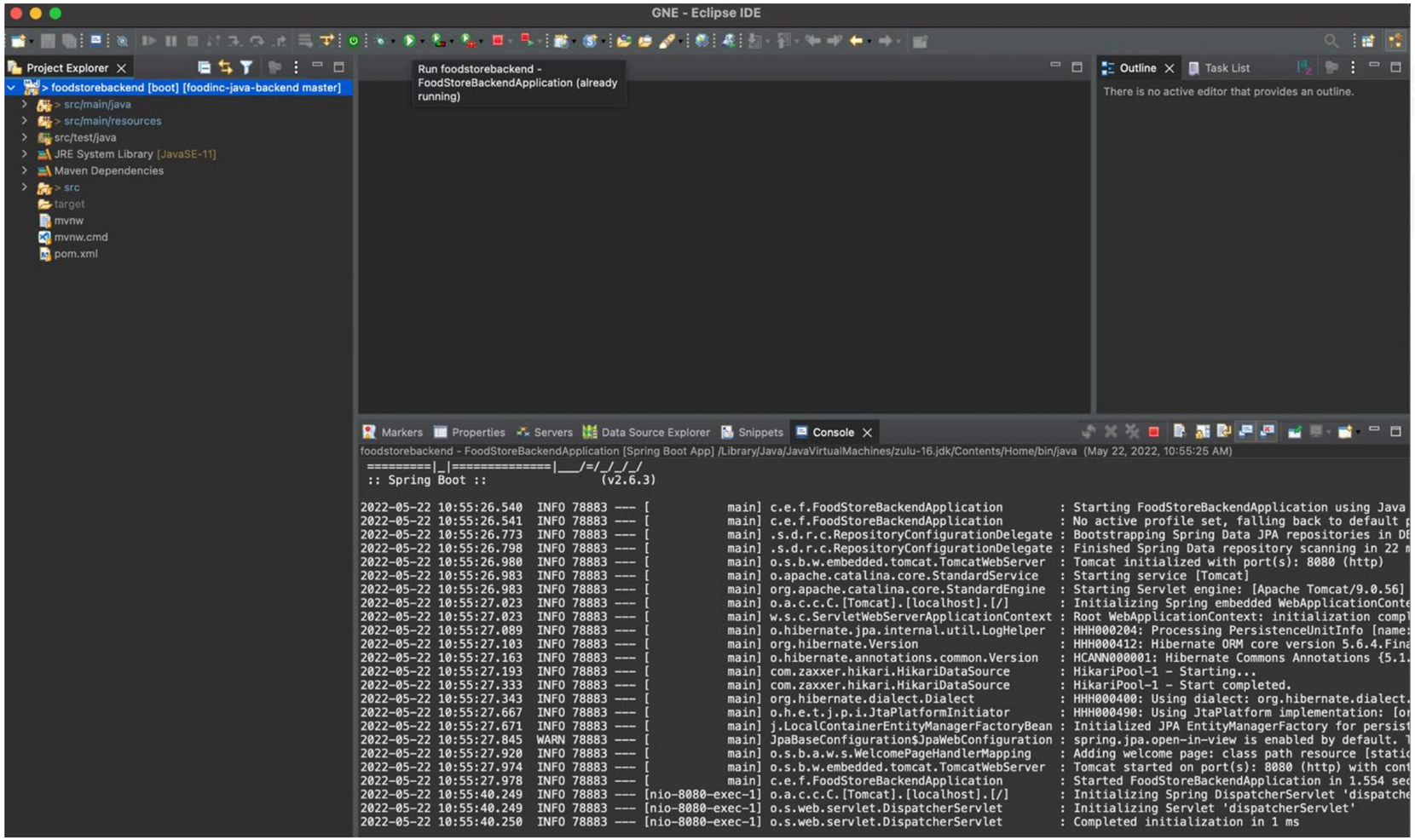
# Build and execute End User Web App Project

Your project is up and running fine on localhost port 4200, with initial UI screen to login. Click on Create Account Link and you will be navigated to register web page.



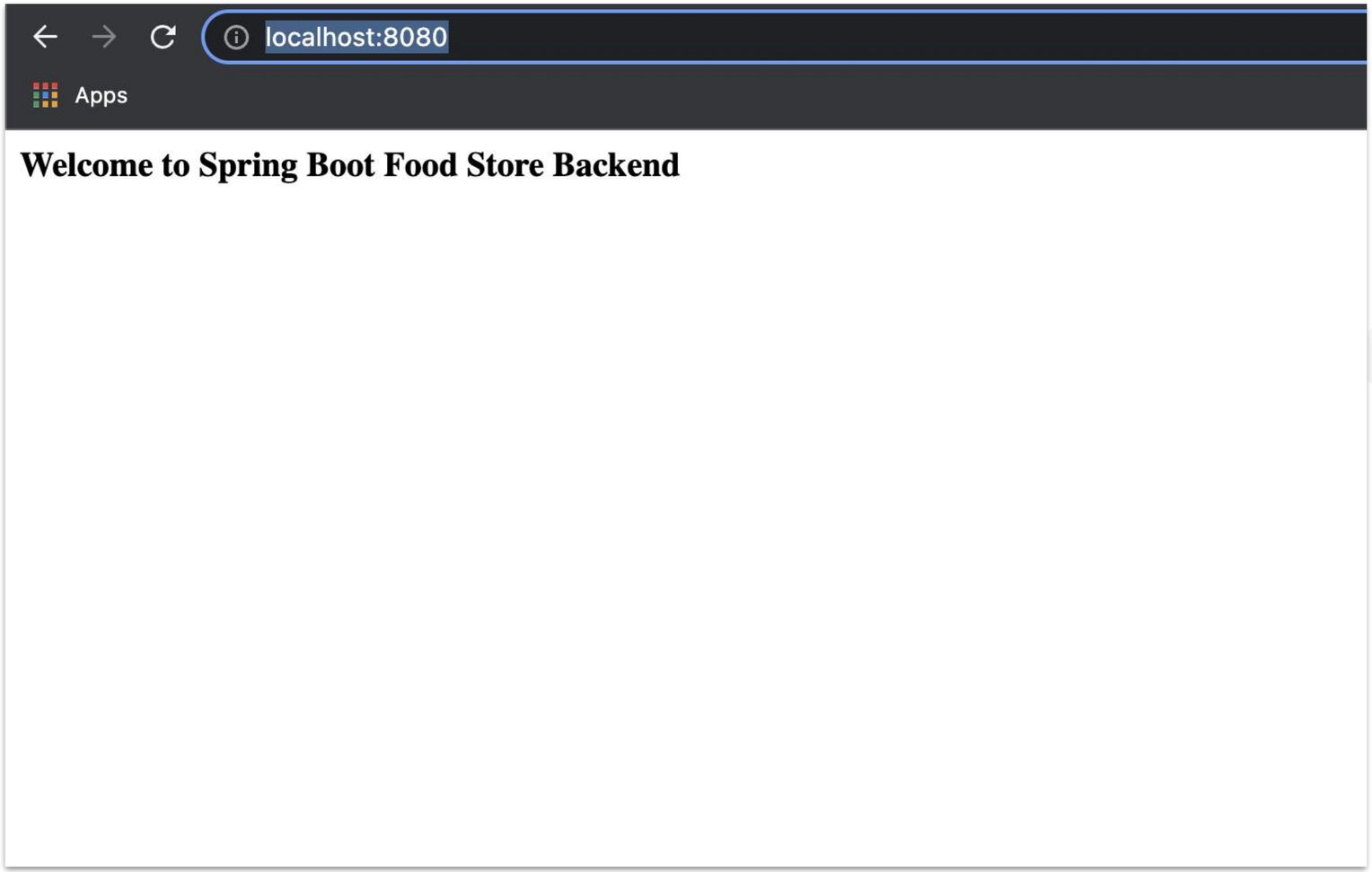
# Build and execute Java Backend Project

For Angular Apps to function with database, java backend should be running. Run Your project from Eclipse EE or You can create jar file by executing a maven build with goal as **clean package** and run it on your terminal by executing command **java -jar foodStoreBackend-0.0.1-SNAPSHOT.jar**



