

# TECHNOLOGY



## 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



## 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



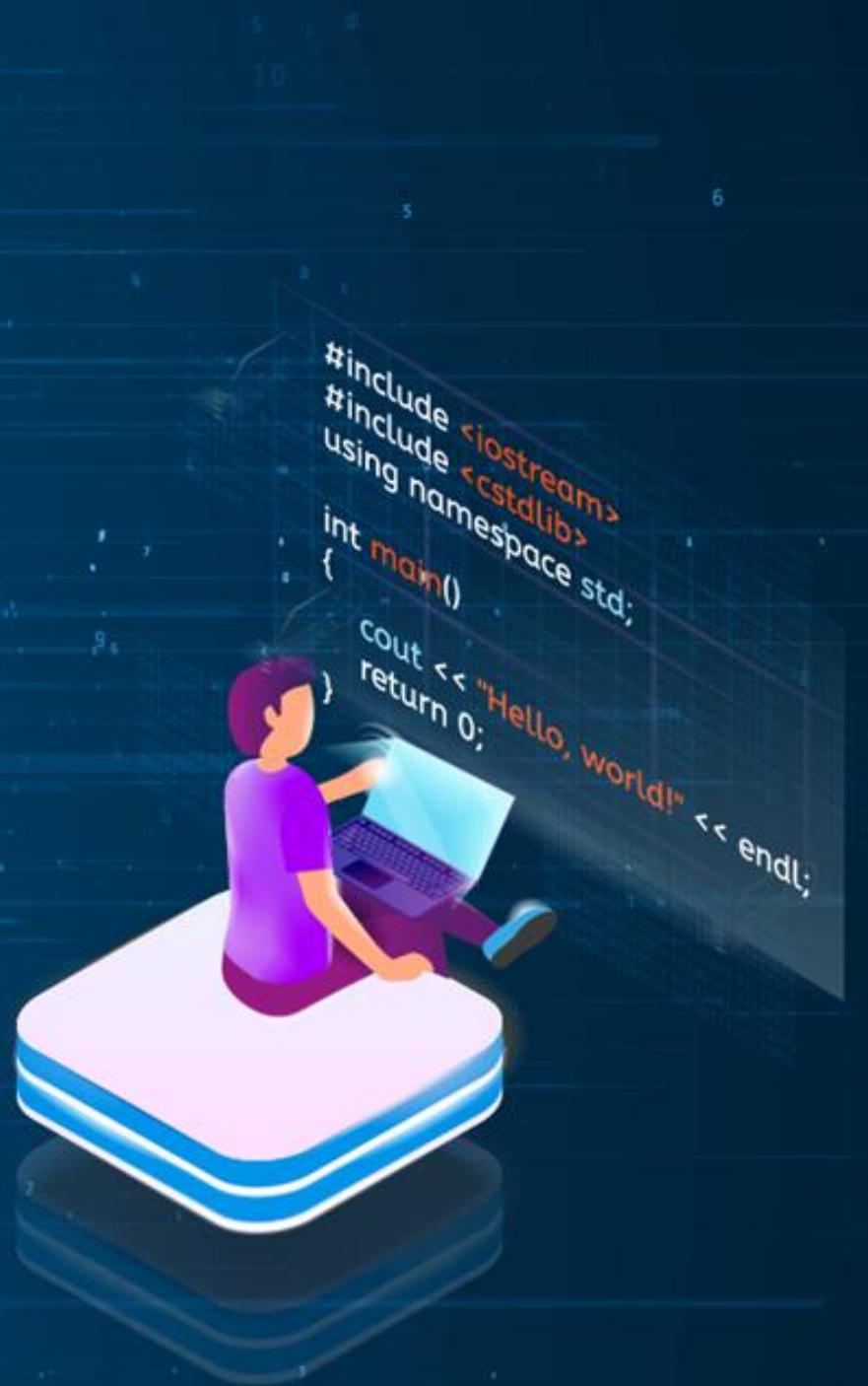
# Learning Objectives

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

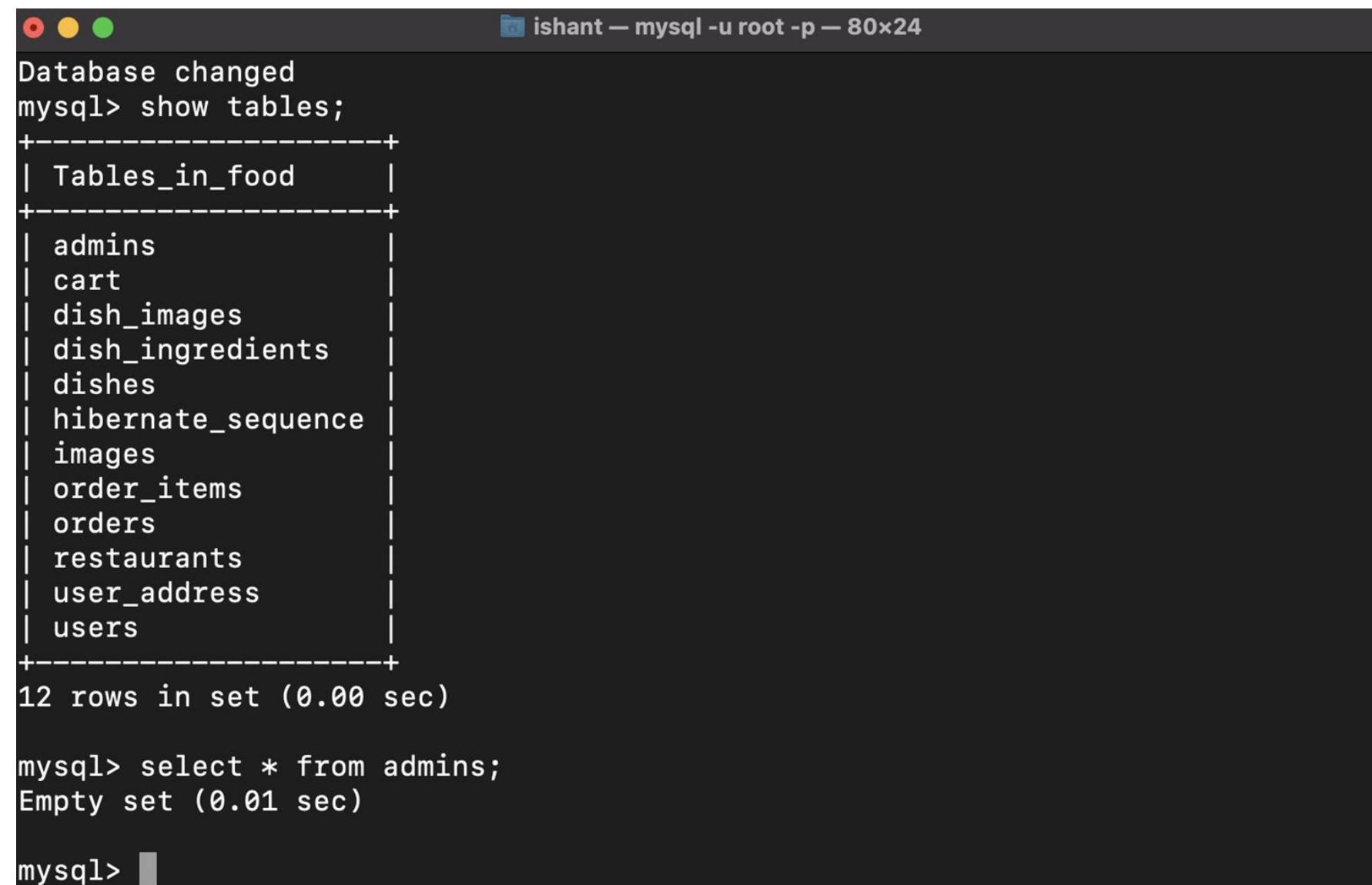


## Task 1: Test the Angular Admin Web Pages with Cypress



# 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.



```
Database changed
mysql> show tables;
+-----+
| Tables_in_food |
+-----+
| admins
| cart
| dish_images
| dish_ingredients
| dishes
| hibernate_sequence
| images
| order_items
| orders
| restaurants
| user_address
| users
+-----+
12 rows in set (0.00 sec)

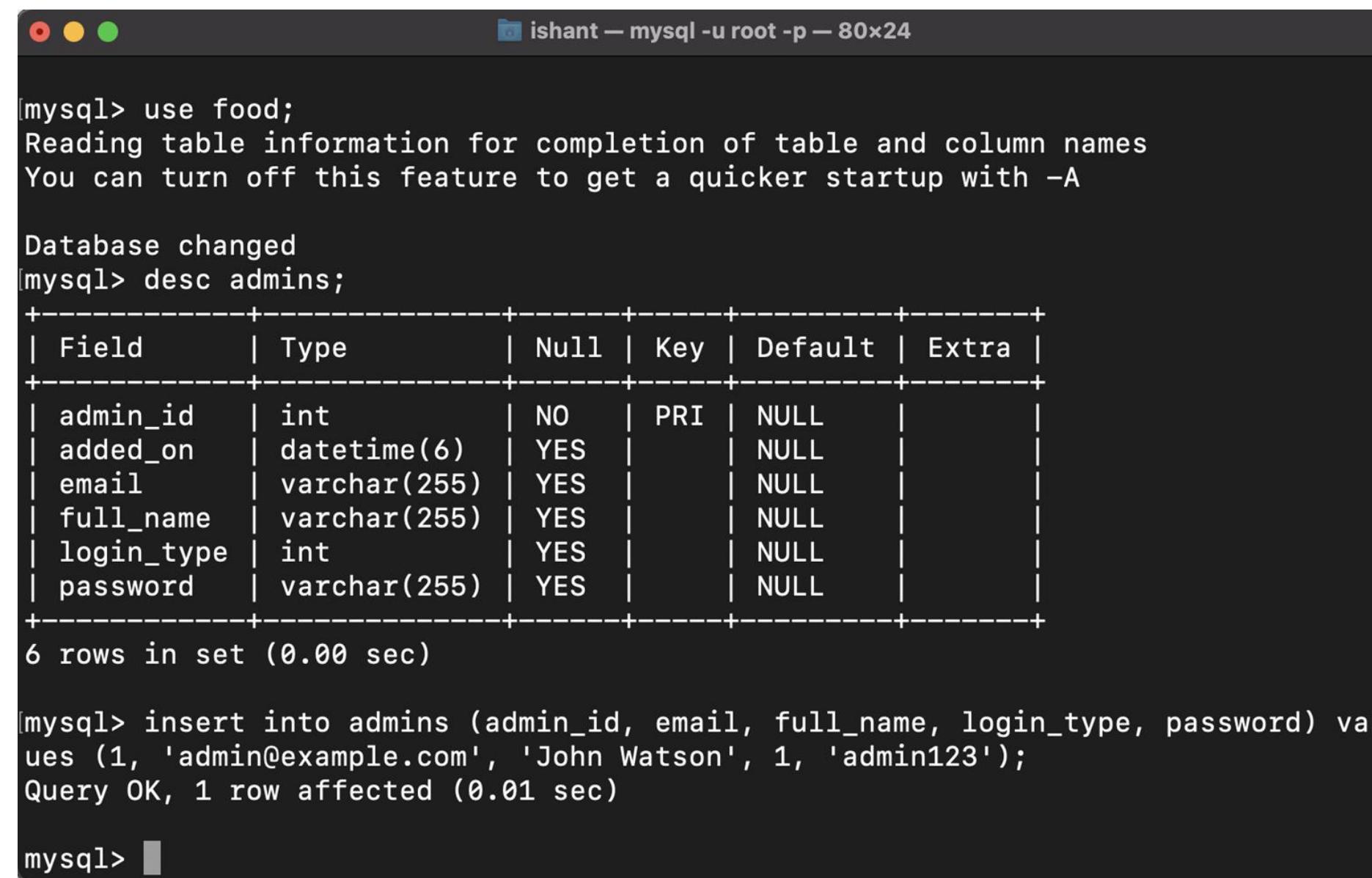
mysql> select * from admins;
Empty set (0.01 sec)

mysql>
```

# 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');
```



The screenshot shows a terminal window titled "ishant — mysql -u root -p — 80x24". The session starts with the command "use food;". It then displays table completion information and a note about turning off the feature for faster startup. The database is changed to "food". The "admins" table is described, showing six columns: admin\_id (int), added\_on (datetime(6)), email (varchar(255)), full\_name (varchar(255)), login\_type (int), and password (varchar(255)). All columns have a default value of NULL. The "desc admins;" command shows 6 rows in set (0.00 sec). Finally, an "insert" command is run to add a new row with values (1, 'admin@example.com', 'John Watson', 1, 'admin123'). The response indicates 1 row affected (0.01 sec).

```
[mysql]> use food;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
[mysql]> desc admins;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| admin_id    | int        | NO   | PRI | NULL    |          |
| added_on    | datetime(6) | YES  |     | NULL    |          |
| email       | varchar(255)| YES  |     | NULL    |          |
| full_name   | varchar(255)| YES  |     | NULL    |          |
| login_type  | int        | YES  |     | NULL    |          |
| password    | varchar(255)| YES  |     | NULL    |          |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

[mysql]> insert into admins (admin_id, email, full_name, login_type, password) values (1, 'admin@example.com', 'John Watson', 1, 'admin123');
Query OK, 1 row affected (0.01 sec)

mysql>
```

# Build and execute Angular Admin Project

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

```
foodinc-admin-dashboard — ng serve -o TMPDIR=/var/folders/5g/p0vt4z4n6rlgcph3nkxkljtr0000gn/T/ __CFBu...
ishant@Ishants-MacBook-Pro foodinc-admin-dashboard % ng serve -o
Your global Angular CLI version (13.2.5) is greater than your local version (13.0.1). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
✓ Browser application bundle generation complete.

Initial Chunk Files
vendor.js
styles.css, styles.js
polyfills.js
scripts.js
main.js
runtime.js

Names          Size
vendor         3.44 MB
styles          382.27 kB
polyfills       339.48 kB
scripts         145.27 kB
main            26.38 kB
runtime         12.82 kB

Initial Total  4.33 MB

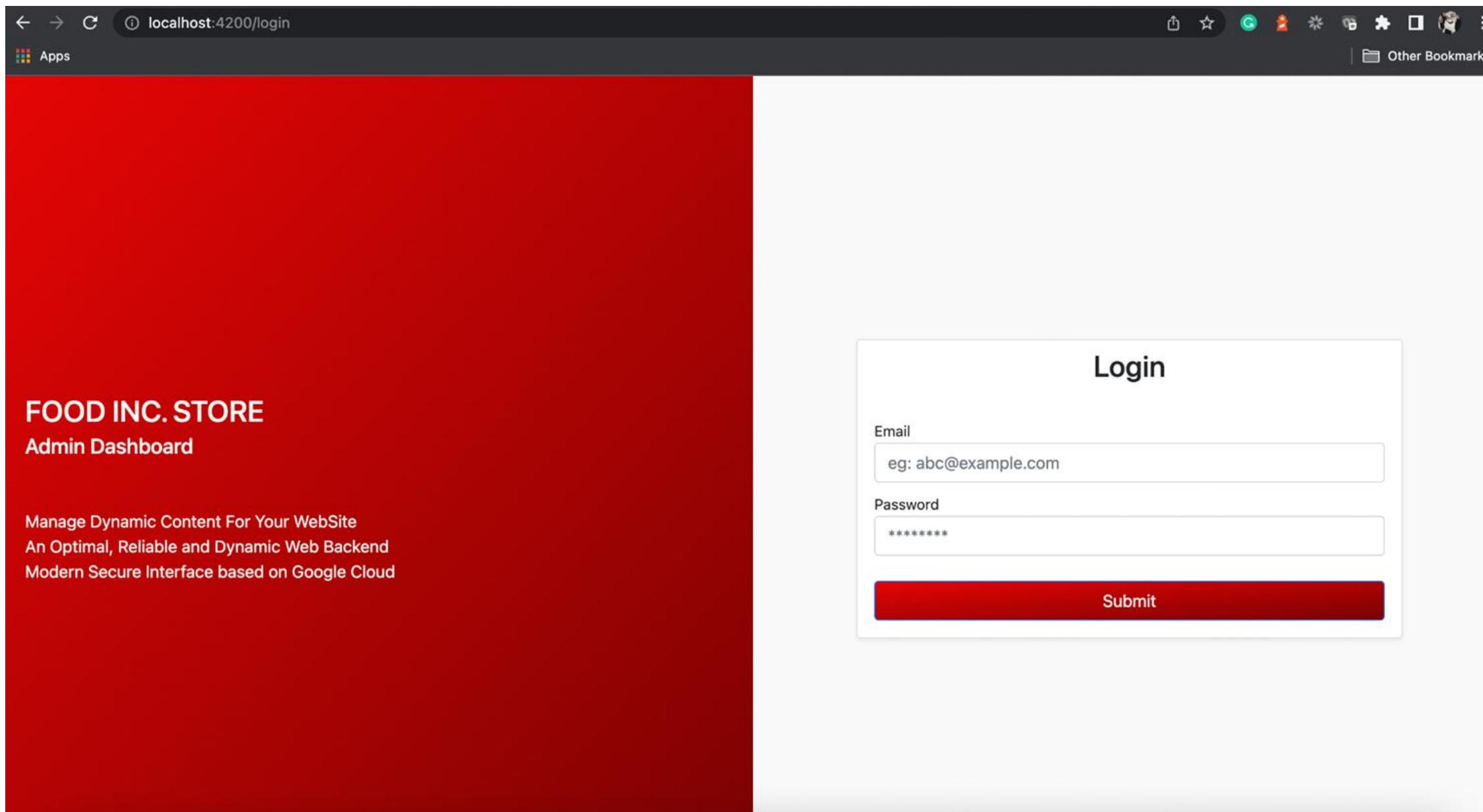
Lazy Chunk Files
src_app_layouts_admins_admins_module_ts.js
src_app_layouts_auths_auths_module_ts.js