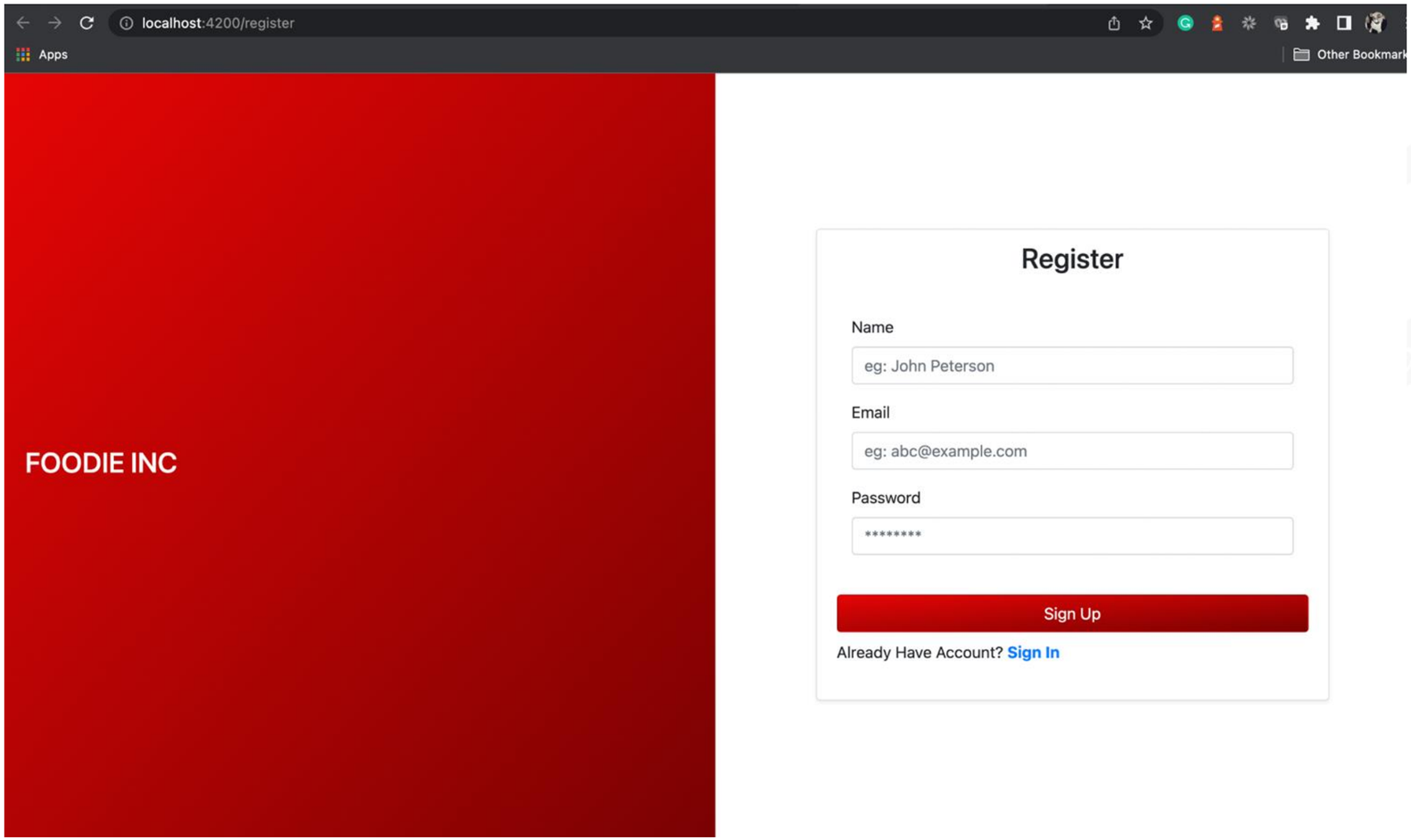
# Check Java Backend Project Status as Running

Open the Web Browser and type http://localhost:8080/ to check if your project is up and running fine



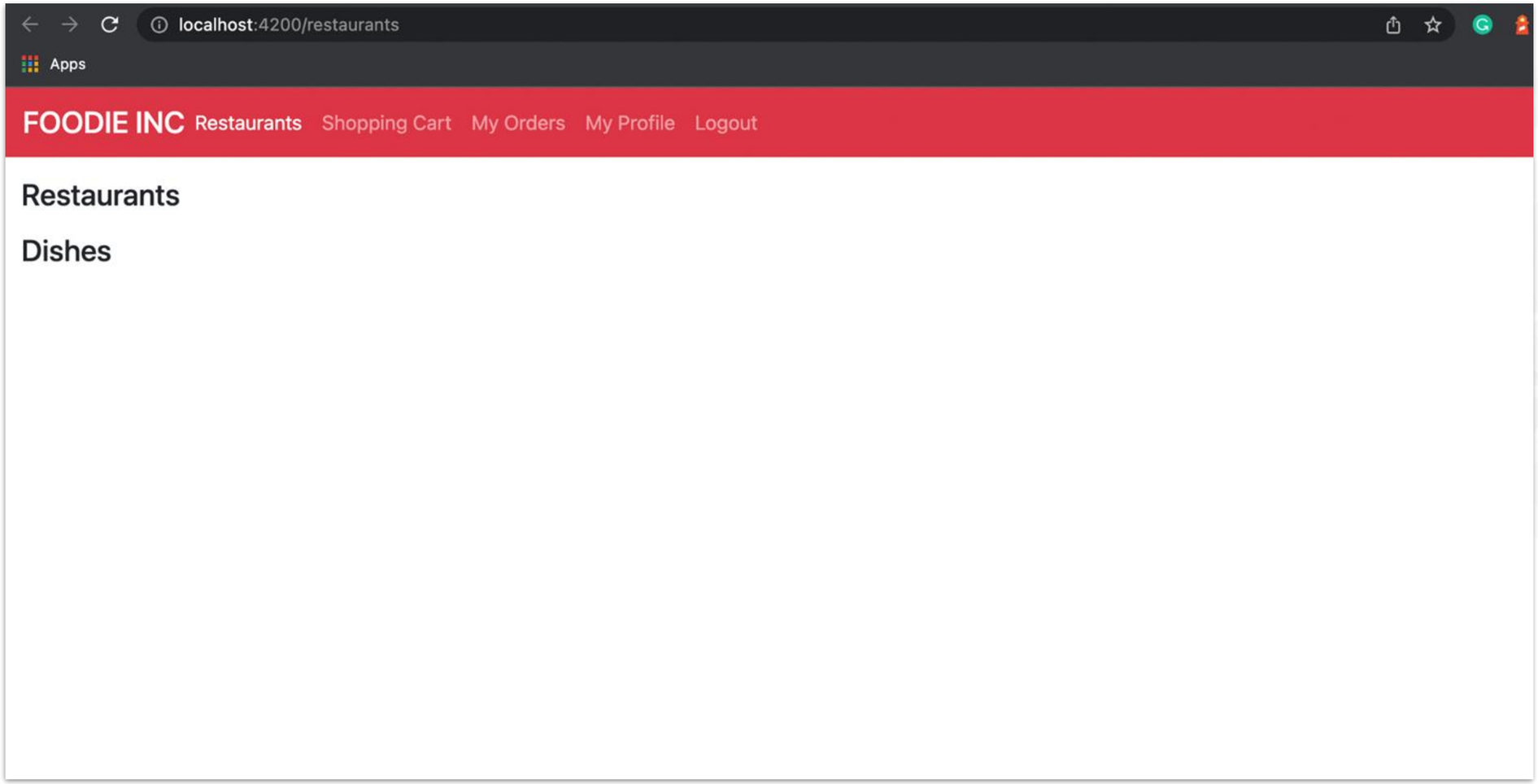
# Register as a new User

Now, enter the credentials to login to the Admin Project and validate the login flow with newly generated email and password.



# Register as a new User

Once, the registration is successful, you will be navigated to home page. As we haven't added any data from Admin for Restaurant or Dishes, you will see an empty web page coming up



# Write Test Case for Register Page

Let us start by writing a simple test for our wnd user web application. We will follow the below steps:

1. Create a directory testcases in your project structure.

2. Create a file register-spec.cy.ts and write the first scenario to test the register form for correct labels

3. Write the second scenario to test the register with name, email and password

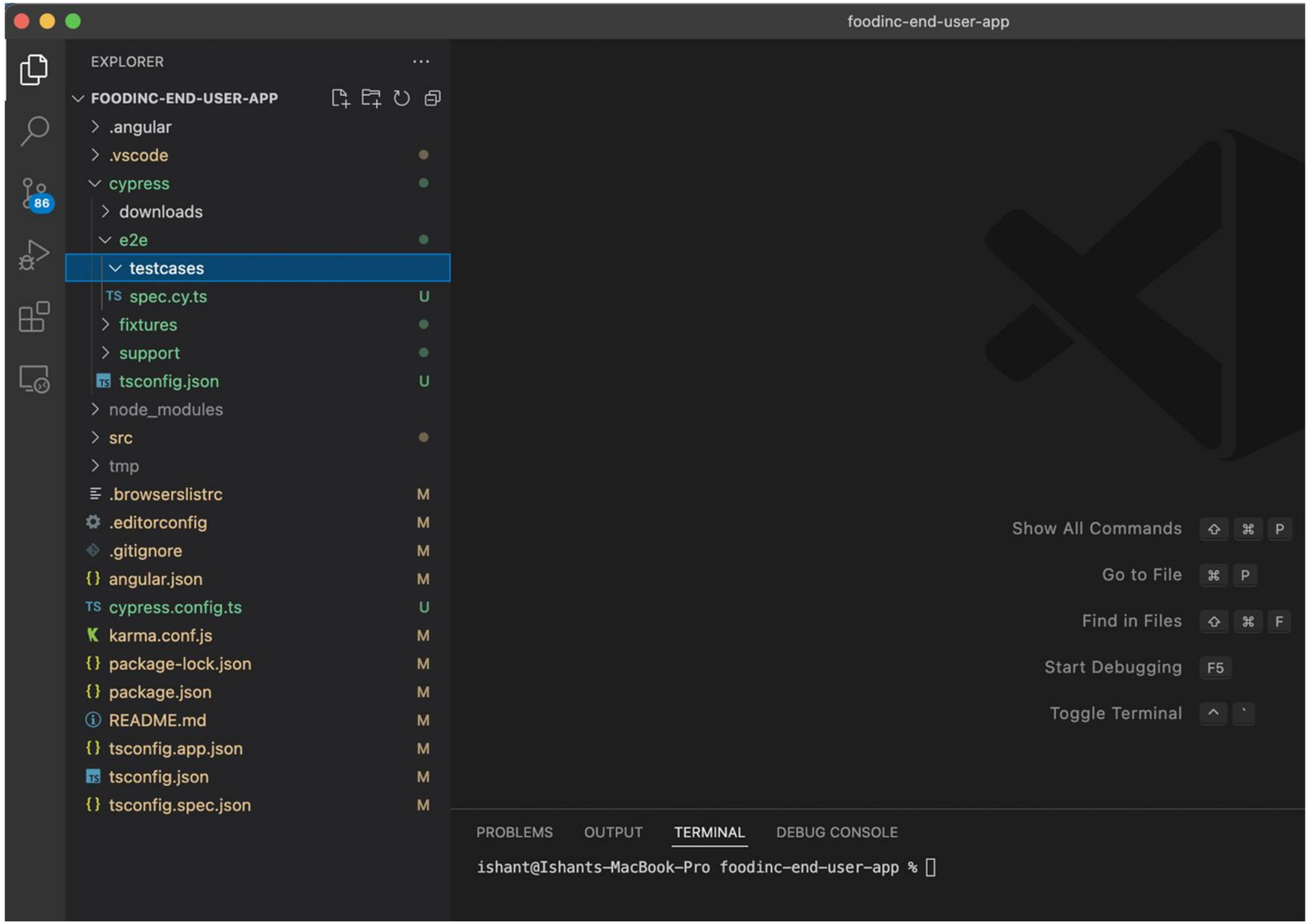
4. Write the third scenario to test the register with name, email and password with redirection