Names          Size
-              3.82 MB
-              18.41 kB

Build at: 2022-05-20T07:26:35.064Z - Hash: 7b3007d12e34dfc0 - Time: 18990ms
```

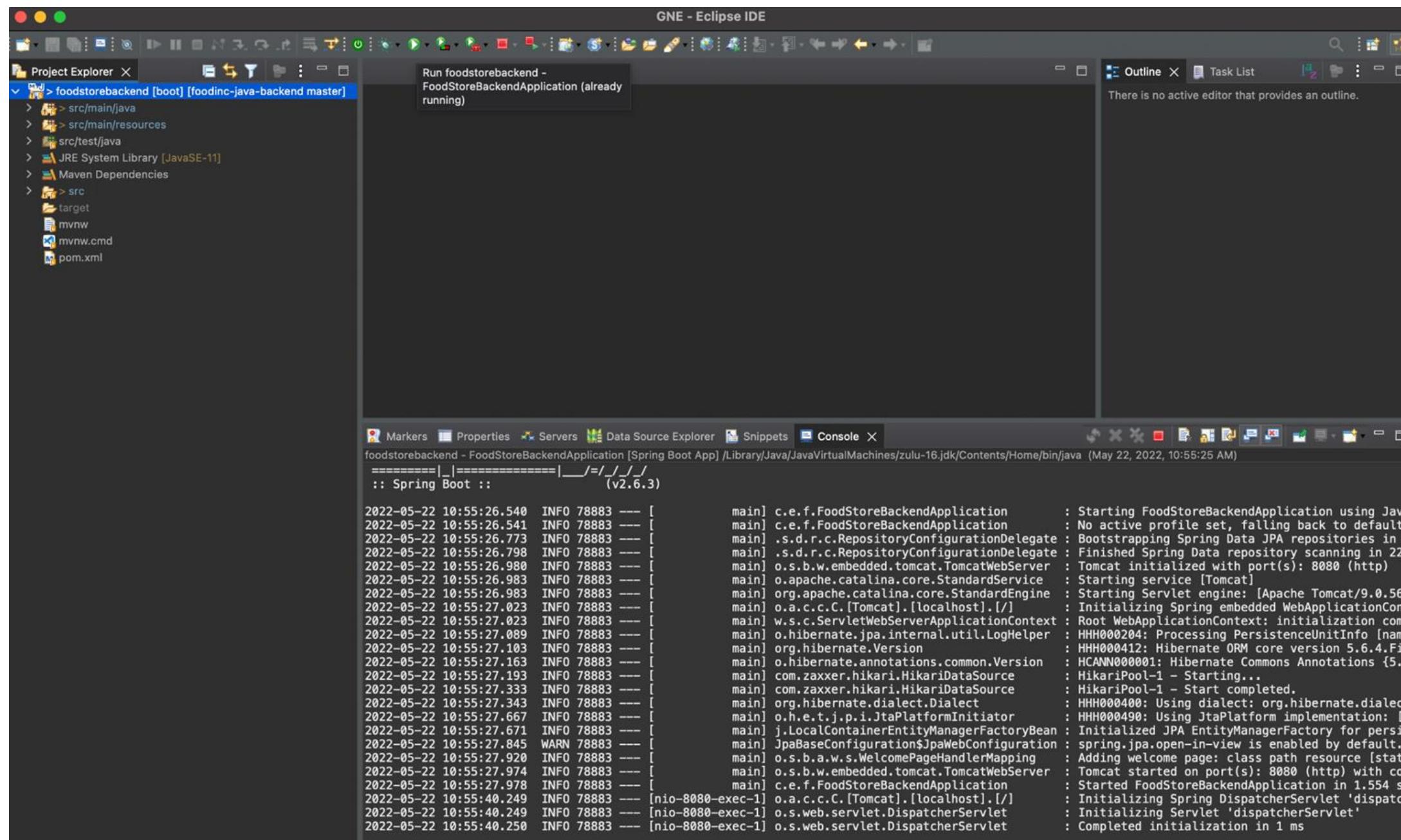
# Build and execute Angular Admin Project

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**

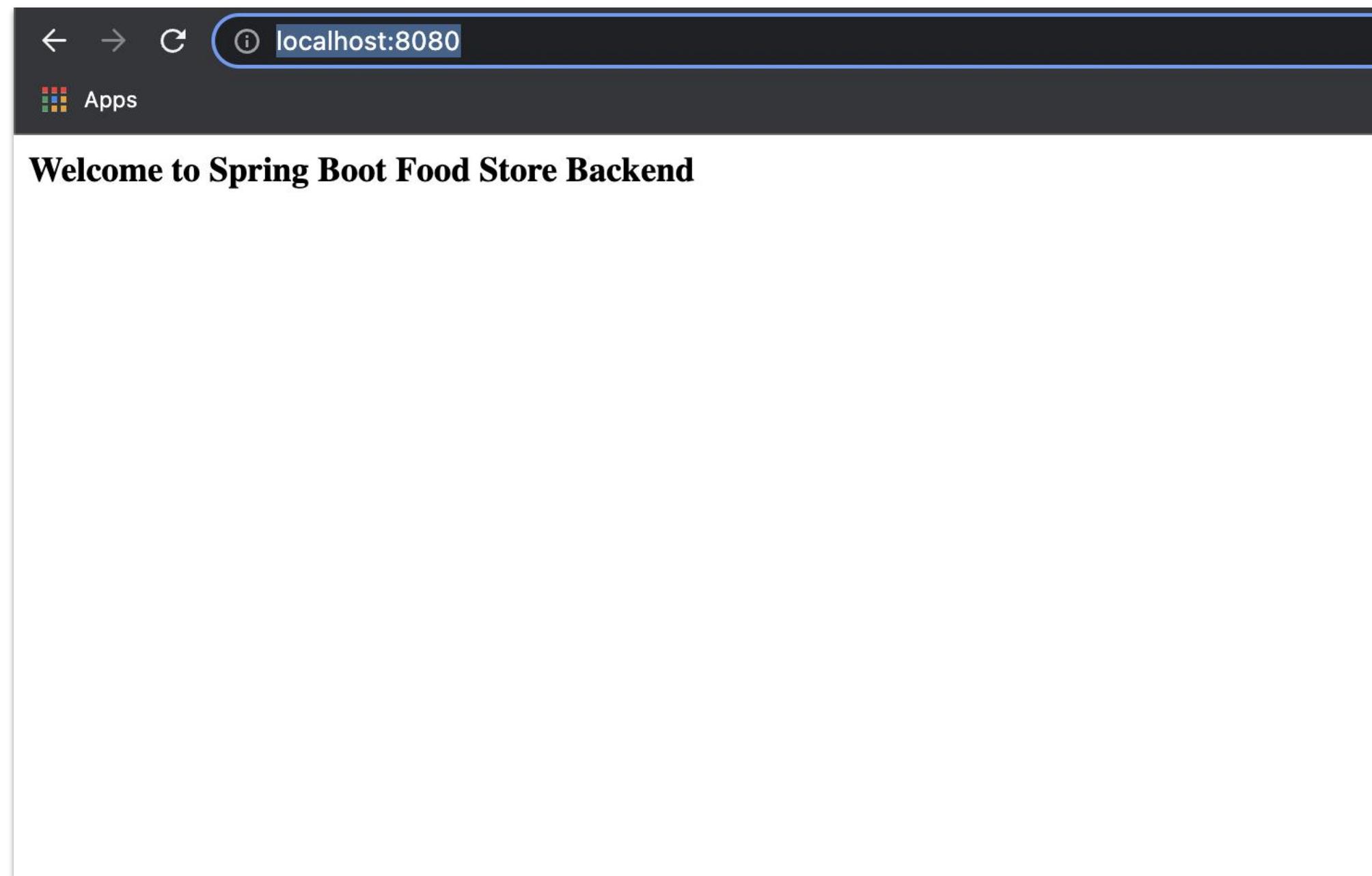


The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "foodstorebackend [boot] [foodinc-java-backend master]" with folders like src/main/java, src/main/resources, src/test/java, JRE System Library [JavaSE-11], Maven Dependencies, src, target, mvnw, mvnw.cmd, and pom.xml.
- Run Configuration:** A tooltip indicates "Run foodstorebackend - FoodStoreBackendApplication (already running)".
- Outline and Task List:** Both are empty.
- Console Tab:** Displays the Spring Boot application startup logs. Key logs include:
  - Starting FoodStoreBackendApplication using Java
  - No active profile set, falling back to default
  - Bootstrapping Spring Data JPA repositories in DB
  - Finished Spring Data repository scanning in 22 ms
  - Tomcat initialized with port(s): 8080 (http)
  - Starting service [Tomcat]
  - Starting Servlet engine: [Apache Tomcat/9.0.56]
  - Initializing Spring embedded WebApplicationContext
  - Root WebApplicationContext: initialization completed
  - HHH000204: Processing PersistenceUnitInfo [name: ...]
  - HHH000412: Hibernate ORM core version 5.6.4.Final
  - HCCANN00001: Hibernate Commons Annotations {5.1.0.Final}
  - HikariPool-1 - Starting...
  - HikariPool-1 - Start completed.
  - HHH000400: Using dialect: org.hibernate.dialect
  - HHH000490: Using JtaPlatform implementation: [org
  - Initialized JPA EntityManagerFactory for persist
  - spring.jpa.open-in-view is enabled by default. T
  - Adding welcome page: class path resource [static
  - Tomcat started on port(s): 8080 (http) with cont
  - Started FoodStoreBackendApplication in 1.554 sec
  - Initializing Spring DispatcherServlet 'dispatcherServlet'