5. Execute the Test Cases

6. Check the Database for User Record

# Write Test Case for Register Page

1. Create a directory testcases in your project structure in your cypress/e2e directory.

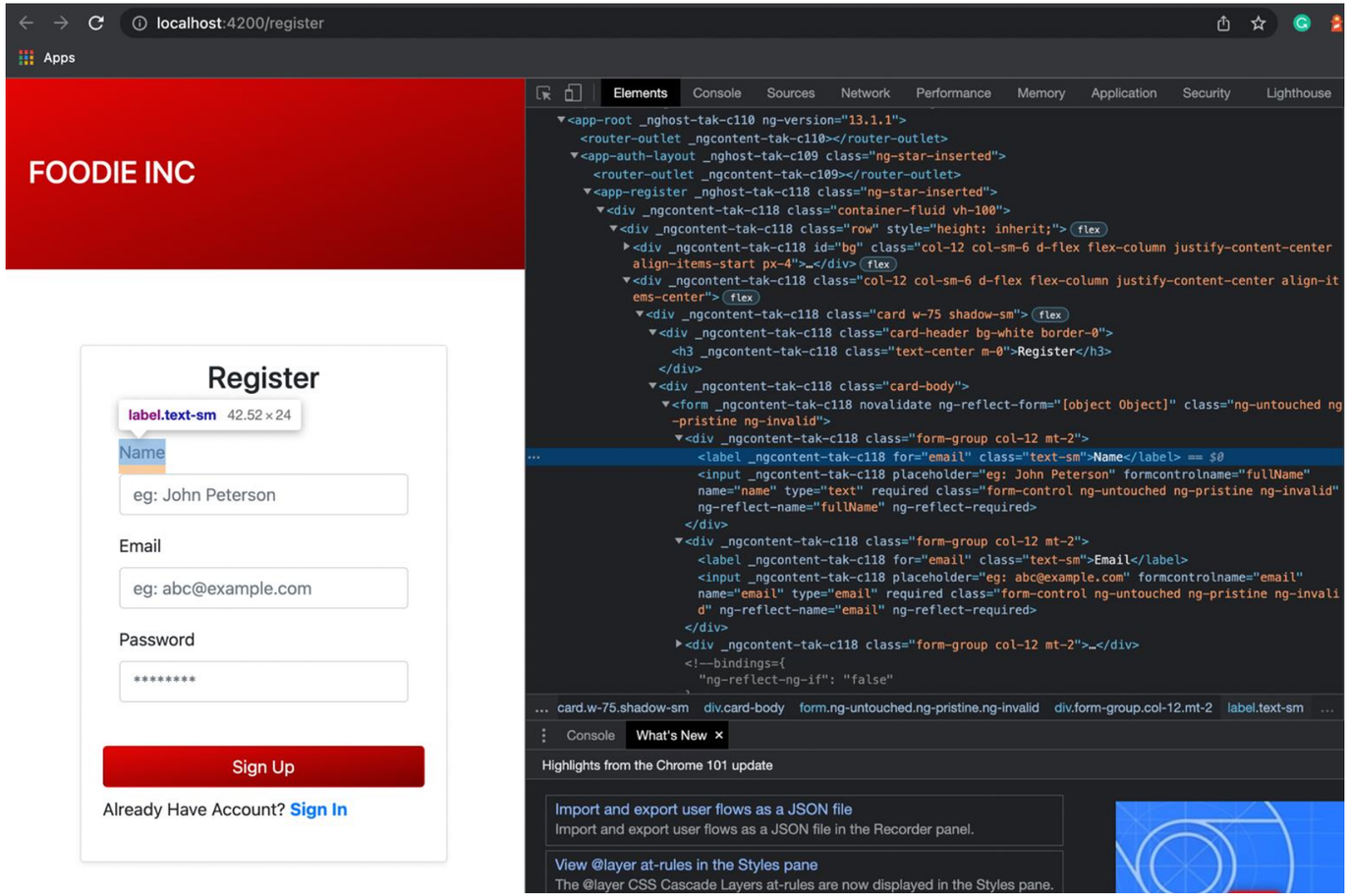


# Write Test Case for Register Page

2. Create a file register-spec.cy.ts and write the first scenario to test the register form for correct labels

We will create a scenario **should have correct labels**

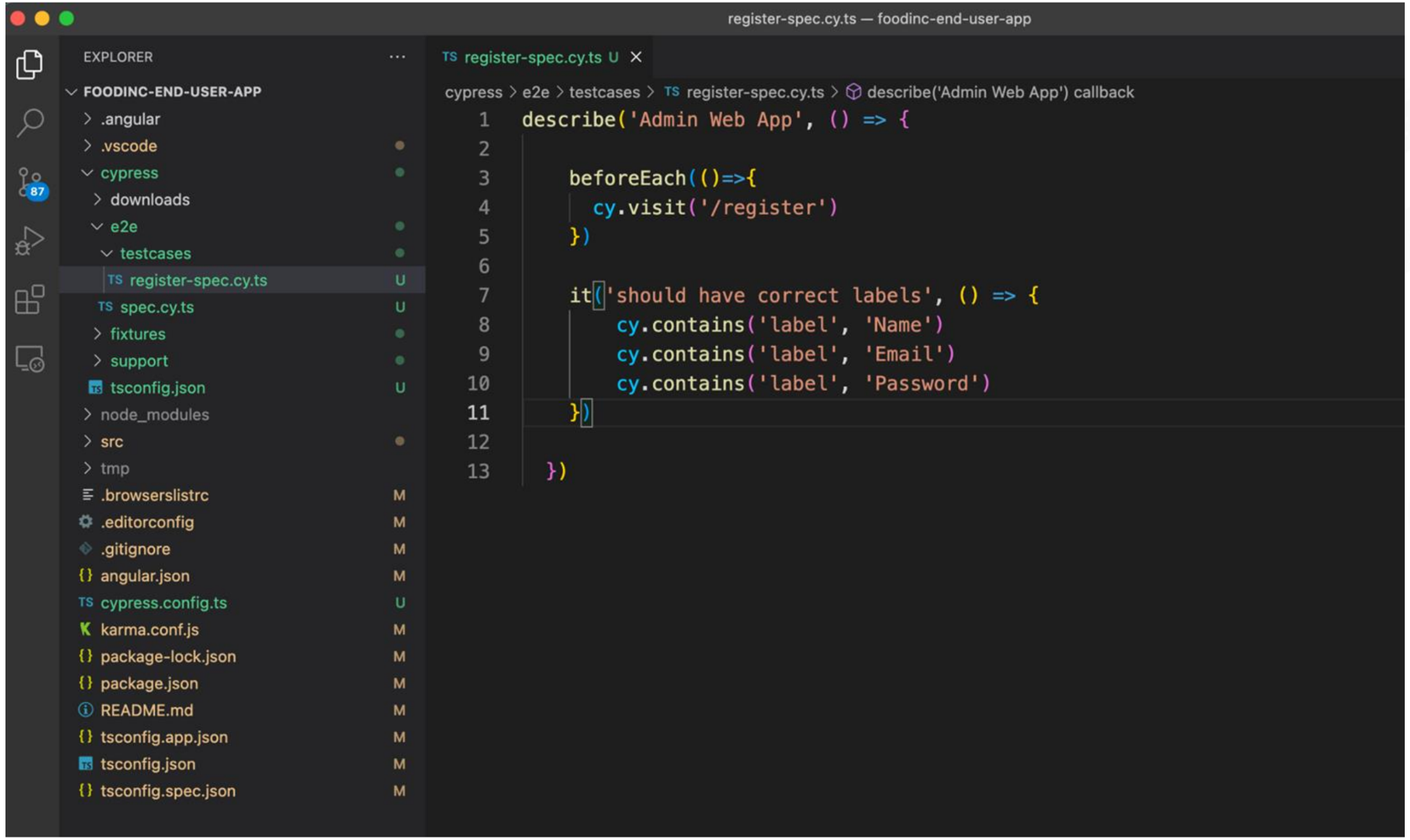
In order to check the labels, we will use CSS Locator. Right click on your web app and open elements to find the css class for the labels.



# Write Test Case for Register Page

2. Create a file register-spec.cy.ts and write the first scenario to test the register form for correct labels

Scenario **should have correct labels**

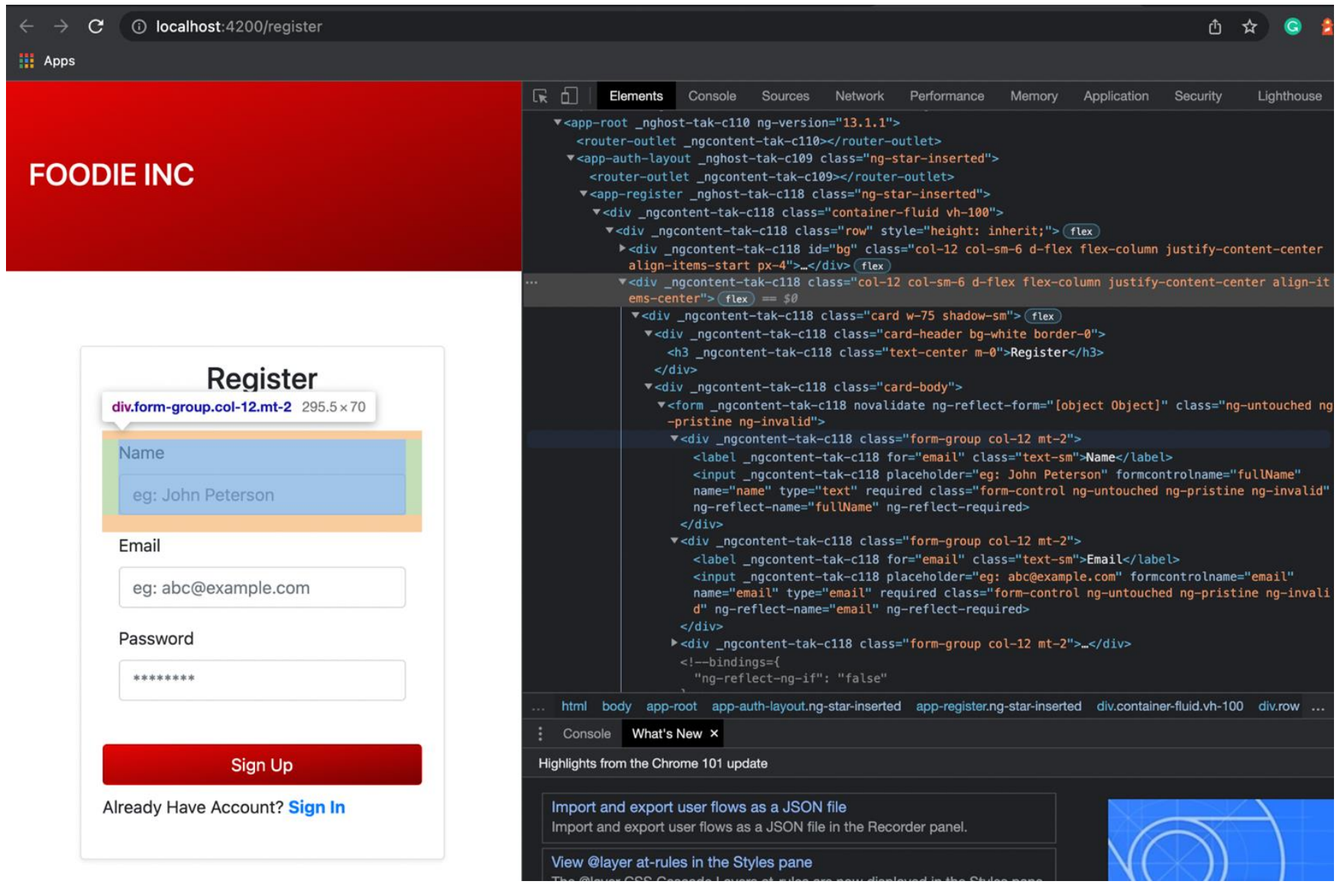


# Write Test Case for Register Page

3. Write the second scenario to test the register with email and password

We will now create a scenario **should register with name email and password**

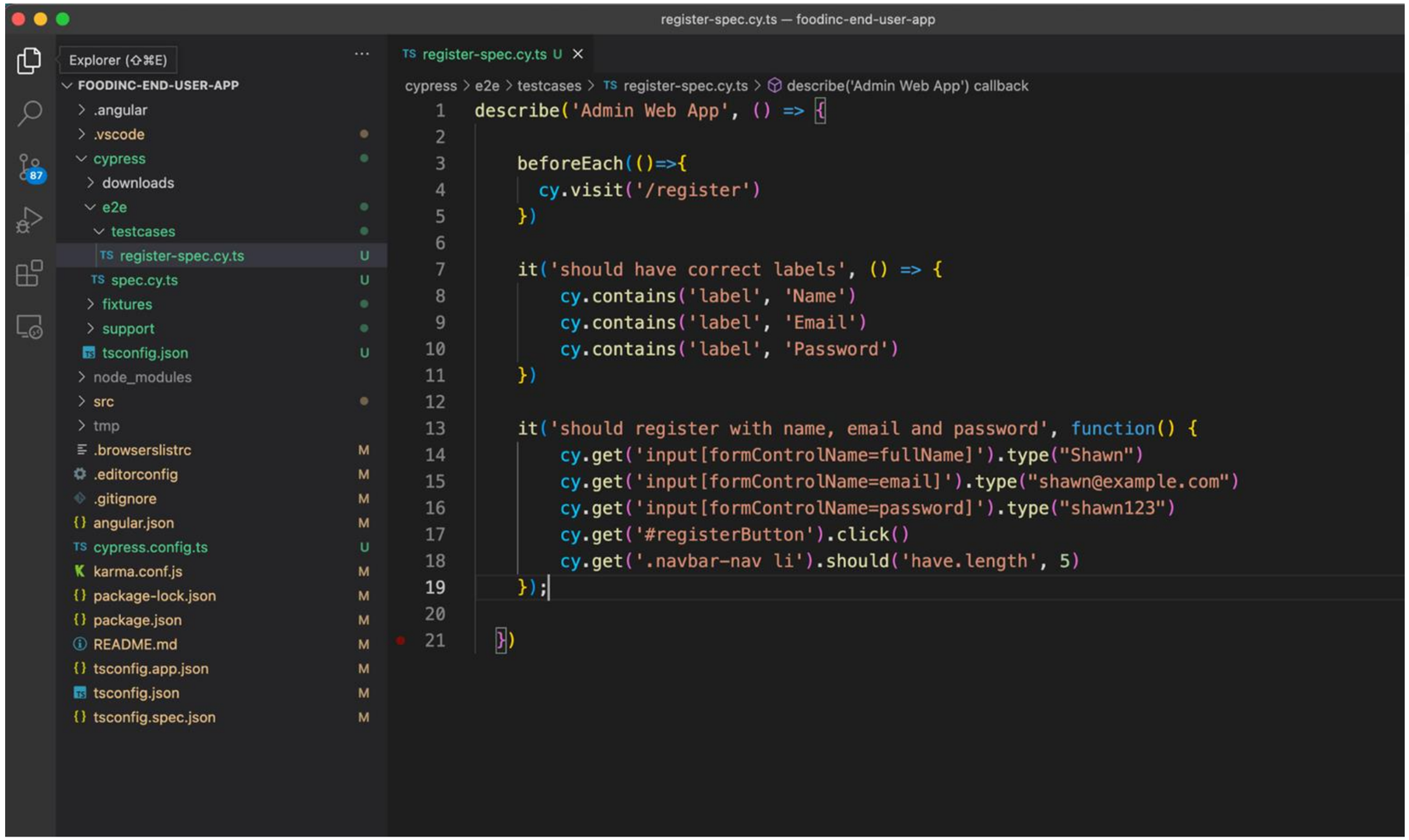
In order to send data to input fields, we will use formControlName Locator. Right click on your web app and open elements to find the formControlName and The Button Id.