  - Initializing Servlet 'dispatcherServlet'
  - Completed initialization in 1 ms

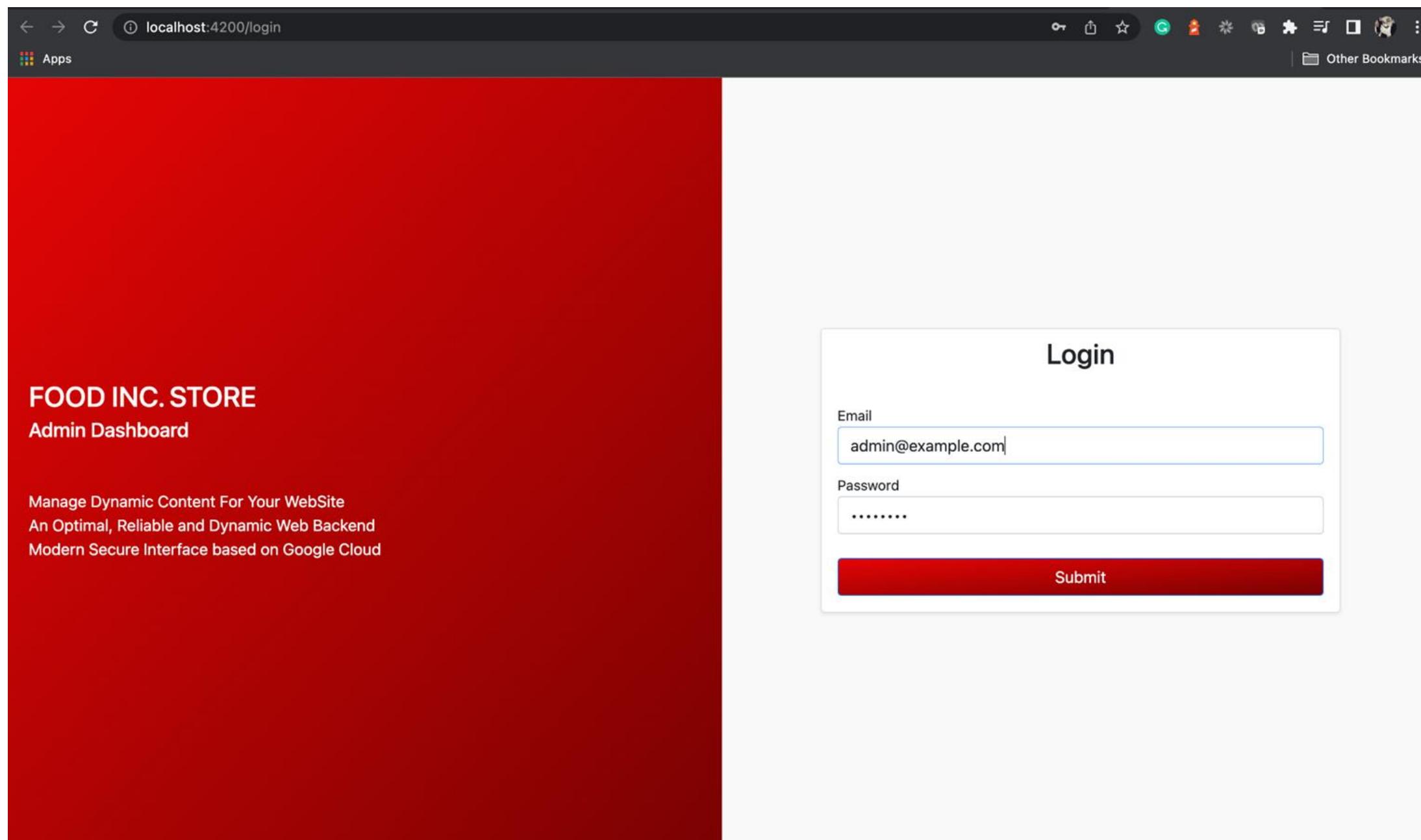
# Check Java Backend Project Status as Running

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



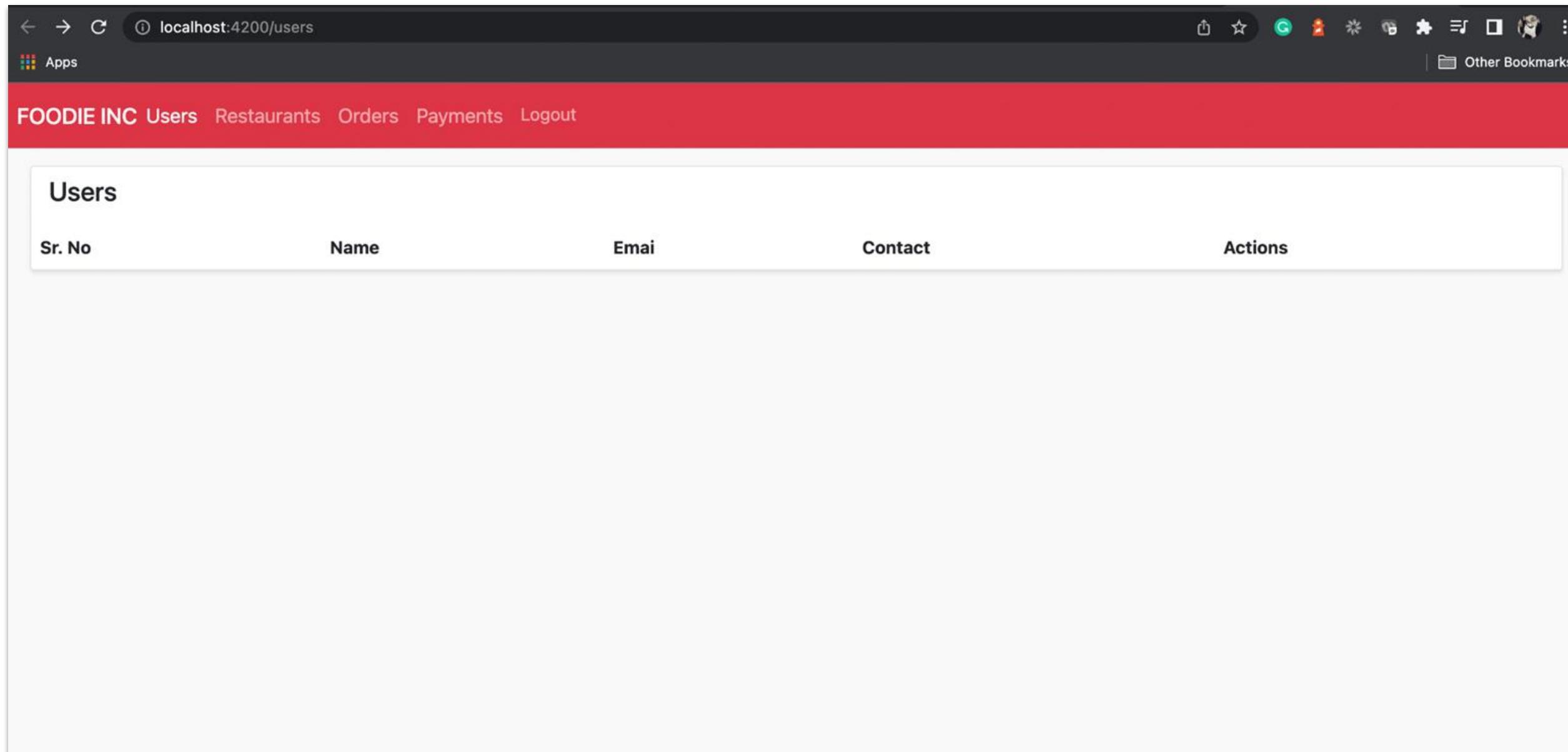
# Login with your Admin Credentials

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.



The screenshot shows a web browser window with the URL `localhost:4200/users` in the address bar. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons. The main content area has a red header bar with the text "FOODIE INC" and links for "Users", "Restaurants", "Orders", "Payments", and "Logout". Below this is a white table with a header row containing columns for "Sr. No", "Name", "Email", "Contact", and "Actions". The table body is currently empty, showing only the header row.

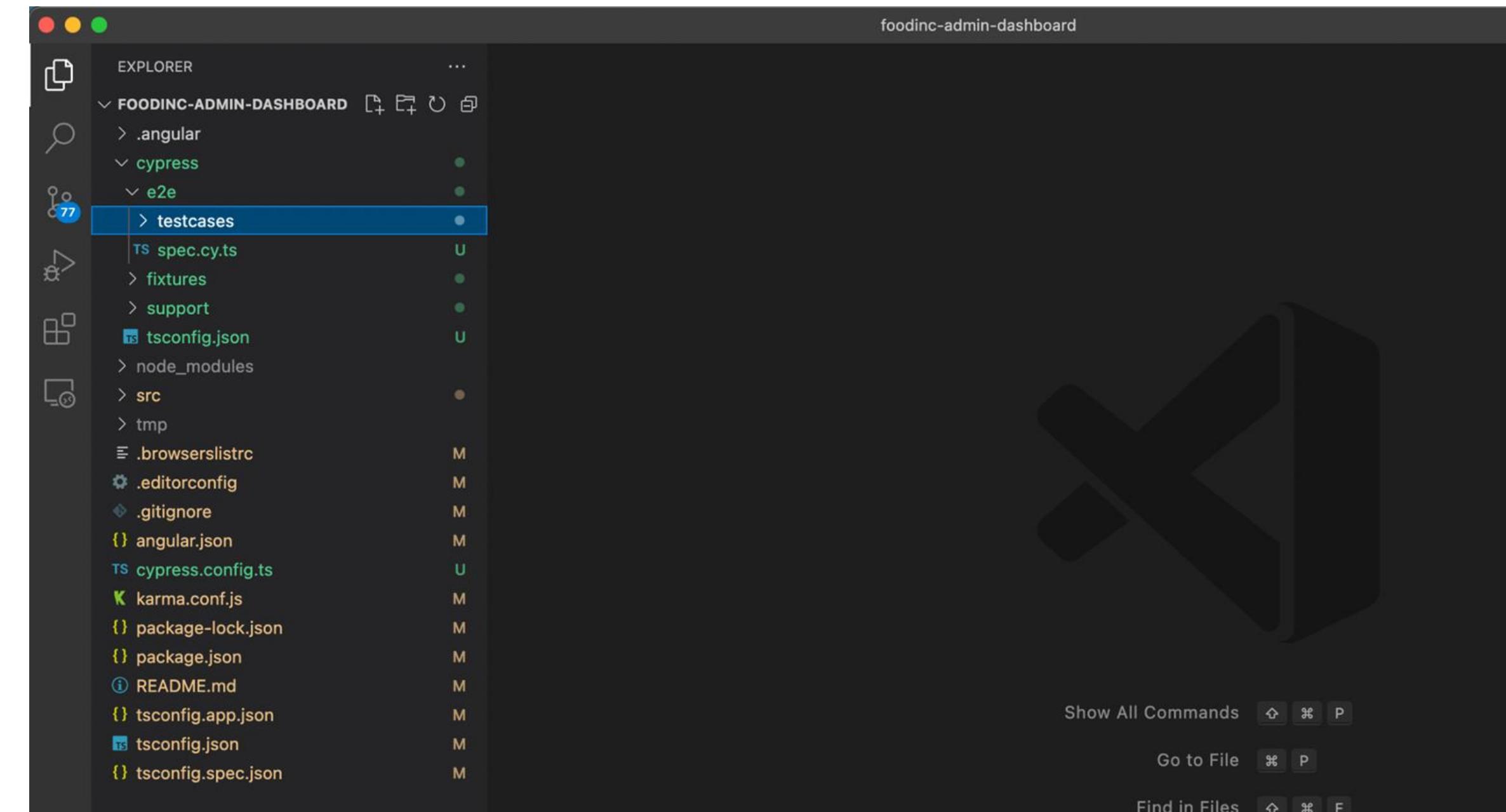
# Write Test Case for Login Page

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.

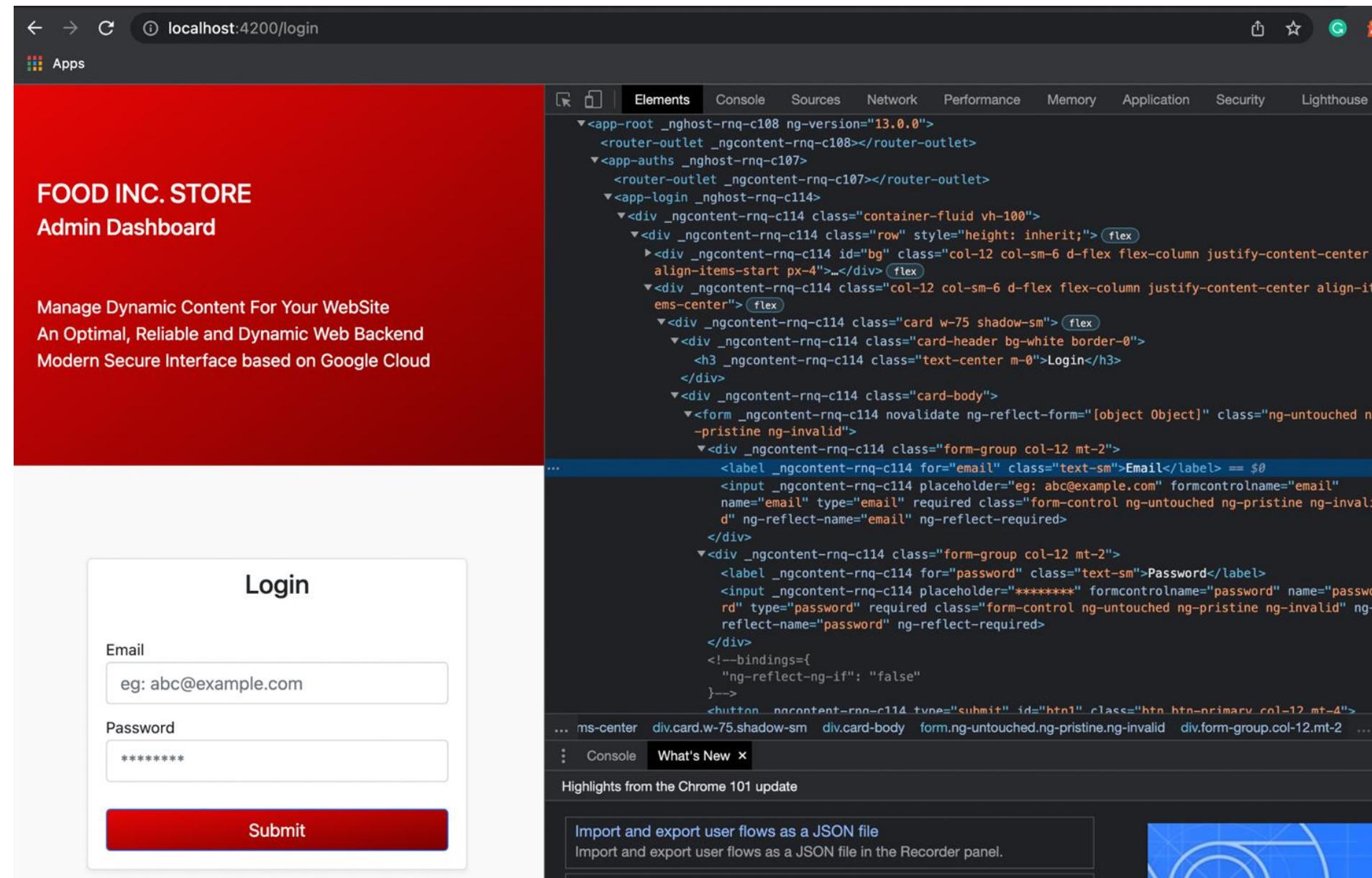


# Write Test Case for Login Page

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**

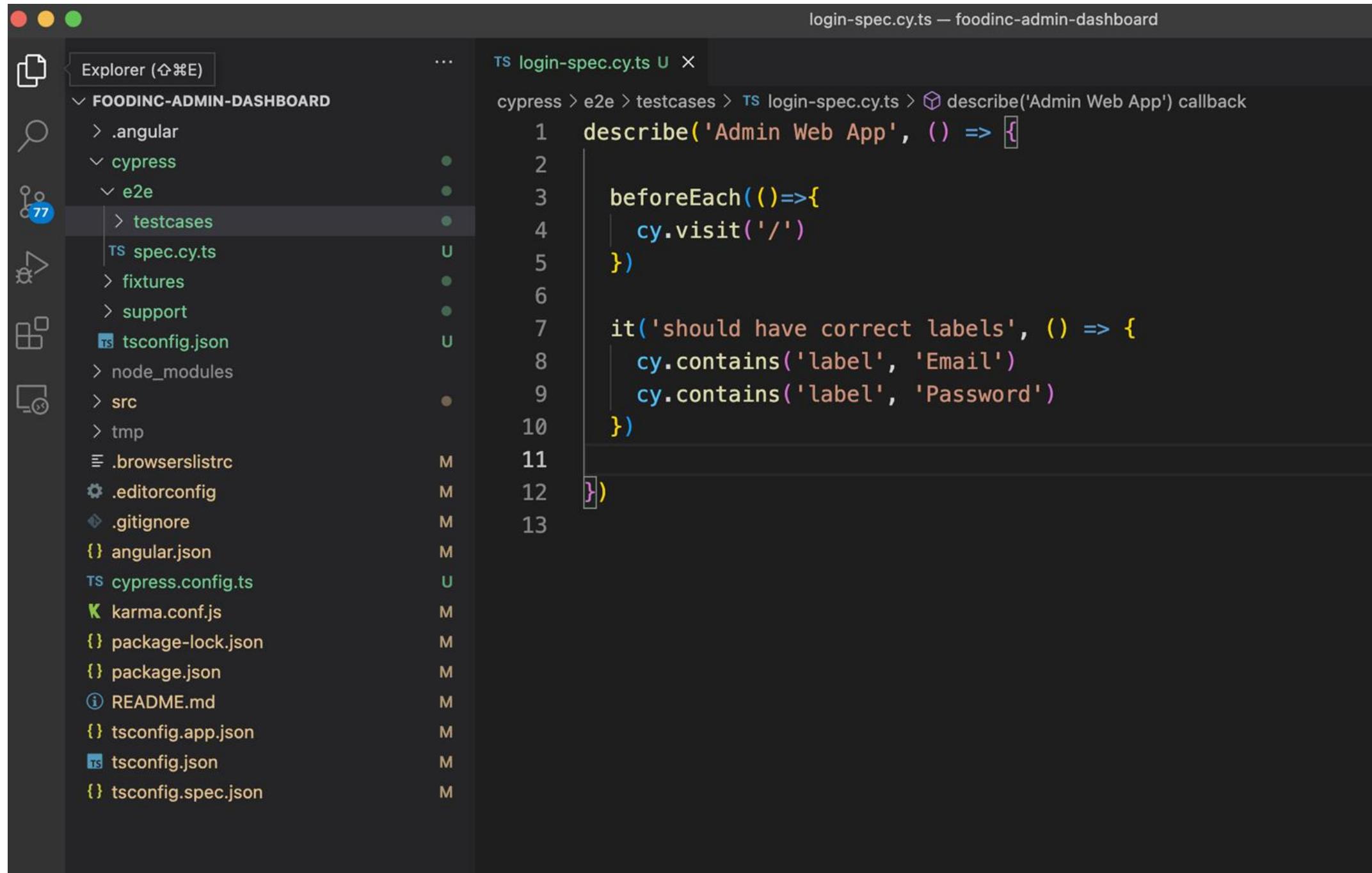
In order to check the labels, we will use CSS Locator. Right click on your web app and open elements to find the css class for the labels.



# 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**



The screenshot shows a dark-themed instance of Visual Studio Code. On the left is the Explorer sidebar, which lists the project structure under 'FOODINC-ADMIN-DASHBOARD'. The 'testcases' folder contains a file named 'spec.cy.ts'. The main editor area on the right displays the code for 'login-spec.cy.ts'. The code is written in Cypress.js, defining a test for the 'Admin Web App' login page. It includes a 'describe' block for the admin web app, a 'beforeEach' block to visit the root URL, and an 'it' block to check for correct labels ('Email' and 'Password').