# Write Test Case for Register Page

3. Write the second scenario to test the register with name, email and password

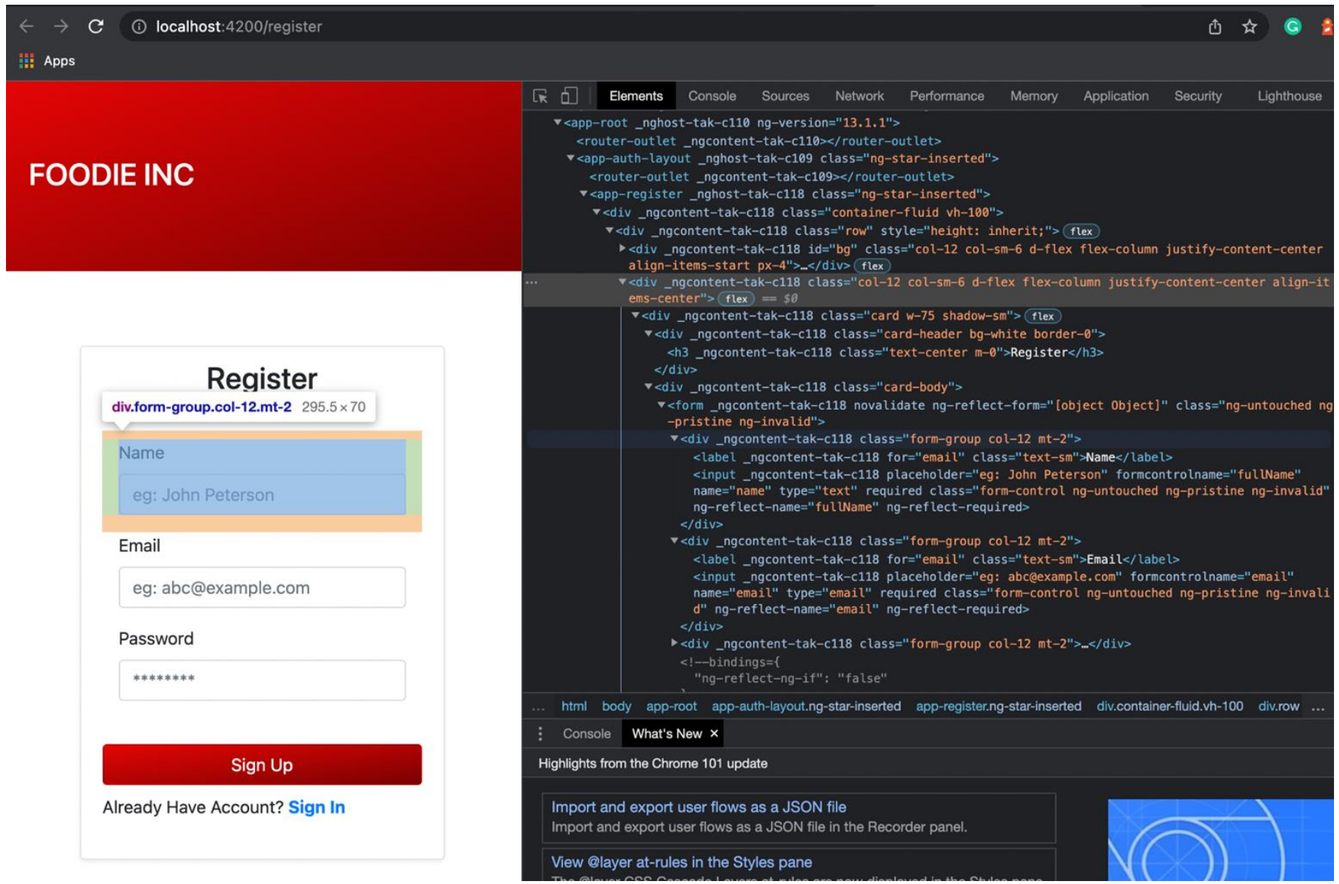
Scenario **should register with name, email and password**



# Write Test Case for Register Page

4. Write the third scenario to test the register with name, email and password with redirection

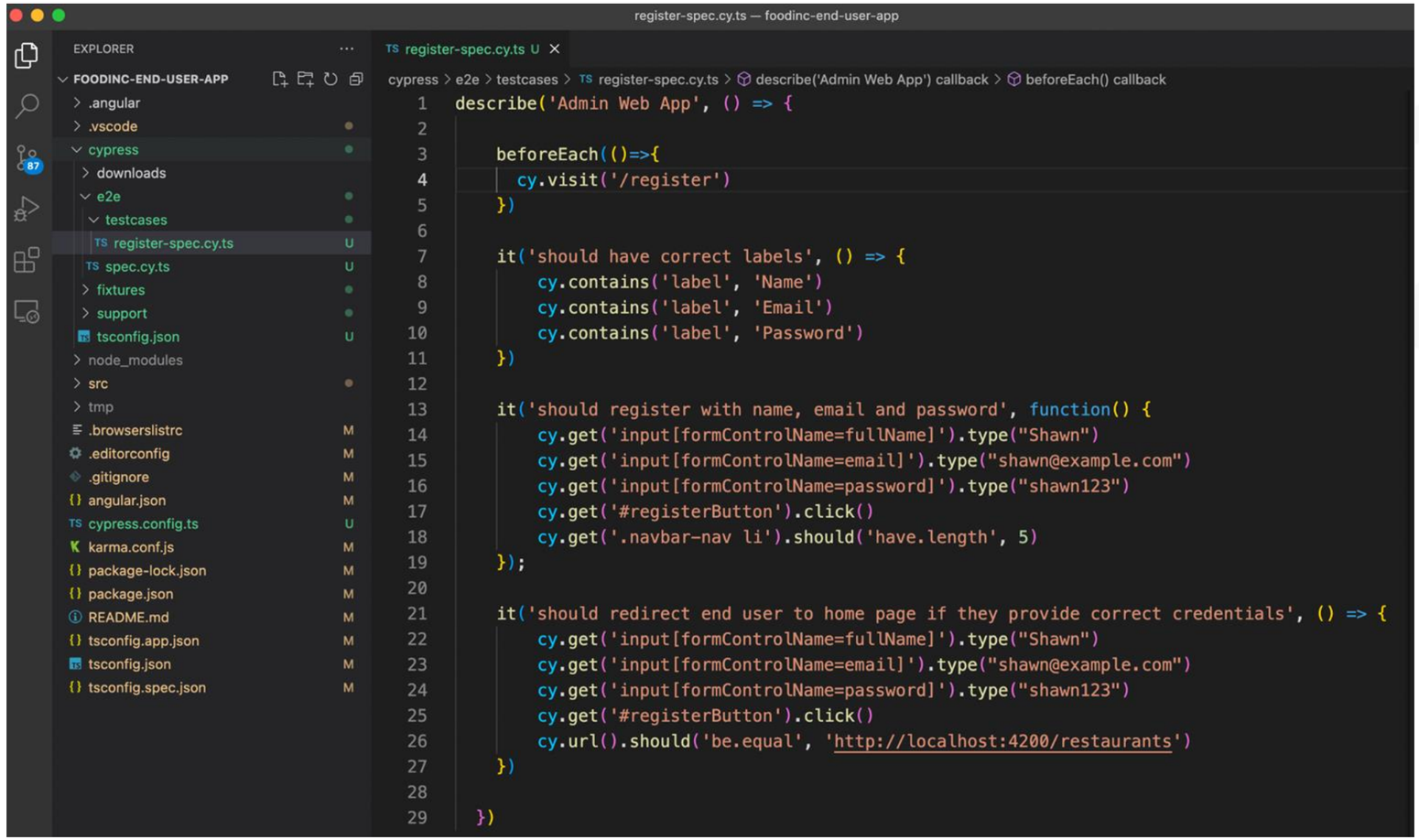
We will now create a scenario **should redirect end user to home page if they provide correct credentials**. In order to send data to input fields, we will use formControlName Locator. Right click on your web app and open elements to find the formControlName and The Button Id.



# Write Test Case for Register Page

4. Write the third scenario to test the register with name, email and password with redirection

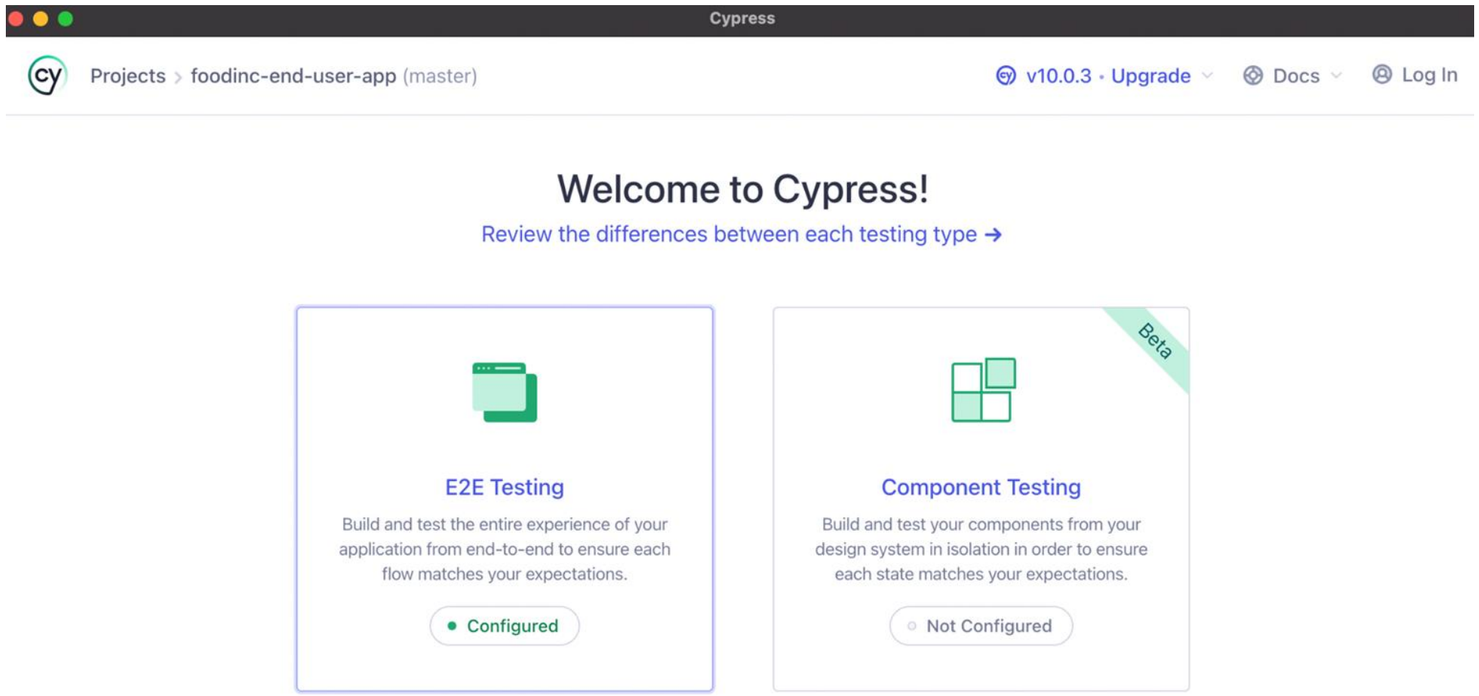
Scenario **should redirect end user to home page if they provide correct credentials** Here, we will use ExpectedConditions to check if url contains the endpoint or not