```
describe('Admin Web App', () => {
  beforeEach(()=>{
    cy.visit('/')
  })
  it('should have correct labels', () => {
    cy.contains('label', 'Email')
    cy.contains('label', 'Password')
  })
})
```

# 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**

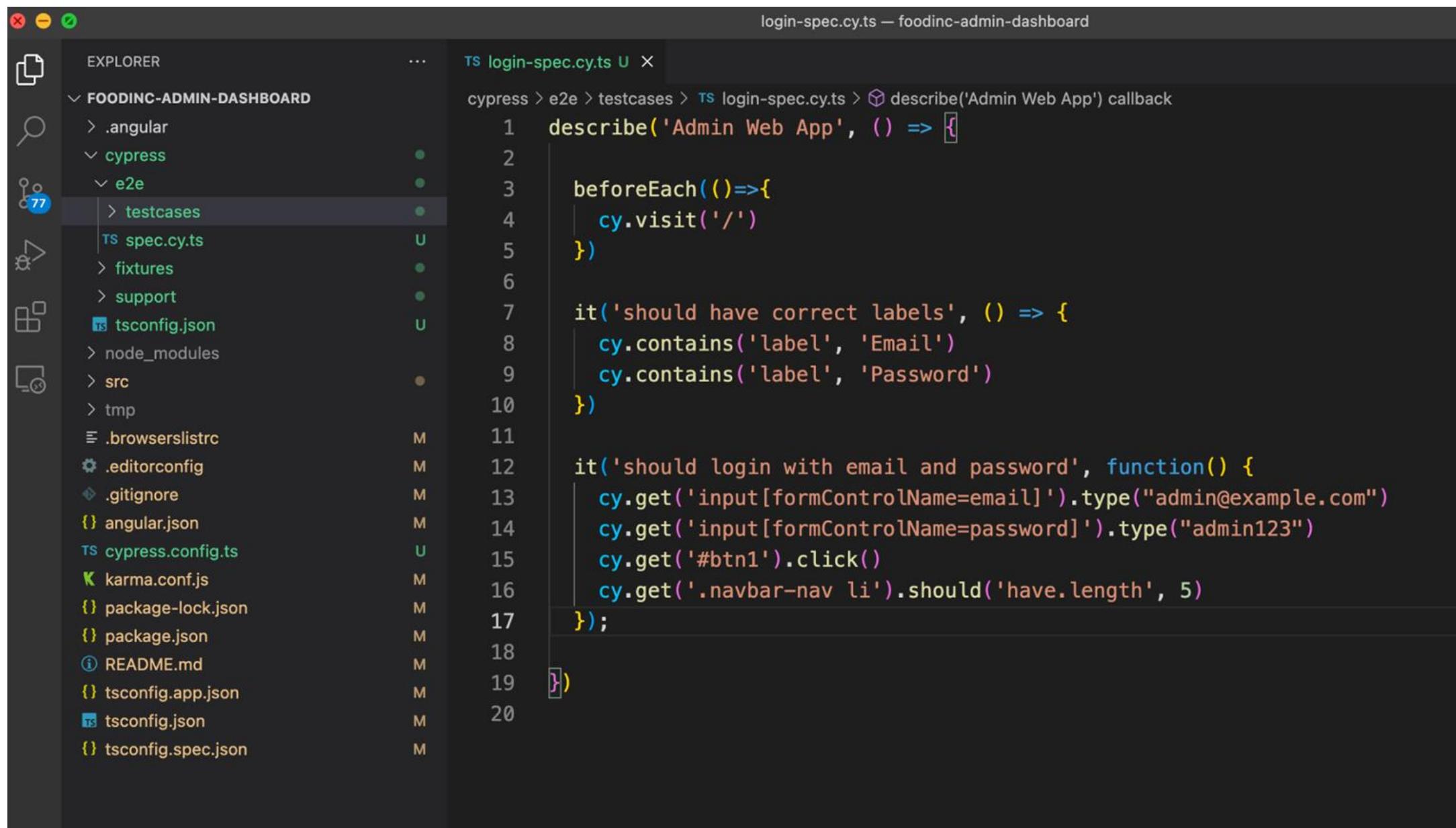
In order to send data to input fields, we will use formControlName Locator. Right click on your web app and open elements to find the formControlName and The Button Id.

The screenshot shows a browser window with the URL `localhost:4200/login`. On the left, the main content area displays the 'Admin Dashboard' of 'FOOD INC. STORE' with a red header and a white body containing a 'Login' form. The 'Login' form has two input fields: 'Email' and 'Password', and a 'Submit' button at the bottom. On the right, the 'Elements' tab of the Chrome DevTools is open, showing the HTML structure of the page. The 'Email' input field is selected, and its class is highlighted as `form-control ng-untouched ng-pristine ng-invalid`. The 'Submit' button is identified by the ID `btn1`.

# 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**

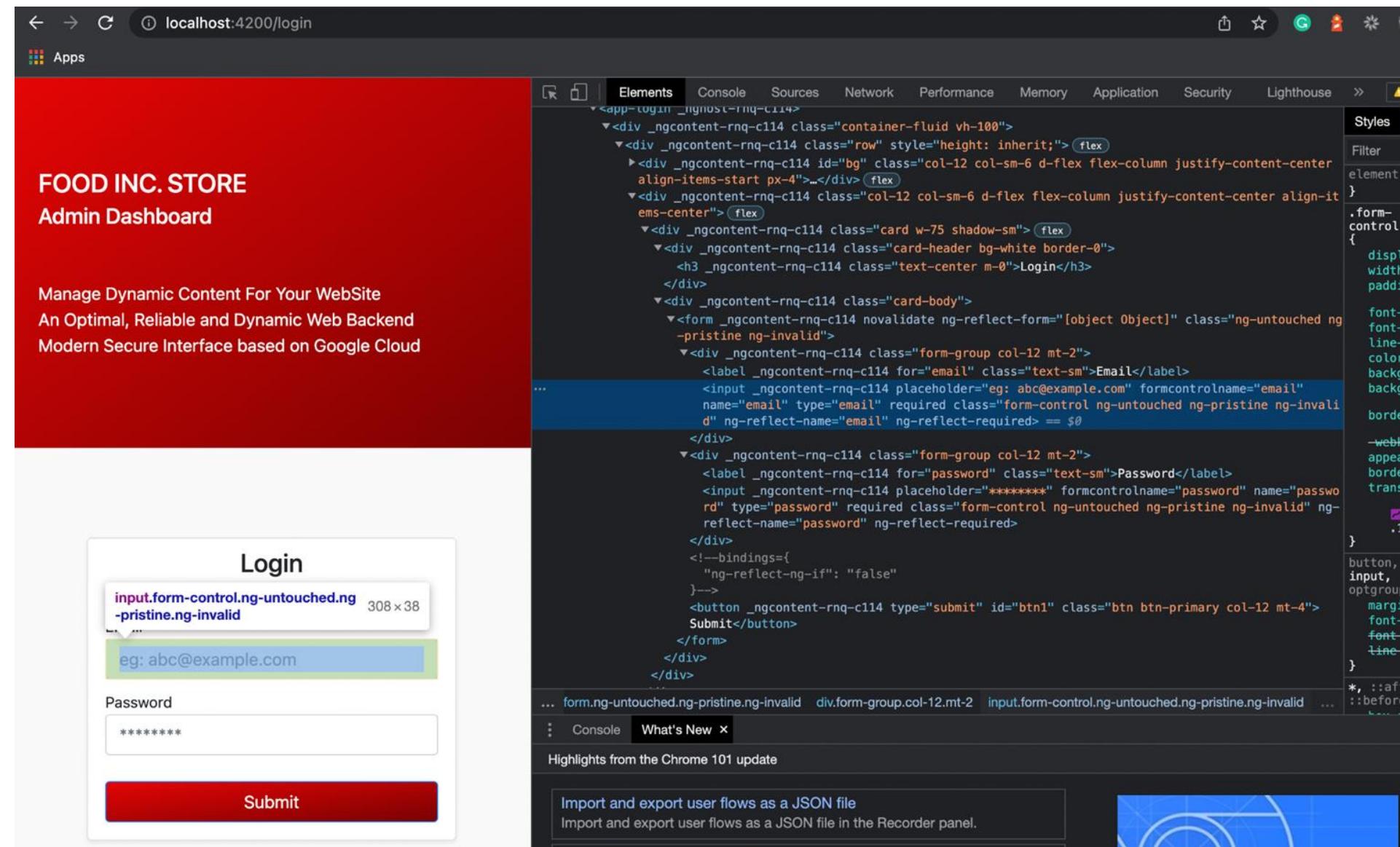


```
login-spec.cy.ts — foodinc-admin-dashboard
cypress > e2e > testcases > TS login-spec.cy.ts > describe('Admin Web App') callback
1  describe('Admin Web App', () => {
2
3    beforeEach(()=>{
4      cy.visit('/')
5    })
6
7    it('should have correct labels', () => {
8      cy.contains('label', 'Email')
9      cy.contains('label', 'Password')
10   })
11
12   it('should login with email and password', function() {
13     cy.get('input[formControlName=email]').type("admin@example.com")
14     cy.get('input[formControlName=password]').type("admin123")
15     cy.get('#btn1').click()
16     cy.get('.navbar-nav li').should('have.length', 5)
17   });
18
19 });
20
```

# 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.

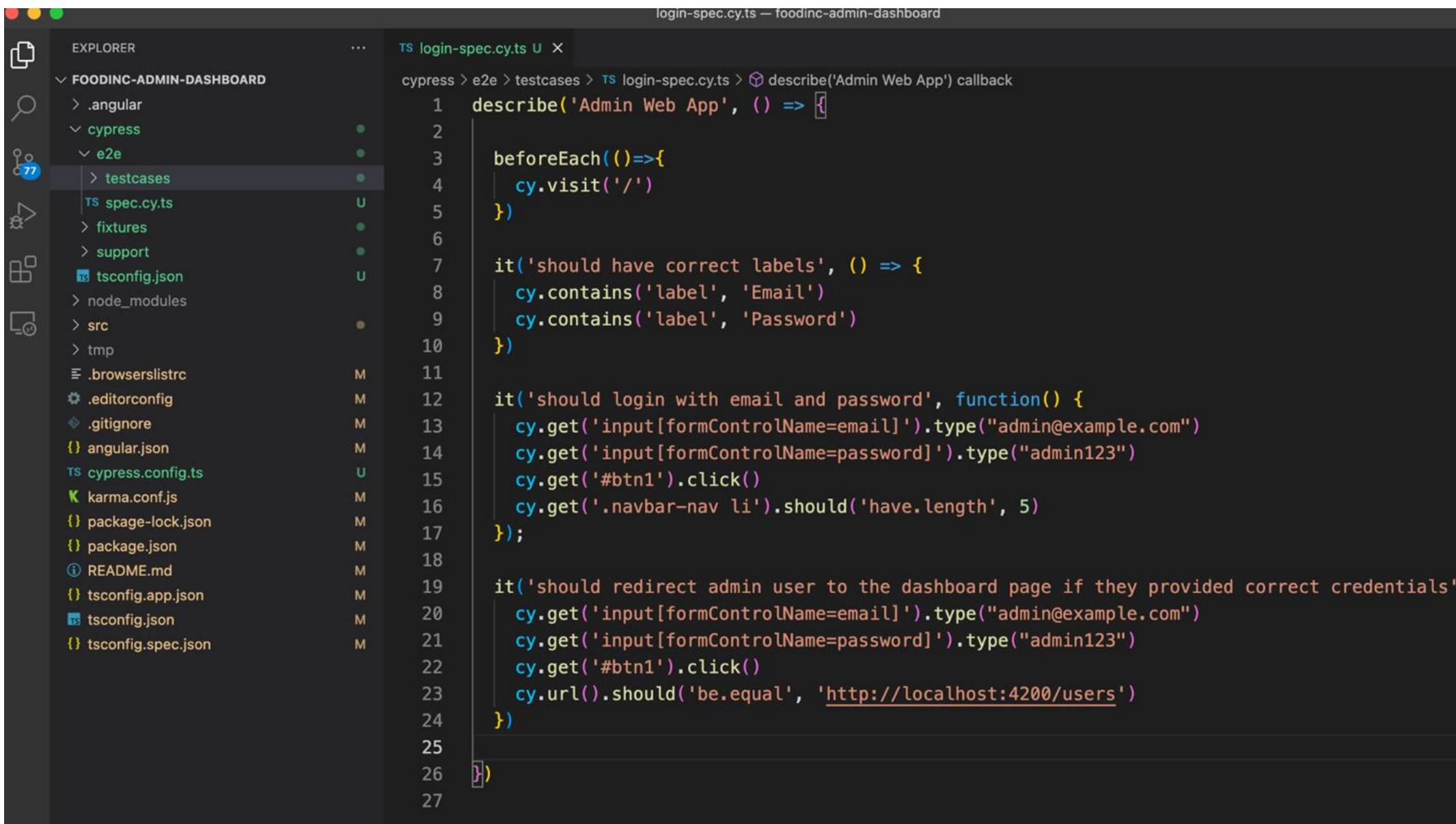


# Write Test Case for Login Page

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

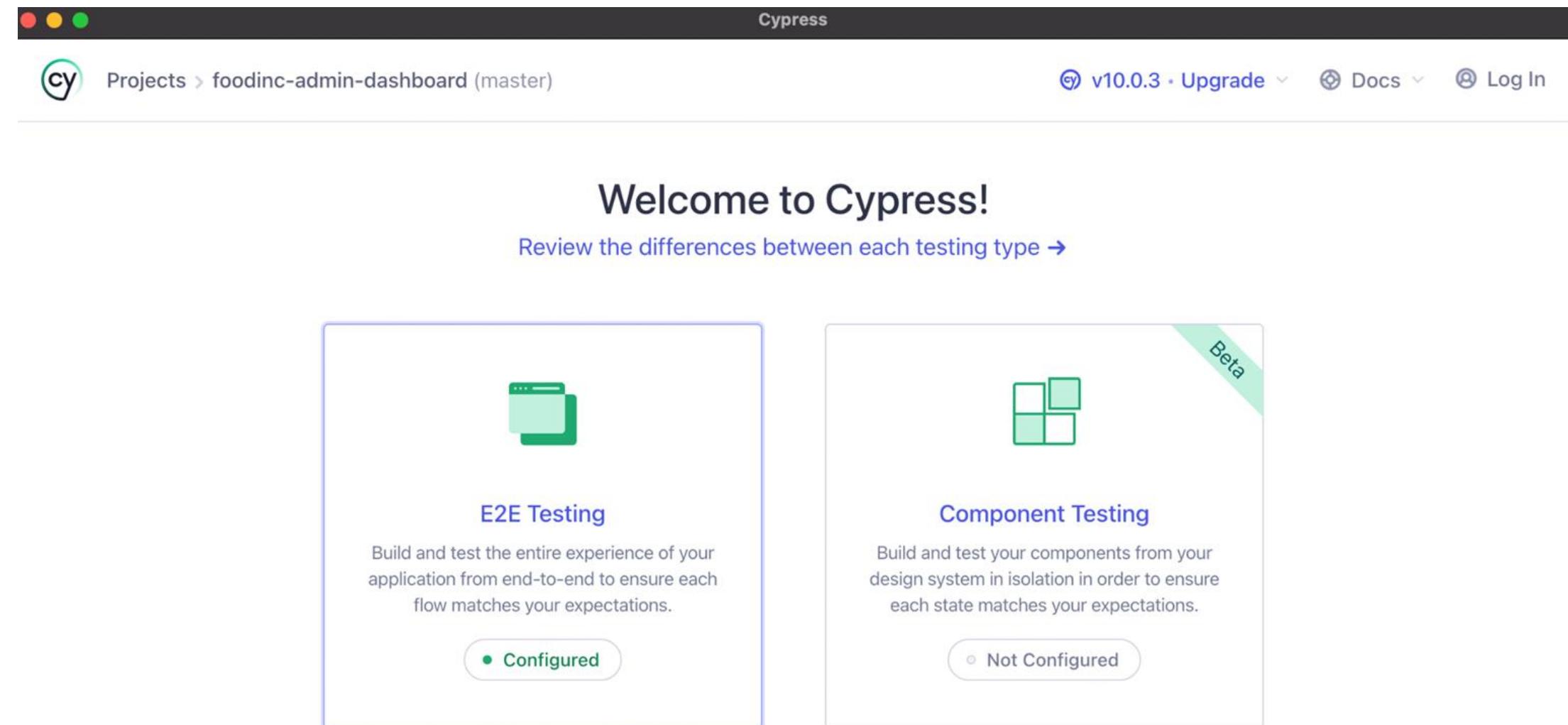