# Write Test Case for Register Page

5. Execute the Test Cases

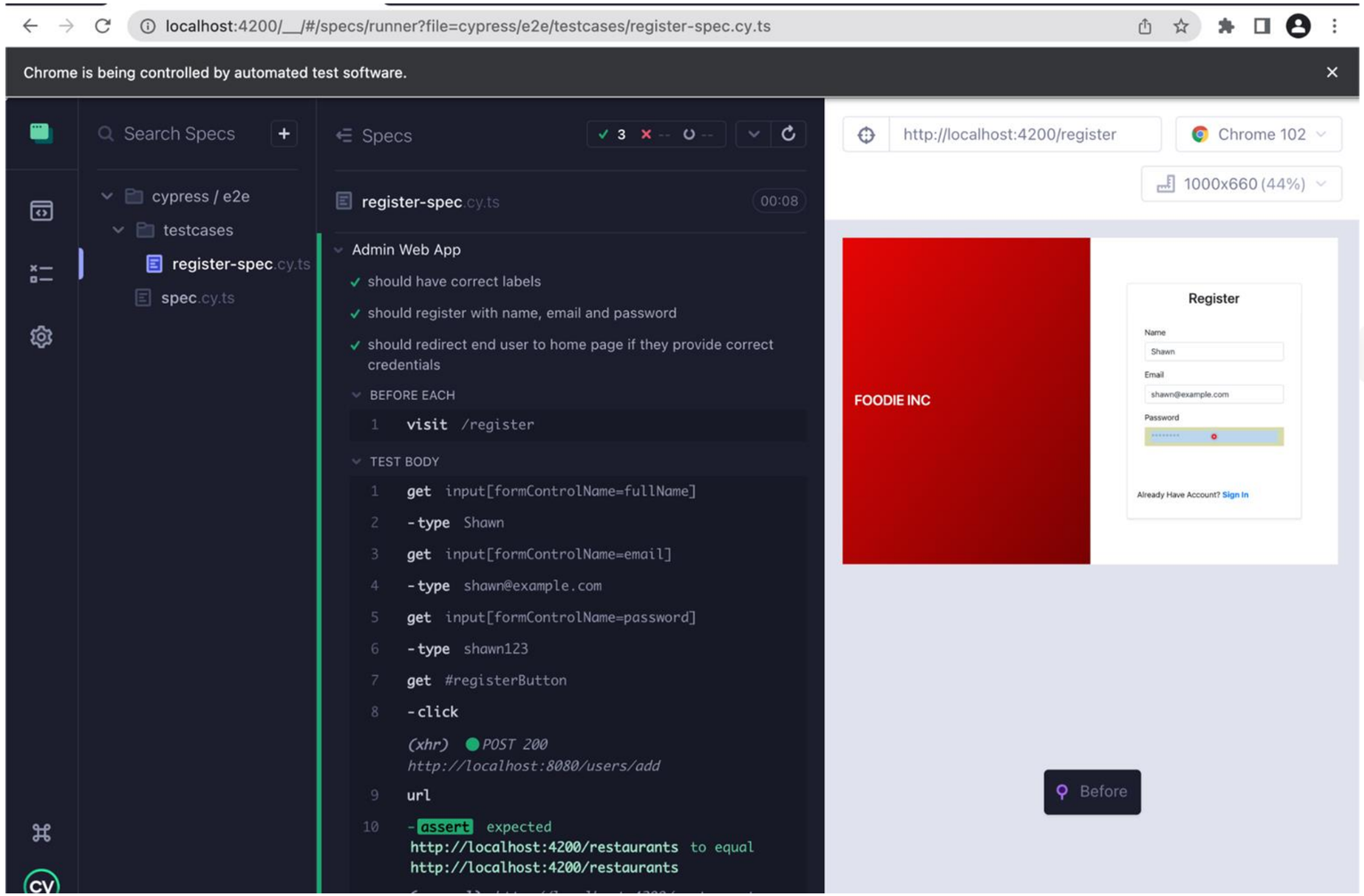
Now, run the command **ng e2e** to launch the cypress launchpad. And choose E2E on the Cypress Launchpad thereafter choose Chrome as browser



# Write Test Case for Register Page

5. Execute the Test Cases

Now, finally let us run the file **register-spec.cy.ts** on the specs section to validate if the testing works fine.



# Write Test Case for Register Page

6. Check the Database for User Record

In order to validate if the User has been created in database, you can login to MySQL and check users table for the added record

```
mysql> select * from users;
+---------+----------------------------+---------+--------------------+-------------+-------+-----------+
| user_id | added_on                   | contact | email              | full_name   | image | password  |
+---------+----------------------------+---------+--------------------+-------------+-------+-----------+
|       1 | 2022-05-23 15:41:06.125000 |       0 | john@example.com   | John        | null  | john123   |
|       2 | 2022-05-23 16:56:30.990000 |       0 | fionna@example.com | Fionna Flynn| null  | fionna123 |
|       3 | 2022-05-23 17:01:00.306000 |       0 | leo@example.com    | Leo         | null  | leo123    |
|       4 | 2022-05-23 17:04:41.143000 |       0 | jake@example.com   | Jake Joe    | null  | jake123   |
|       5 | 2022-05-23 17:04:46.058000 |       0 | kim@example.com    | Kim Shawn   | null  | kim123    |
|       6 | 2022-05-23 17:13:49.391000 |       0 | duke@example.com   | Duke Dan    | null  | duke123   |
|       7 | 2022-05-23 17:18:17.579000 |       0 | jake@example.com   | Jake Joe    | null  | jake123   |
|       8 | 2022-05-23 17:18:22.372000 |       0 | kim@example.com    | Kim Shawn   | null  | kim123    |
|       9 | 2022-05-23 17:18:26.878000 |       0 | duke@example.com   | Duke Dan    | null  | duke123   |
|      10 | 2022-05-23 17:25:14.215000 |       0 | jake@example.com   | Jake Joe    | null  | jake123   |
|      11 | 2022-05-23 17:25:19.609000 |       0 | kim@example.com    | Kim Shawn   | null  | kim123    |
|      12 | 2022-05-23 17:25:24.689000 |       0 | duke@example.com   | Duke Dan    | null  | duke123   |
|      13 | 2022-06-13 02:47:14.980000 |       0 | shawn@example.com  | Shawn       | null  | shawn123  |
```

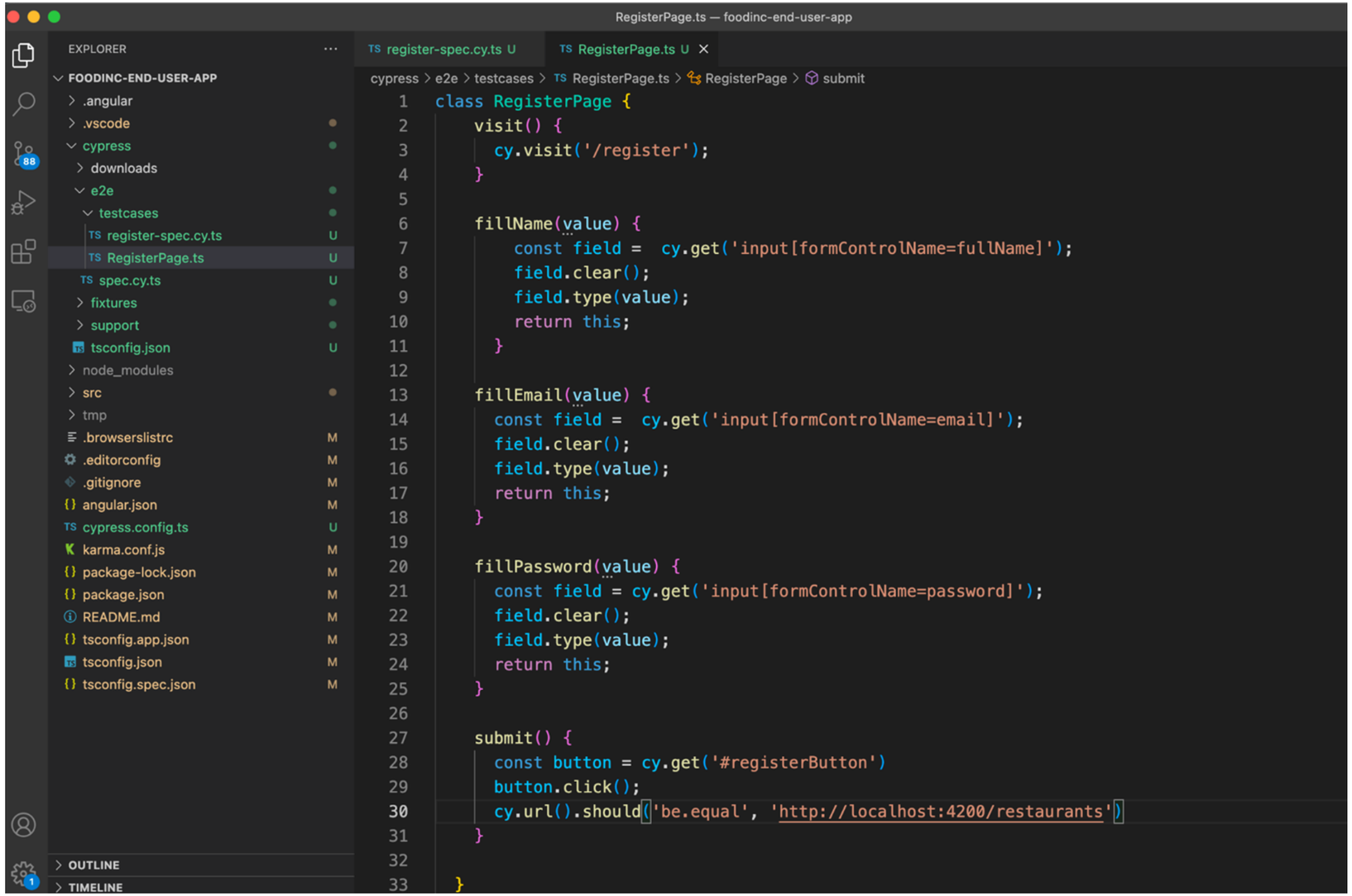
# Page Object Model for Register Page

Now, we will see Page Object Model in order to write test cases to test the register with name, email and password scenario. For this, we will follow below steps:

1. Create the RegisterPage.ts file to define the Page Object Model
2. Create the register-pageobject-spec.cy.ts to define the test case using Page Object Model
3. Execute the Test Case

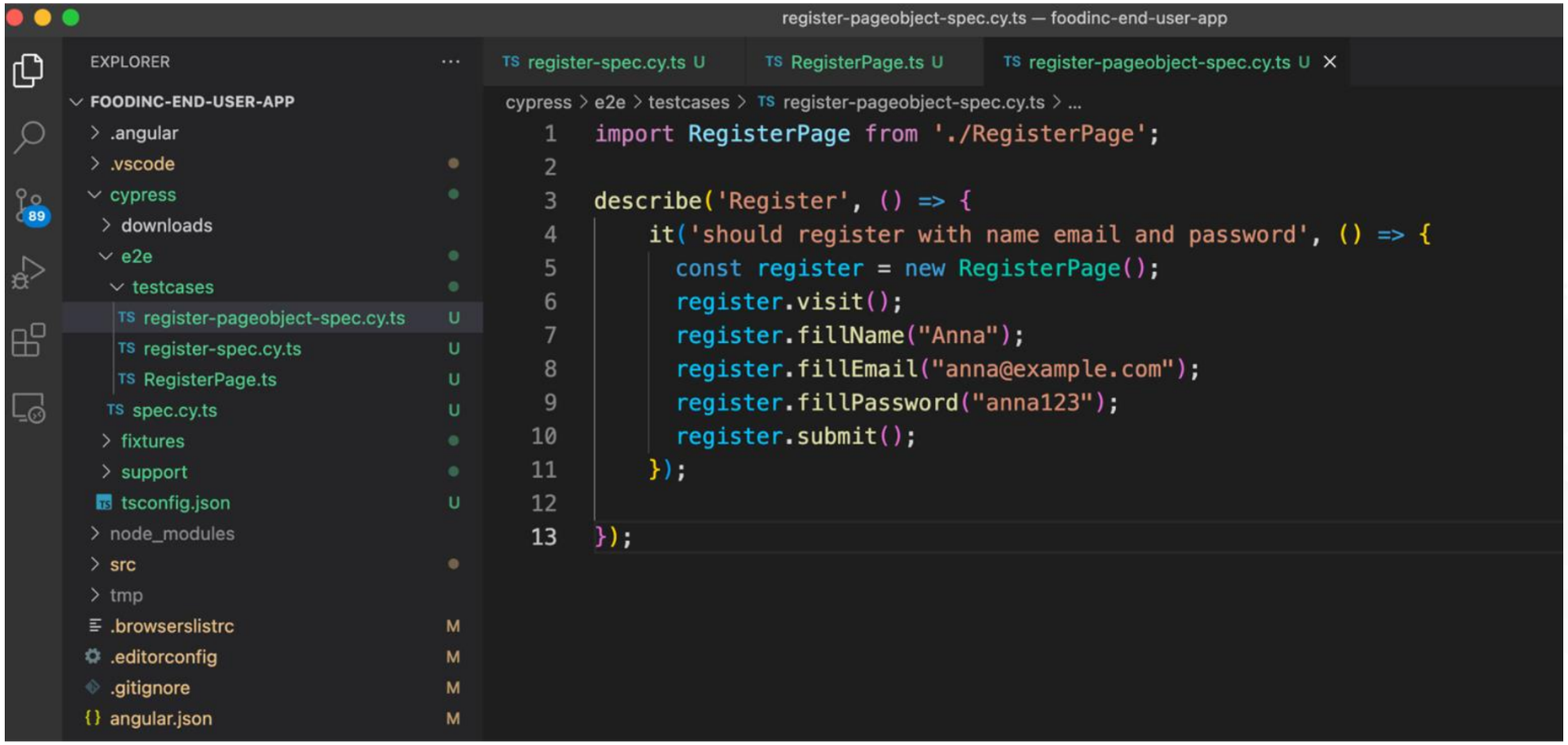
# Page Object Model for Register Page

1. Create the RegisterPage.ts file to define the Page Object Model in testcases directory



# Page Object Model for Register Page

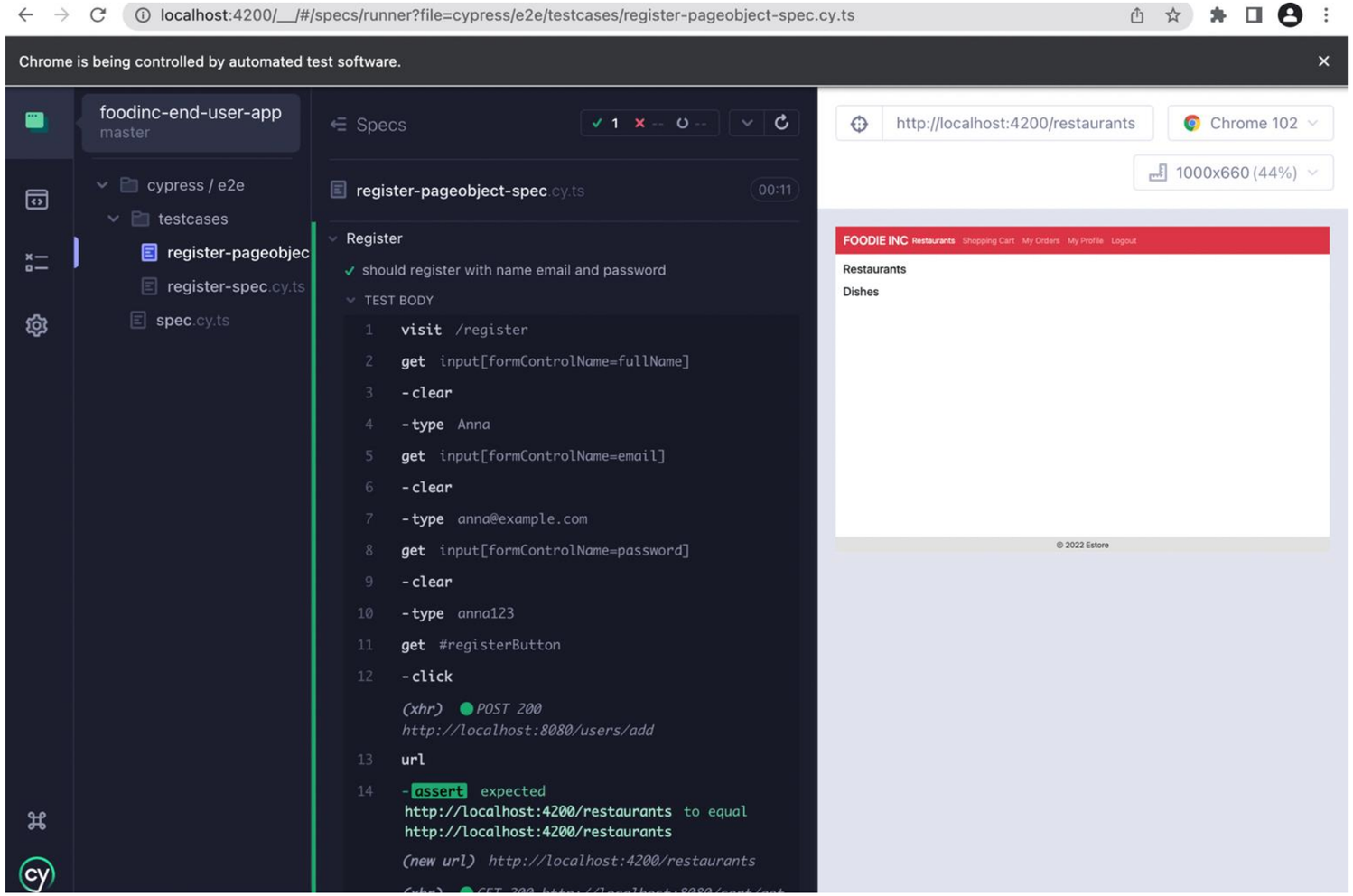
2. Create theregister-pageobject-spec.cy.ts to define the test case using Page Object Model



# Page Object Model for Register Page

3. Execute the Test Case

Now, finally let us run the file **register-pageobject-spec.cy.ts** on the specs section to validate if the testing works fine.



# Cypress Mochawesome Reporter

You can work with several reporting tools available in the market for cypress. We will configure cypress-mochawesome-reporter

You need to initially install the library.

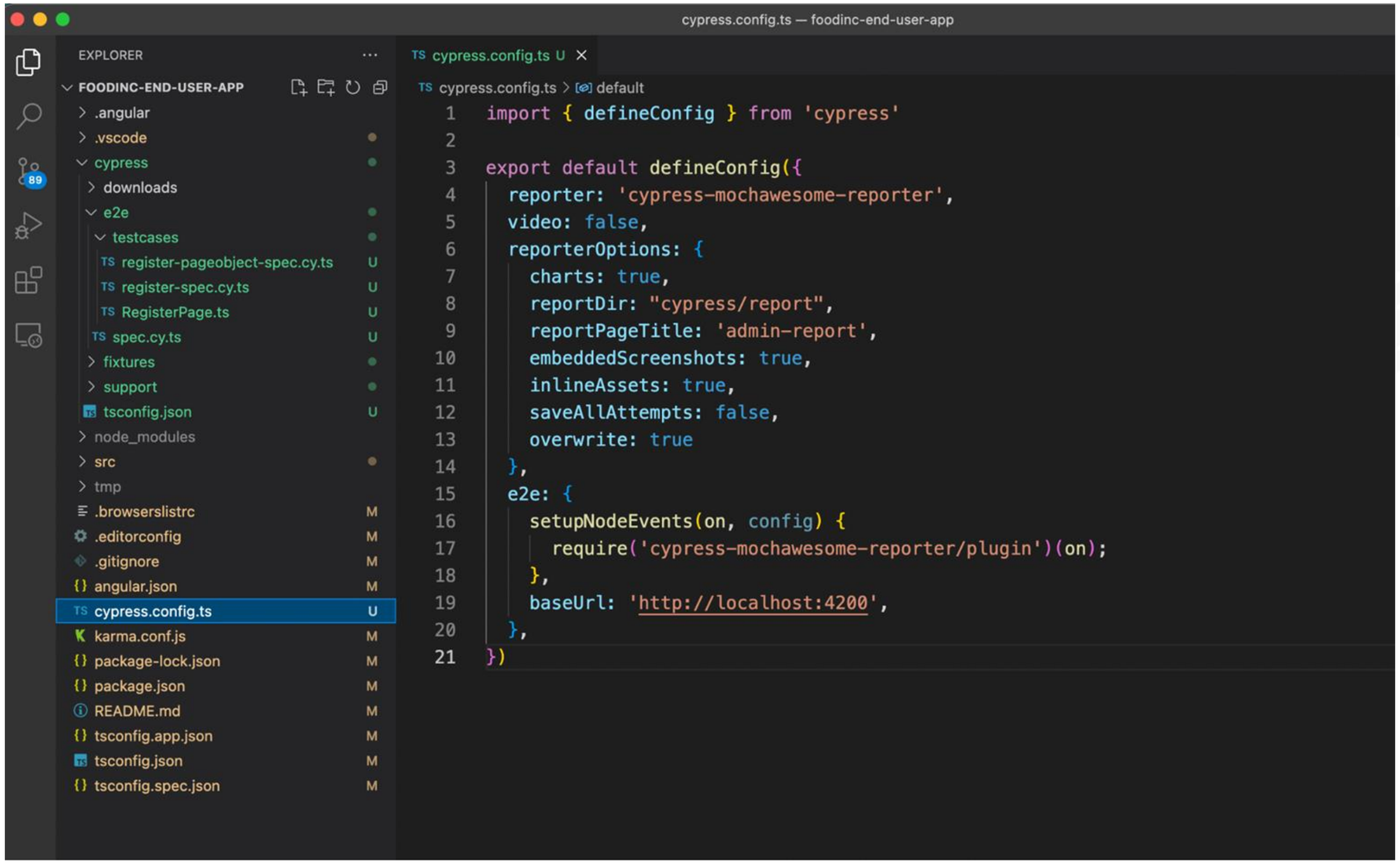
Use the below command:

**npm i --save-dev cypress-mochawesome-reporter**

You can also add the flag --legacy-peer-deps in case you find some error

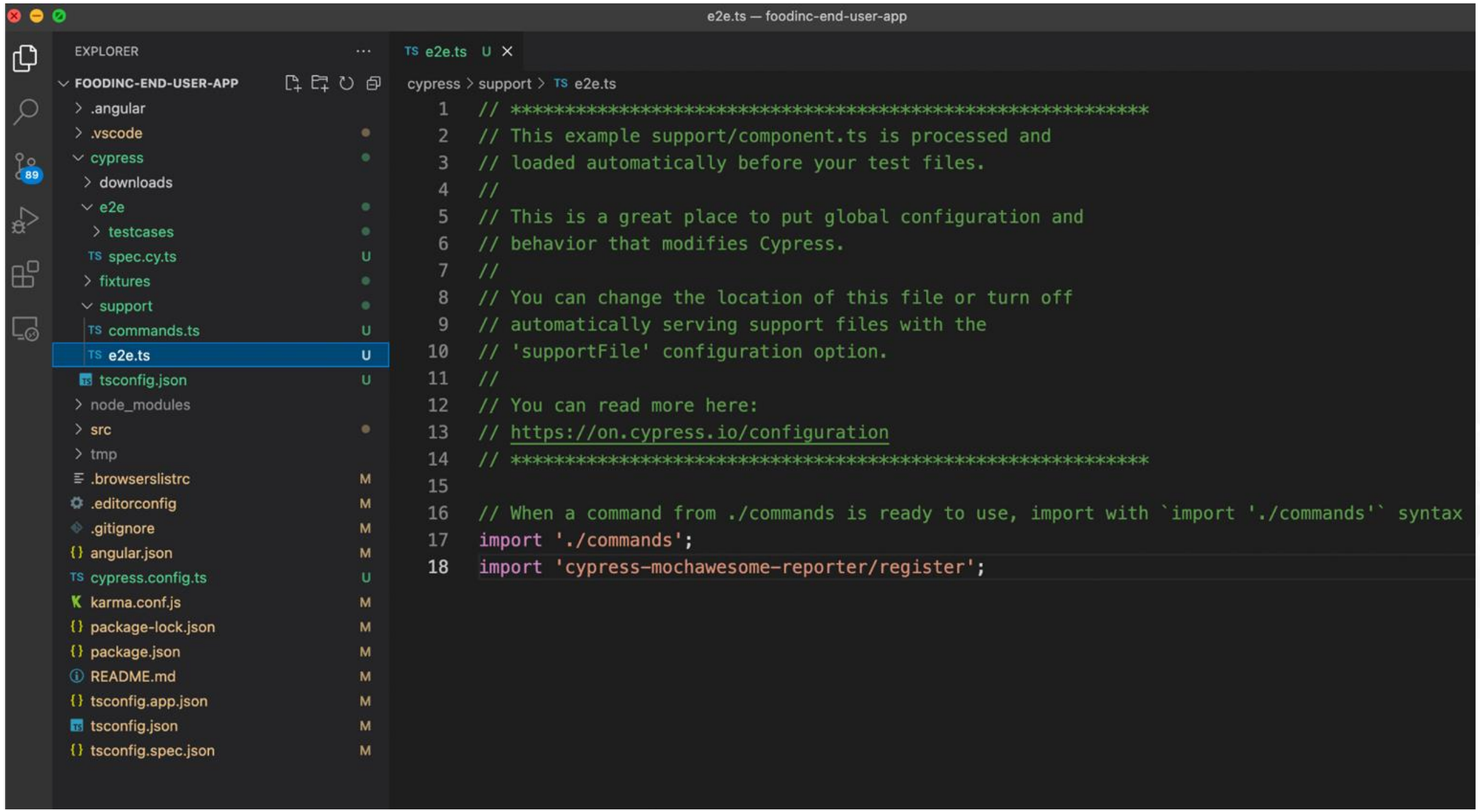
# Cypress Mochawesome Reporter

Now, configure reporter tool in your cypress.config.ts file. Add reporter options in the configuration as per your need.



# Cypress Mochawesome Reporter

Navigate to the cypress/support directory and import **cypress-mochawesome-reporter/register**



# Key Takeaways

- Create Test Cases in Cypress
- Work with Cypress Launchpad
- Hands On Page Object Model
- Configure mocha awesome reporter

# Before the Next Class

- Review Java
- Hands on with Eclipse Maven Project
- Review shared Java Backend Code
- Go through Selenium Framework APIs
- Go through the Selenium Configuration

# What's Next?

- Work with  Java to perform Automation Testing
- Work with Selenium to perform testing in real browser
- We will write simple test scripts for Admin Angular Project
- We will write simple test scripts for End User Angular Project