```
login-spec.cy.ts — foodinc-admin-dashboard
cypress > e2e > testcases > TS login-spec.cy.ts > describe('Admin Web App') callback
1  describe('Admin Web App', () => [
2
3    beforeEach(()=>{
4      cy.visit('/')
5    })
6
7    it('should have correct labels', () => {
8      cy.contains('label', 'Email')
9      cy.contains('label', 'Password')
10   })
11
12   it('should login with email and password', function() {
13     cy.get('input[formControlName=email]').type("admin@example.com")
14     cy.get('input[formControlName=password]').type("admin123")
15     cy.get('#btn1').click()
16     cy.get('.navbar-nav li').should('have.length', 5)
17   })
18
19   it('should redirect admin user to the dashboard page if they provided correct credentials',
20     cy.get('input[formControlName=email]').type("admin@example.com")
21     cy.get('input[formControlName=password]').type("admin123")
22     cy.get('#btn1').click()
23     cy.url().should('be.equal', 'http://localhost:4200/users')
24   )
25
26 ]
27 )
```

# 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.

The screenshot shows the Cypress Test Runner interface on the left and a browser window on the right. The browser window displays the 'Users' page of the 'FOODIE INC' admin application. The table lists 10 users with columns for Sr. No, Name, Email, Contact, and Actions (View Orders). The browser's address bar shows 'localhost:4200/users' and its zoom level is set to '1000x660 (44%)'. The Cypress runner shows the test results for 'login-spec.cy.ts' under the 'Admin Web App' section, with three green checkmarks indicating successful tests: 'should have correct labels', 'should login with email and password', and 'should redirect admin user to the dashboard page if they provided correct credentials'.

Sr. No	Name	Email	Contact	Actions
1	John	john@example.com	0	<button>View Orders</button>
2	Fionna Flynn	fionna@example.com	0	<button>View Orders</button>
3	Leo	leo@example.com	0	<button>View Orders</button>
4	Jake Joe	jake@example.com	0	<button>View Orders</button>
5	Kim Shawn	kim@example.com	0	<button>View Orders</button>
6	Duke Dan	duke@example.com	0	<button>View Orders</button>
7	Jake Joe	jake@example.com	0	<button>View Orders</button>
8	Kim Shawn	kim@example.com	0	<button>View Orders</button>
9	Duke Dan	duke@example.com	0	<button>View Orders</button>
10	Jake Joe	jake@example.com	0	<button>View Orders</button>

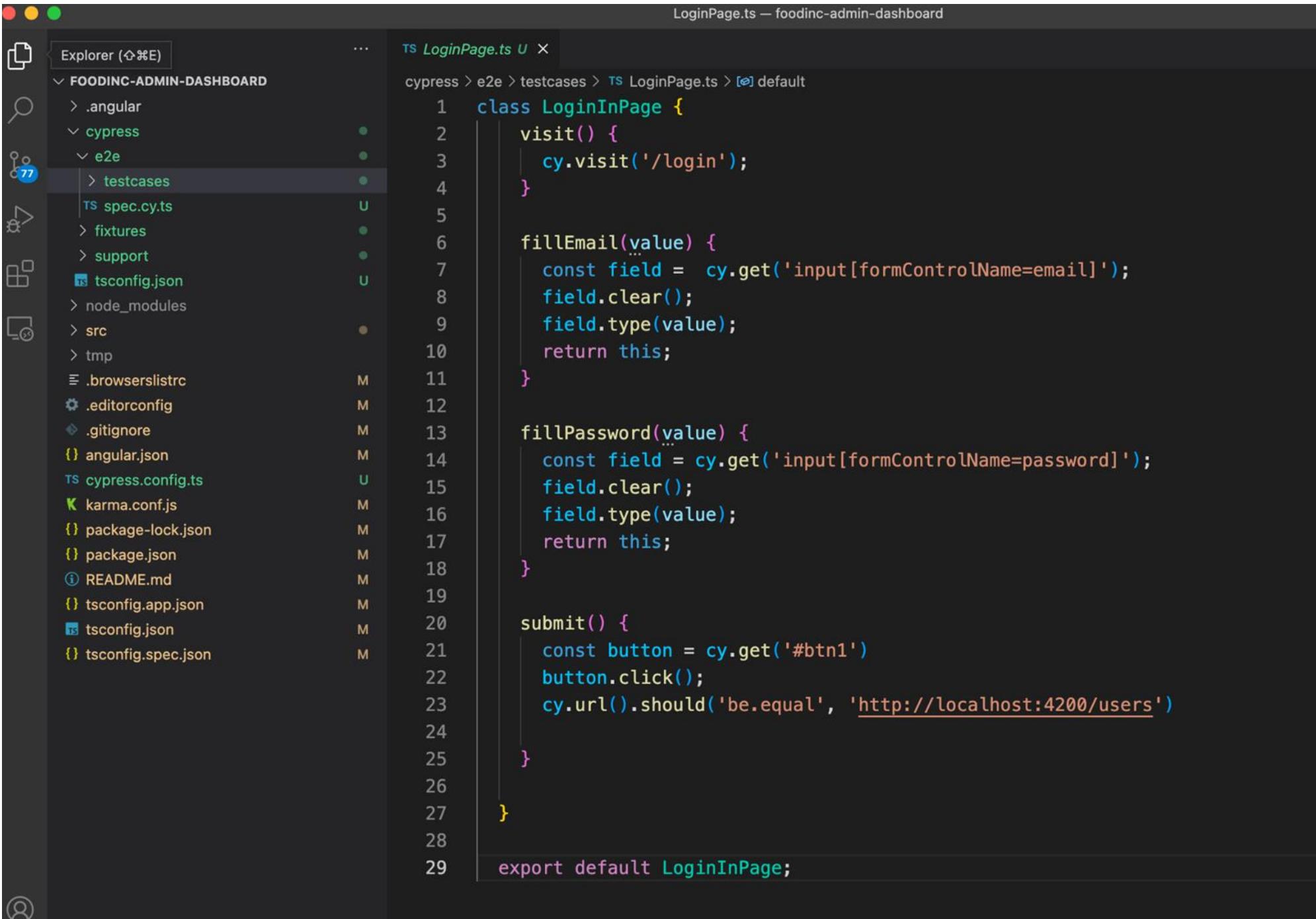
## Page Object Model for Login Page

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

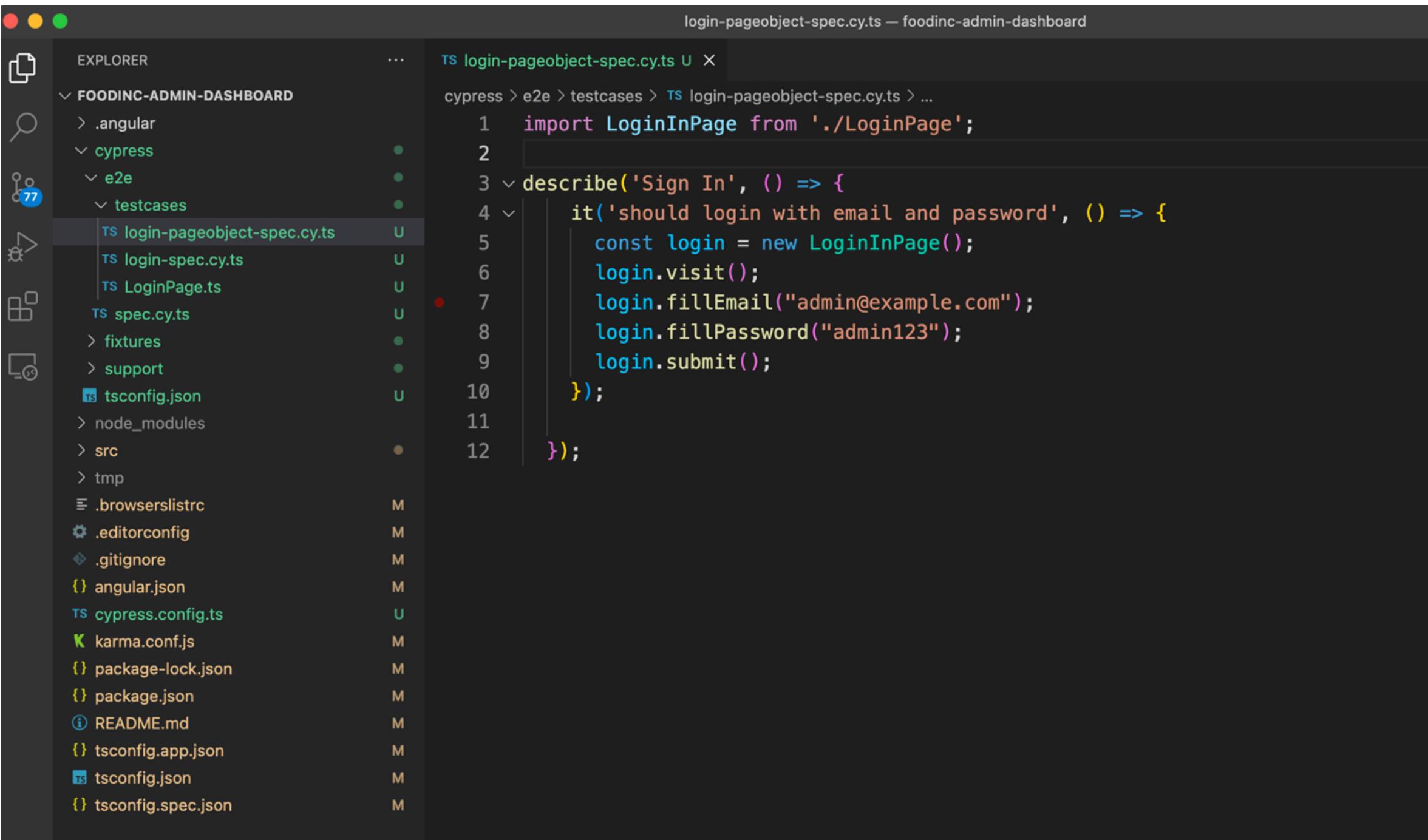


The screenshot shows a dark-themed instance of Visual Studio Code. On the left is the Explorer sidebar, which lists the project structure under 'FOODINC-ADMIN-DASHBOARD'. The 'testcases' folder is selected, containing files like 'spec.cy.ts', 'fixtures', 'support', 'tsconfig.json', and 'LoginPage.ts'. The main editor area displays the content of 'LoginPage.ts'.

```
LoginPage.ts — foodinc-admin-dashboard
cypress > e2e > testcases > LoginPage.ts > default
1  class LoginInPage {
2    visit() {
3      cy.visit('/login');
4    }
5
6    fillEmail(value) {
7      const field = cy.get('input[formControlName=email]');
8      field.clear();
9      field.type(value);
10     return this;
11   }
12
13   fillPassword(value) {
14     const field = cy.get('input[formControlName=password]');
15     field.clear();
16     field.type(value);
17     return this;
18   }
19
20   submit() {
21     const button = cy.get('#btn1')
22     button.click();
23     cy.url().should('be.equal', 'http://localhost:4200/users')
24   }
25
26 }
27
28
29 export default LoginInPage;
```

# Page Object Model for Login Page

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



```
login-pageobject-spec.cy.ts — foodinc-admin-dashboard
cypress > e2e > testcases > TS login-pageobject-spec.cy.ts > ...
1 import LoginPage from './LoginPage';
2
3 ∵ describe('Sign In', () => {
4   ∵ it('should login with email and password', () => {
5     const login = new LoginPage();
6     login.visit();
7     login.fillEmail("admin@example.com");
8     login.fillPassword("admin123");
9     login.submit();
10   });
11
12 });

EXPLORER
FOODINC-ADMIN-DASHBOARD
> .angular
< cypress
  < e2e
    < testcases
      TS login-pageobject-spec.cy.ts
      TS login-spec.cy.ts
      TS LoginPage.ts
      TS spec.cy.ts
    > fixtures
    > support
    TS tsconfig.json
  > node_modules
  > src
  > tmp
  .browserslistrc
  .editorconfig
  .gitignore
  {} angular.json
  TS cypress.config.ts
  K karma.conf.js
  {} package-lock.json
  {} package.json
  ⓘ README.md
  {} tsconfig.app.json
  TS tsconfig.json
  {} tsconfig.spec.json
```

# 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.

The screenshot shows the Cypress Test Runner interface. On the left, the test tree is visible with the file `login-pageobject-spec.cy.ts` selected. The main area displays the test code:

```
1 visit '/login'
2 get input[formControlName=email]
3 -clear
4 -type admin@example.com
5 get input[formControlName=password]
6 -clear
7 -type admin123
8 get #btn1
9 -click
(xhr) POST 200
http://localhost:8080/adminauth/login
(new url) http://localhost:4200/users
(xhr) GET 200 http://localhost:8080/users/get
10 url
11 -assert expected http://localhost:4200/users
to equal http://localhost:4200/users
```

To the right, a browser window shows a user management application titled "FOODIE INC". The "Users" table lists 10 entries:

Sr. No	Name	Email	Contact	Actions
1	John	john@example.com	0	<button>View Orders</button>
2	Fionna Flynn	fionna@example.com	0	<button>View Orders</button>
3	Leo	leo@example.com	0	<button>View Orders</button>
4	Jake Joe	jake@example.com	0	<button>View Orders</button>
5	Kim Shawn	kim@example.com	0	<button>View Orders</button>
6	Duke Dan	duke@example.com	0	<button>View Orders</button>
7	Jake Joe	jake@example.com	0	<button>View Orders</button>
8	Kim Shawn	kim@example.com	0	<button>View Orders</button>
9	Duke Dan	duke@example.com	0	<button>View Orders</button>
10	Jake Joe	jake@example.com	0	<button>View Orders</button>

# 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.

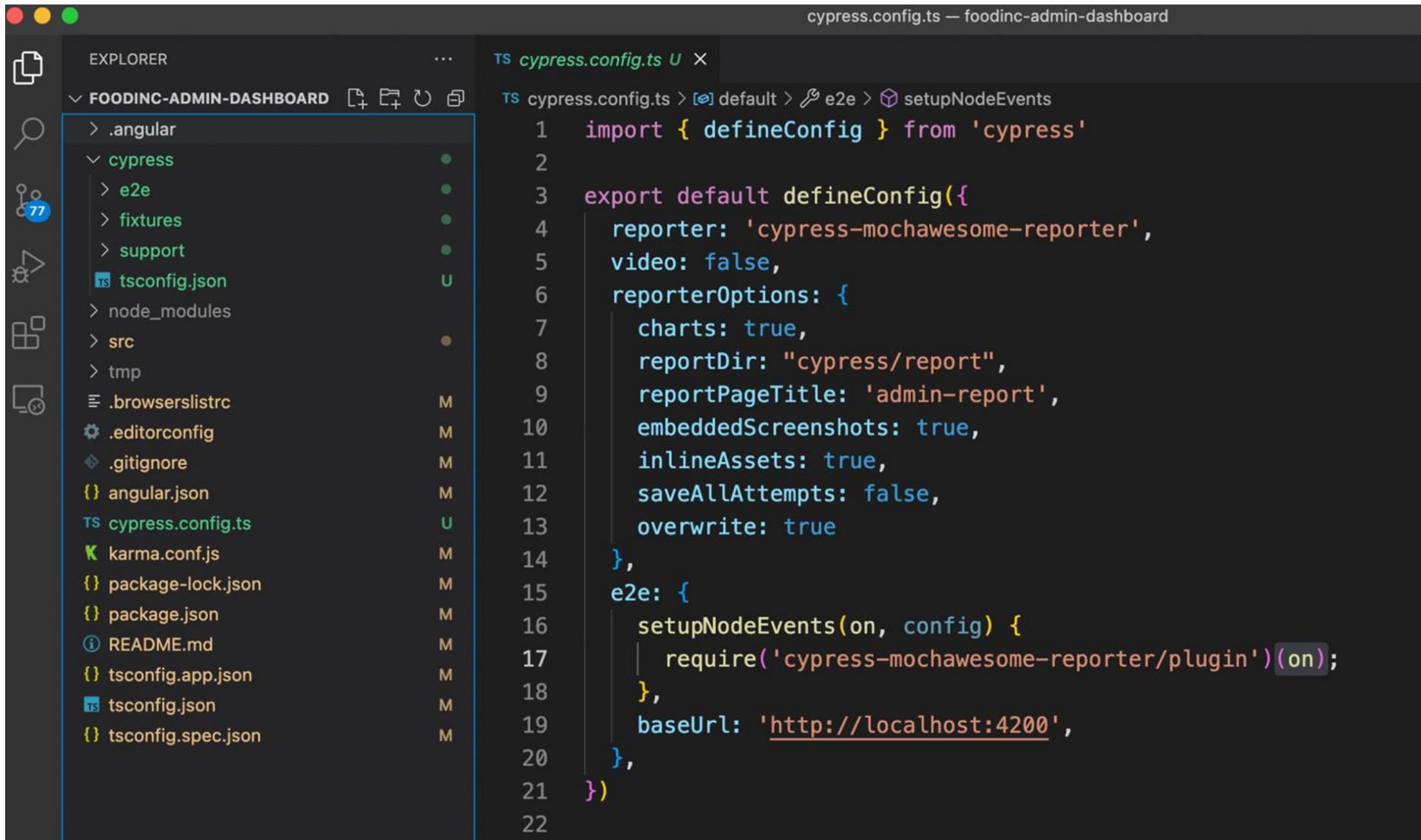
Use the below command:

**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

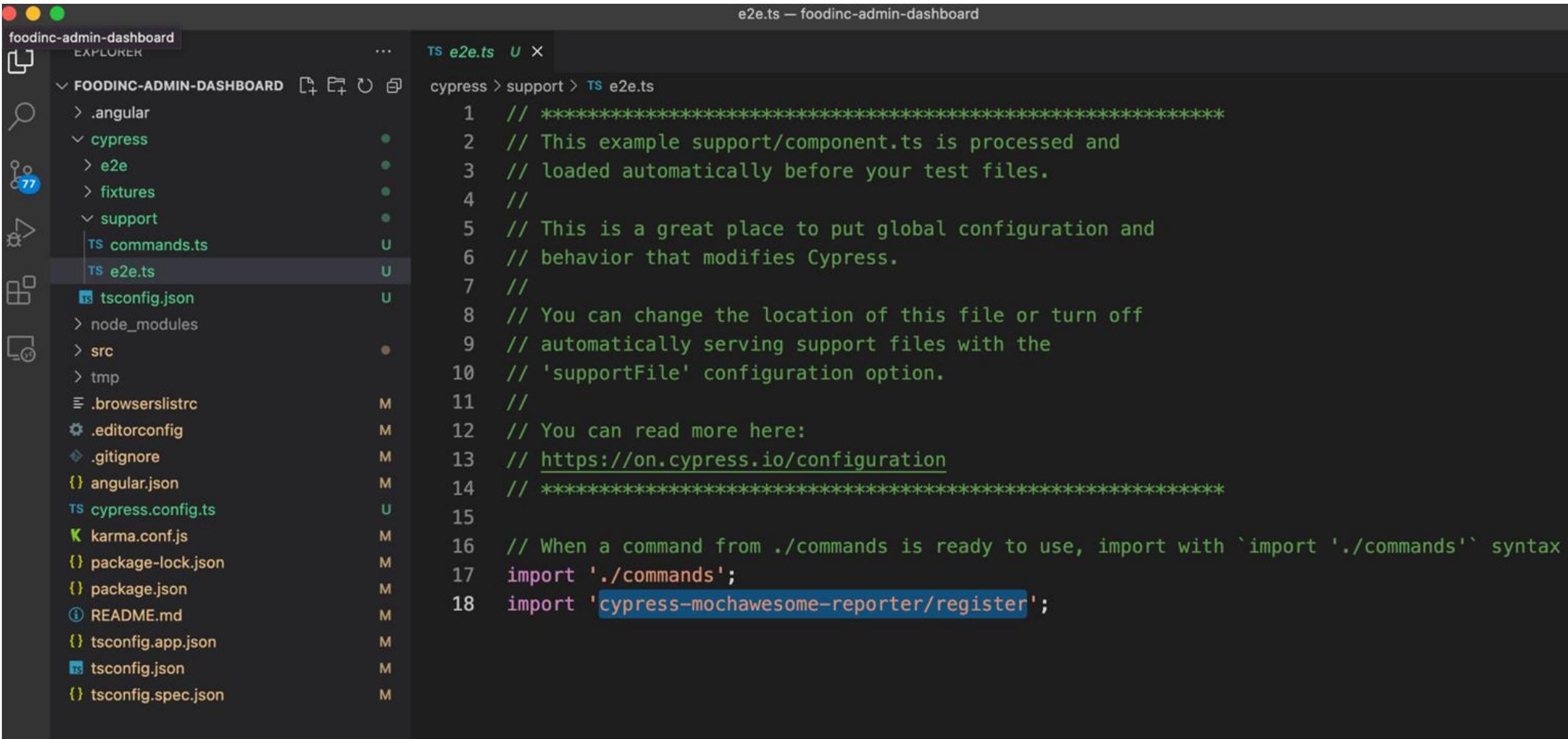
Now, configure reporter tool in your cypress.config.ts file. Add reporter options in the configuration as per your need.



```
cypress.config.ts — foodinc-admin-dashboard
cypress.config.ts — foodinc-admin-dashboard
TS cypress.config.ts > [o] default > e2e > setupNodeEvents
1 import { defineConfig } from 'cypress'
2
3 export default defineConfig({
4   reporter: 'cypress-mochawesome-reporter',
5   video: false,
6   reporterOptions: {
7     charts: true,
8     reportDir: "cypress/report",
9     reportPageTitle: 'admin-report',
10    embeddedScreenshots: true,
11    inlineAssets: true,
12    saveAllAttempts: false,
13    overwrite: true
14  },
15  e2e: {
16    setupNodeEvents(on, config) {
17      require('cypress-mochawesome-reporter/plugin')(on);
18    },
19    baseUrl: 'http://localhost:4200',
20  },
21})
22
```

# Cypress Mochawesome Reporter

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



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "FOODINC-ADMIN-DASHBOARD". The "support" folder contains "commands.ts" and "e2e.ts".
- Code Editor:** The file "e2e.ts" is open, showing its content. The code includes comments explaining the support file's purpose and configuration options, followed by an import statement for the Mochawesome reporter.
- Code Content:**

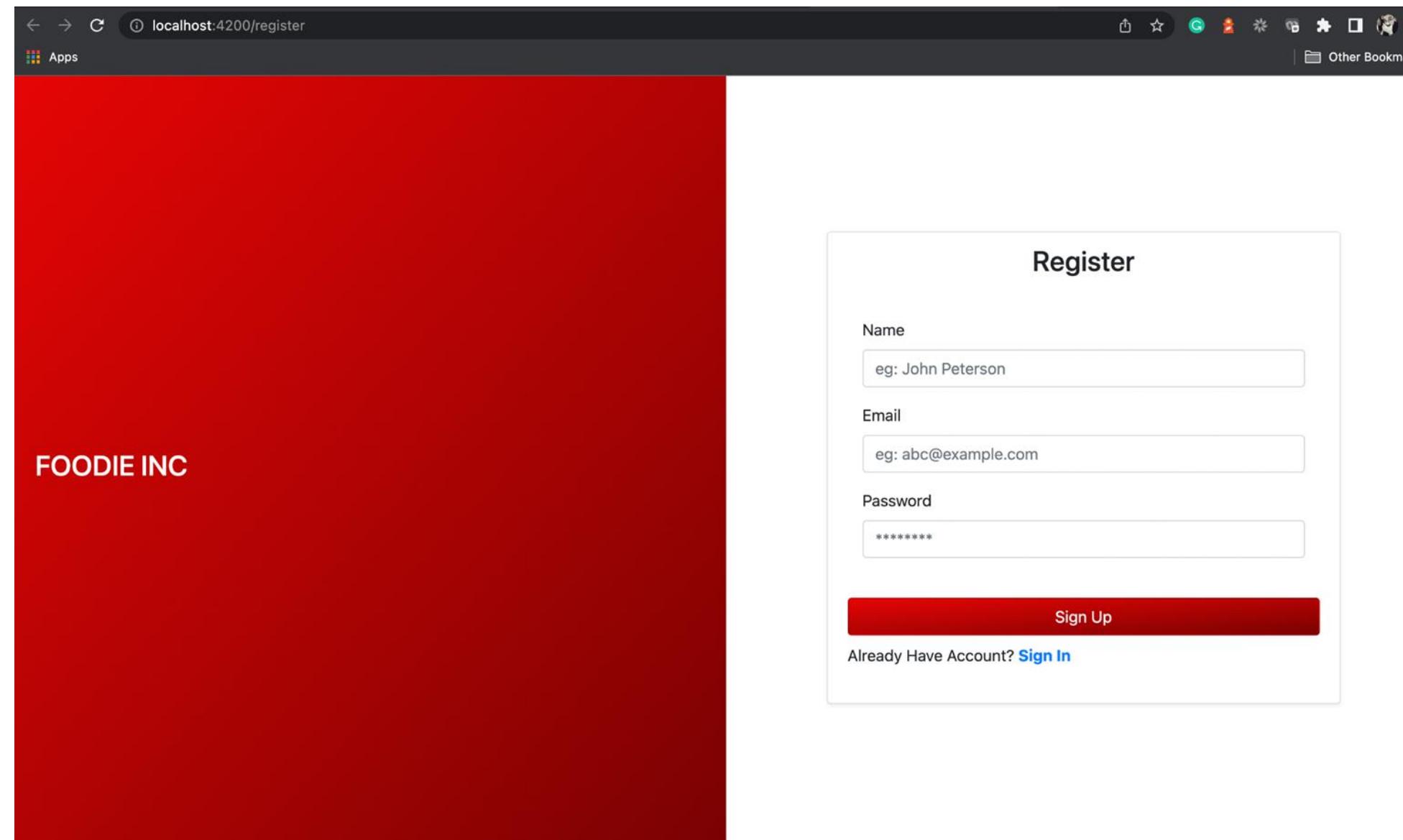
```
// ****
// This example support/component.ts is processed and
// loaded automatically before your test files.
//
// This is a great place to put global configuration and
// behavior that modifies Cypress.
//
// You can change the location of this file or turn off
// automatically serving support files with the
// 'supportFile' configuration option.
//
// You can read more here:
// https://on.cypress.io/configuration
// ****
// When a command from ./commands is ready to use, import with `import './commands'` syntax
import './commands';
import 'cypress-mochawesome-reporter/register';
```

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



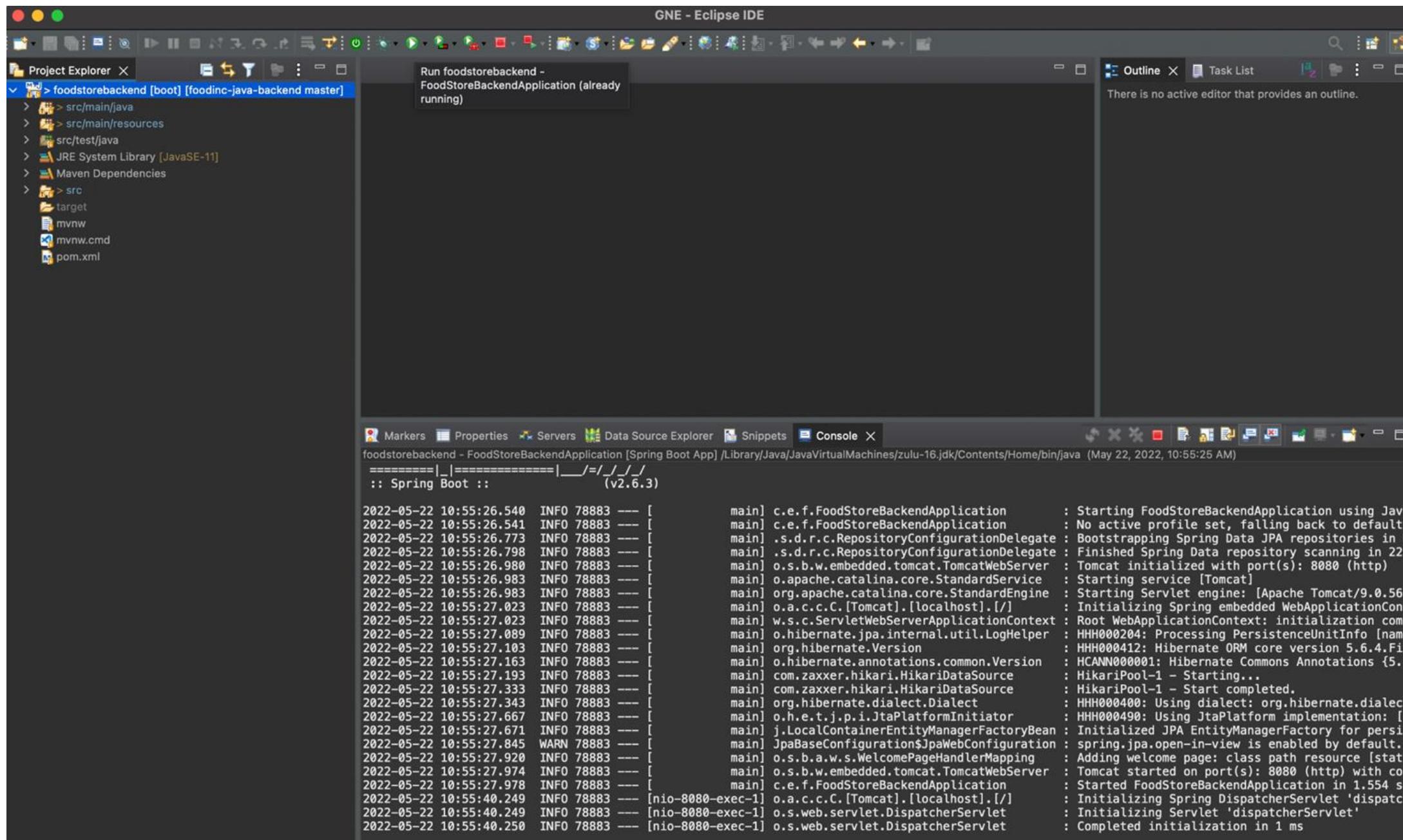
# Build and execute End User Web App Project

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**



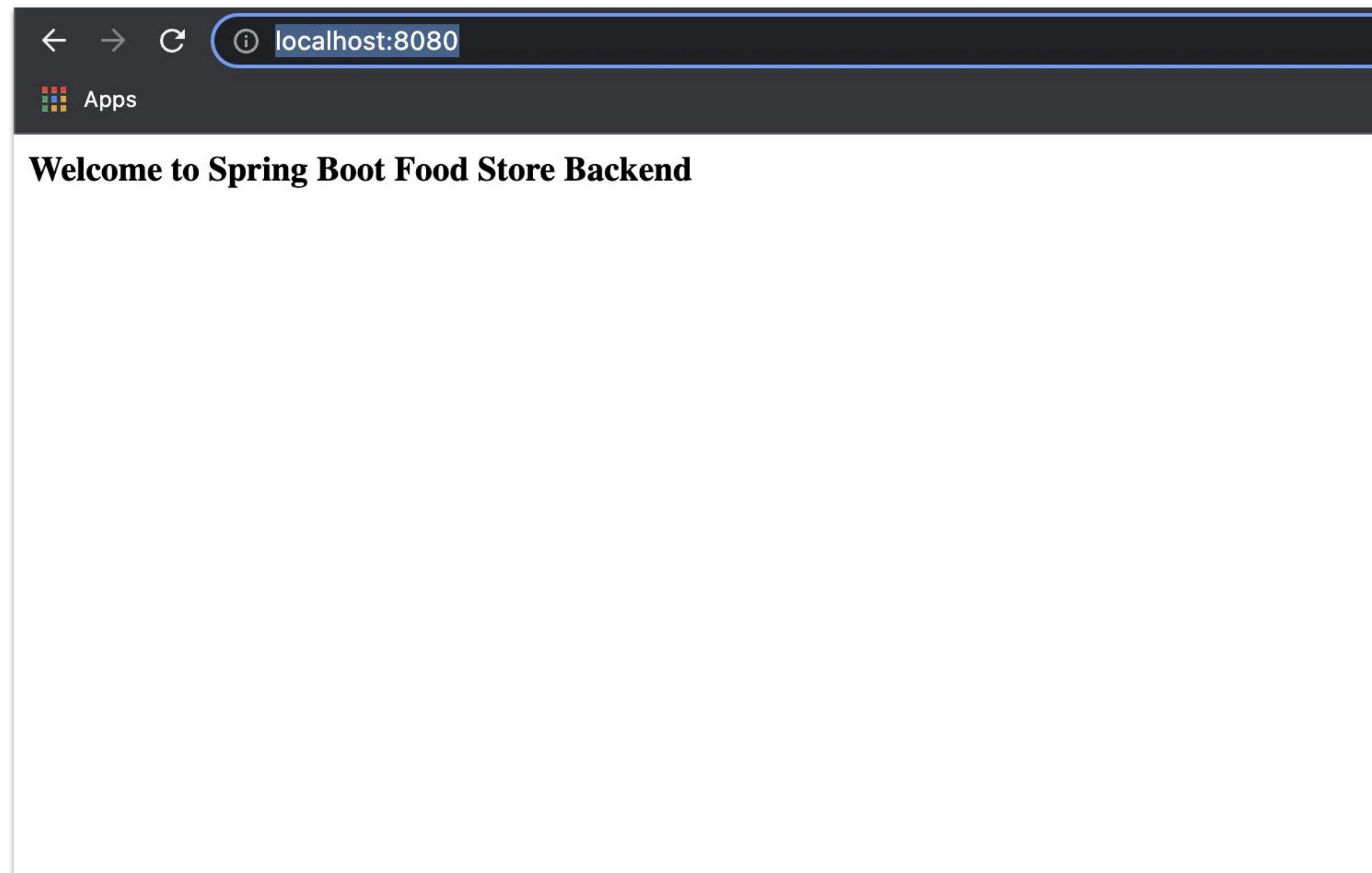
The screenshot shows the Eclipse IDE interface with a dark theme. The Project Explorer view on the left displays a Java project named "foodstorebackend". The Console view at the bottom shows the application's startup logs:

```
=====[|_]======|__=/__/_/
:: Spring Boot ::          (v2.6.3)

2022-05-22 10:55:26.540  INFO 78883 --- [           main] c.e.f.FoodStoreBackendApplication      : Starting FoodStoreBackendApplication using Java
2022-05-22 10:55:26.541  INFO 78883 --- [           main] c.e.f.FoodStoreBackendApplication      : No active profile set, falling back to default: []
2022-05-22 10:55:26.773  INFO 78883 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate: Bootstrapping Spring Data JPA repositories in D
2022-05-22 10:55:26.798  INFO 78883 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate: Finished Spring Data repository scanning in 22 m
2022-05-22 10:55:26.980  INFO 78883 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat initialized with port(s): 8080 (http)
2022-05-22 10:55:26.983  INFO 78883 --- [           main] o.apache.catalina.core.StandardService: Starting service [Tomcat]
2022-05-22 10:55:26.983  INFO 78883 --- [           main] org.apache.catalina.core.StandardEngine: Starting Servlet engine: [Apache Tomcat/9.0.56]
2022-05-22 10:55:27.023  INFO 78883 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]: Initializing Spring embedded WebApplicationContext
2022-05-22 10:55:27.023  INFO 78883 --- [           main] w.s.c.ServletWebServerApplicationContext: Root WebApplicationContext: initialization completed in 1 ms
2022-05-22 10:55:27.089  INFO 78883 --- [           main] o.hibernate.jpa.internal.util.LogHelper: HHH000204: Processing PersistenceUnitInfo [name: persistence.xml]
2022-05-22 10:55:27.103  INFO 78883 --- [           main] org.hibernate.Version: HHH000412: Hibernate ORM core version 5.6.4.Final
2022-05-22 10:55:27.163  INFO 78883 --- [           main] o.hibernate.annotations.common.Version: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-05-22 10:55:27.193  INFO 78883 --- [           main] com.zaxxer.hikari.HikariDataSource: HikariPool-1 - Starting...
2022-05-22 10:55:27.333  INFO 78883 --- [           main] com.zaxxer.hikari.HikariDataSource: HikariPool-1 - Start completed.
2022-05-22 10:55:27.343  INFO 78883 --- [           main] org.hibernate.dialect.Dialect: HHH000400: Using dialect: org.hibernate.dialect.
2022-05-22 10:55:27.667  INFO 78883 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator: HHH000490: Using JtaPlatform implementation: [org
2022-05-22 10:55:27.671  INFO 78883 --- [           main] j.LocalContainerEntityManagerFactoryBean: Initialized JPA EntityManagerFactory for persist
2022-05-22 10:55:27.845  WARN 78883 --- [           main] JpaBaseConfiguration$JpaWebConfiguration: spring.jpa.open-in-view is enabled by default. T
2022-05-22 10:55:27.920  INFO 78883 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping: Adding welcome page: class path resource [static
2022-05-22 10:55:27.974  INFO 78883 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8080 (http) with cont
2022-05-22 10:55:27.978  INFO 78883 --- [           main] c.e.f.FoodStoreBackendApplication: Started FoodStoreBackendApplication in 1.554 sec
2022-05-22 10:55:40.249  INFO 78883 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]: Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-05-22 10:55:40.249  INFO 78883 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet: Initializing Servlet 'dispatcherServlet'
2022-05-22 10:55:40.250  INFO 78883 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet: Completed initialization in 1 ms
```

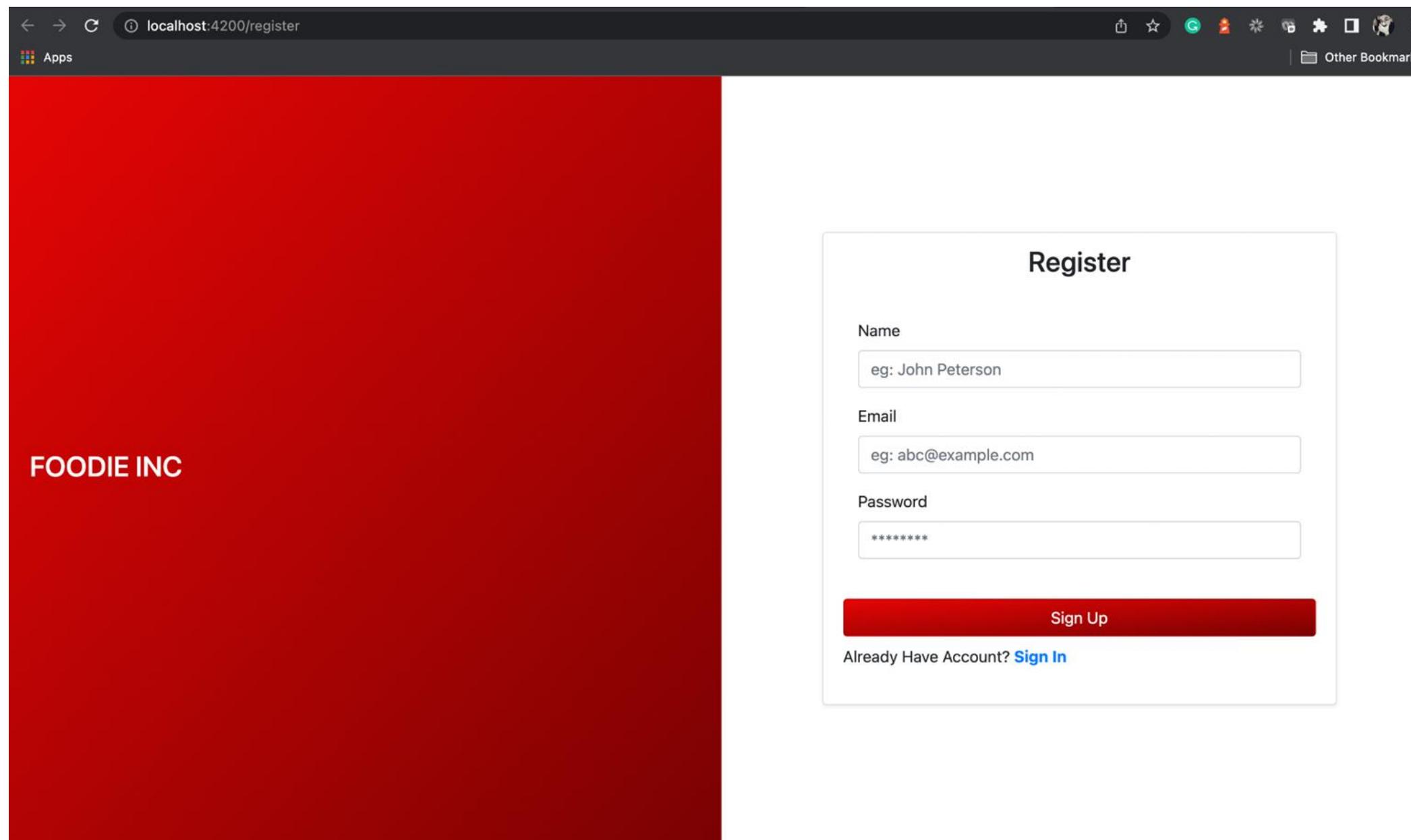
# Check Java Backend Project Status as Running

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



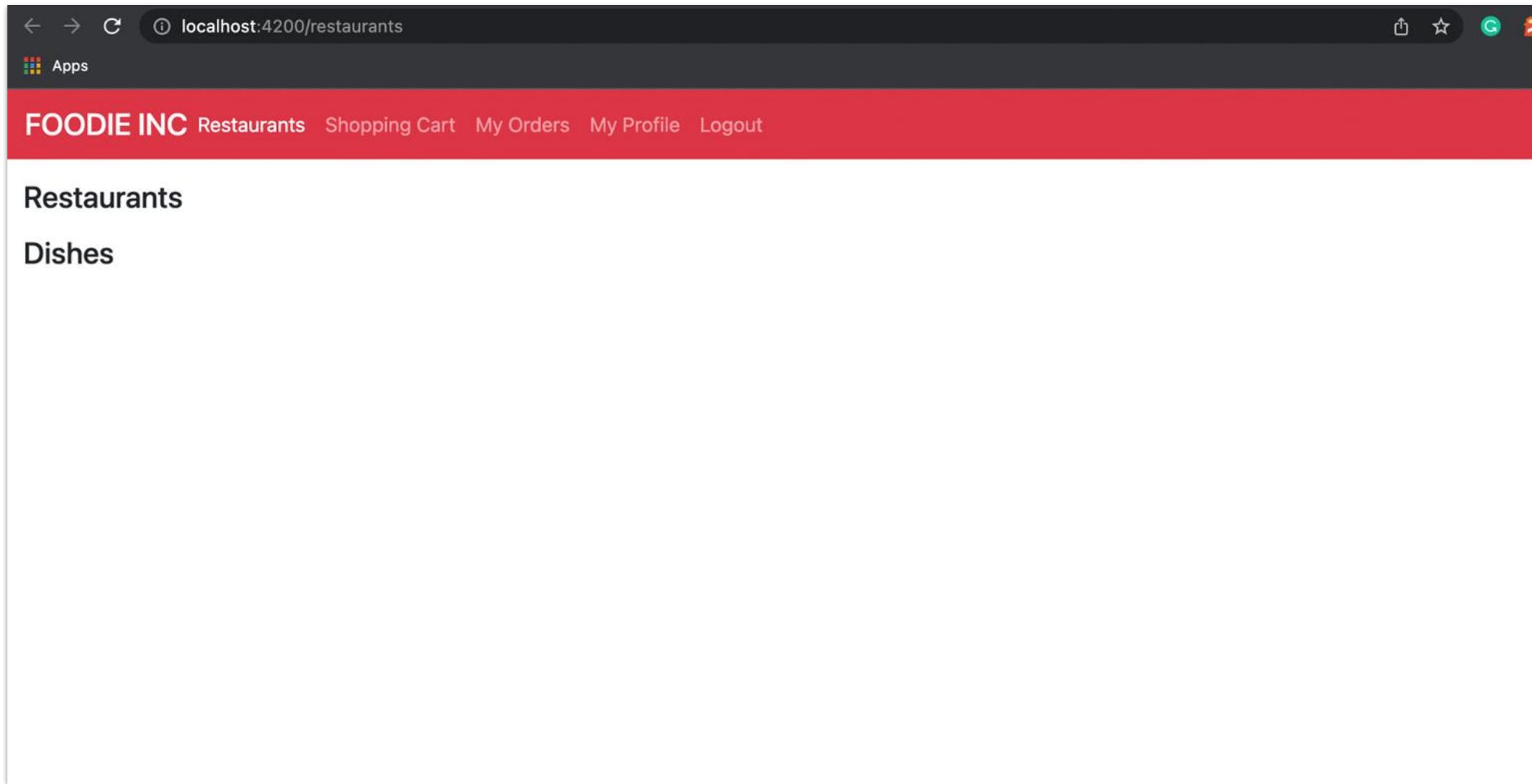
# Register as a new User

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



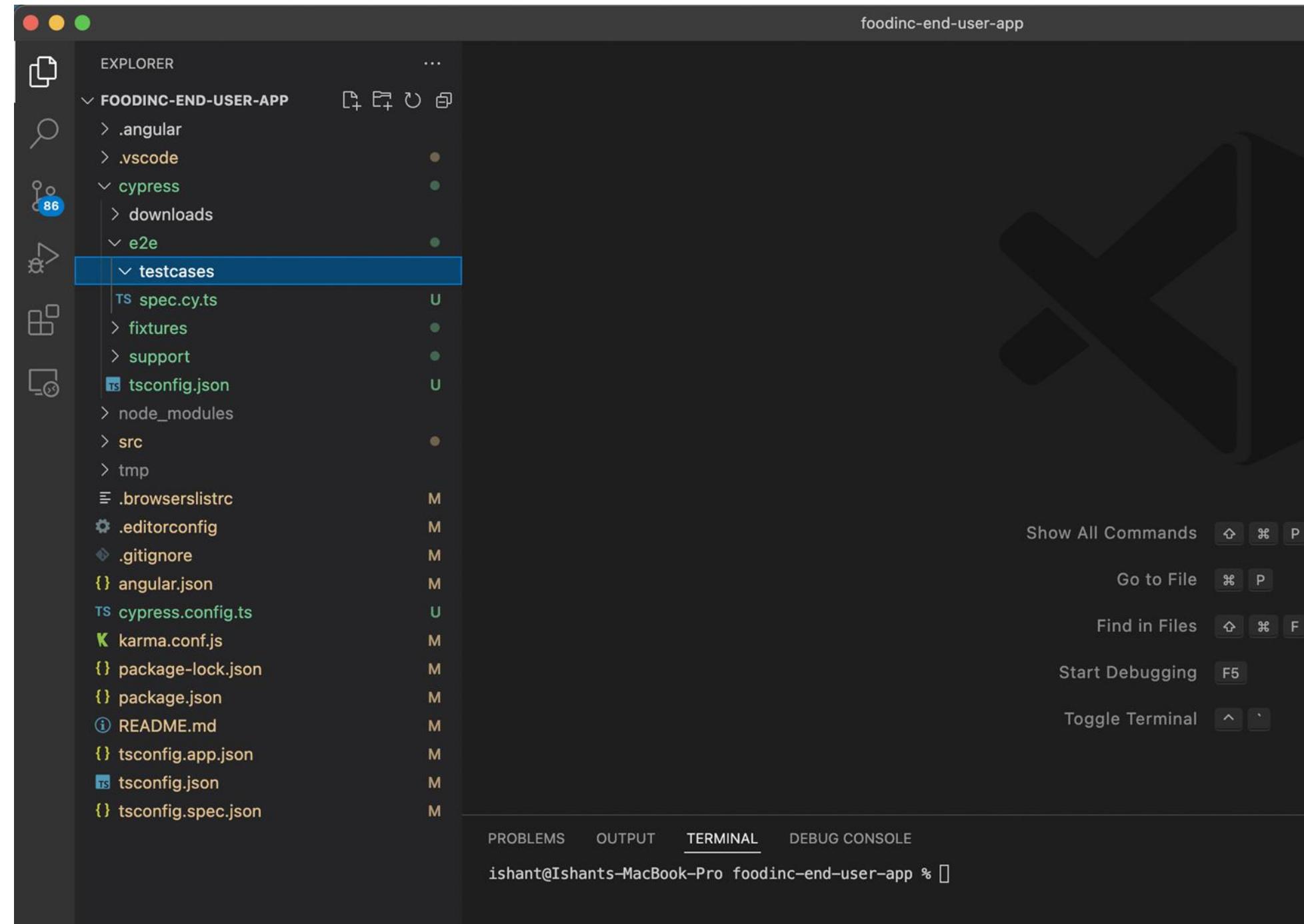
# Write Test Case for Register Page

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**

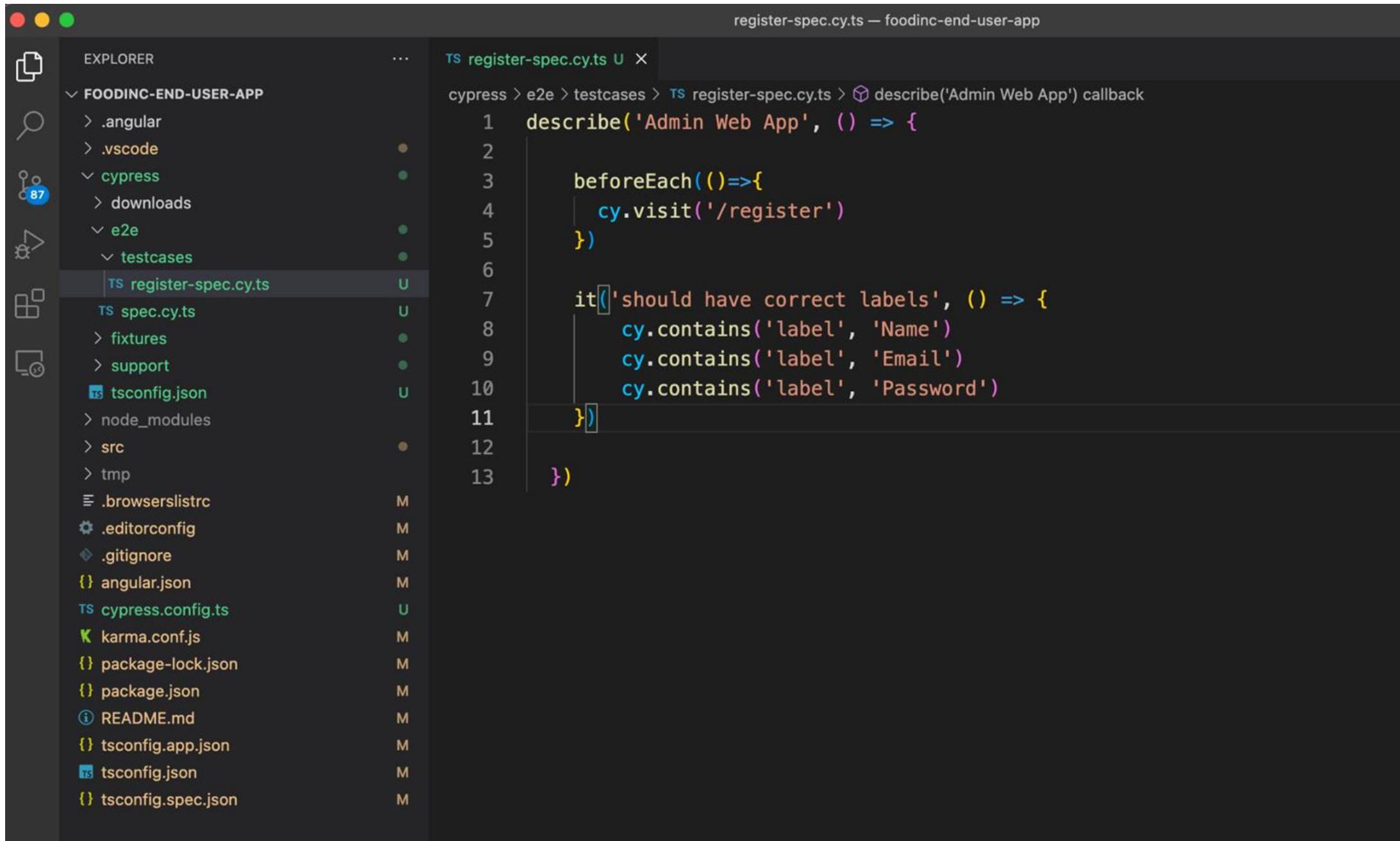
In order to check the labels, we will use CSS Locator. Right click on your web app and open elements to find the css class for the labels.

The screenshot shows a browser window with the URL `localhost:4200/register`. The main content is a registration form titled "Register". The form has three input fields: "Name" (placeholder: "eg: John Peterson"), "Email" (placeholder: "eg: abc@example.com"), and "Password" (placeholder: "\*\*\*\*\*"). Below the form is a red "Sign Up" button and a link "Already Have Account? [Sign In](#)". The background of the page is red with the text "FOODIE INC". On the right side of the browser, the DevTools Elements tab is open, displaying the DOM structure of the page. The "label.text-sm" class is highlighted in blue, indicating it is the CSS locator used to identify the labels for testing.

# 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**



```
register-spec.cy.ts — foodinc-end-user-app
cypress > e2e > testcases > register-spec.cy.ts > describe('Admin Web App') callback
1  describe('Admin Web App', () => {
2
3    beforeEach(()=>{
4      cy.visit('/register')
5    })
6
7    it('should have correct labels', () => {
8      cy.contains('label', 'Name')
9      cy.contains('label', 'Email')
10     cy.contains('label', 'Password')
11   })
12
13 })
```

# 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.

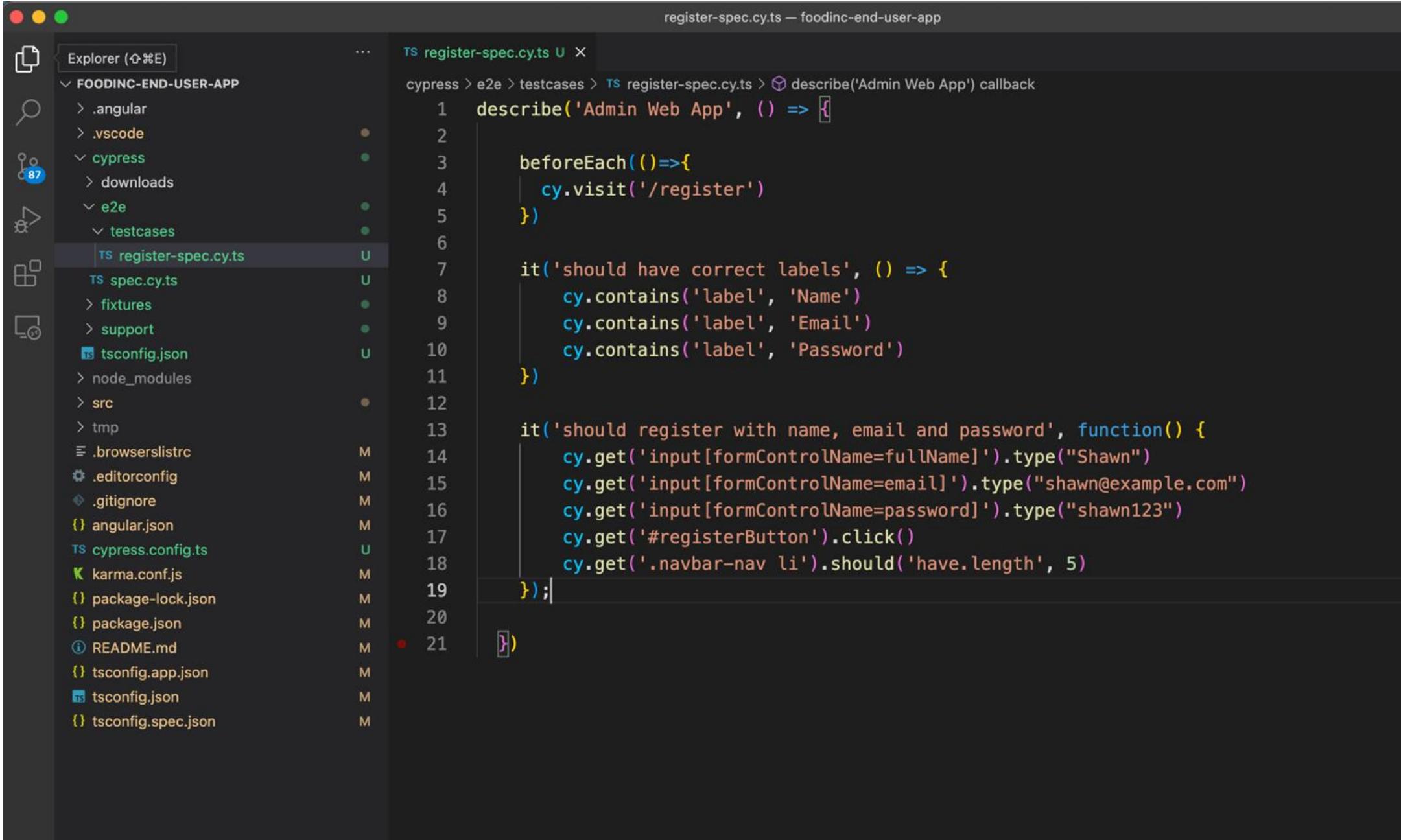
The screenshot shows a browser window with the URL `localhost:4200/register`. The main content is the 'Register' page for 'FOODIE INC'. The page has three input fields: 'Name' (placeholder 'eg: John Peterson'), 'Email' (placeholder 'eg: abc@example.com'), and 'Password' (placeholder '\*\*\*\*\*'). Below the inputs is a red 'Sign Up' button and a link 'Already Have Account? [Sign In](#)'. To the right of the browser is the Chrome DevTools interface, specifically the 'Elements' tab. The DOM tree is expanded to show the HTML structure of the register page. The 'Email' input field is highlighted with a blue selection box, and its corresponding `formControlName` is visible in the code as `email`. The 'Sign Up' button is also highlighted with a blue selection box, and its ID is shown as `#__next`.

```
<app-root _ngcontent-tak-c10 ng-version="13.1.1">
  <router-outlet _ngcontent-tak-c110></router-outlet>
  <app-auth-layout _ngcontent-tak-c109 class="ng-star-inserted">
    <router-outlet _ngcontent-tak-c109></router-outlet>
    <app-register _ngcontent-tak-c118 class="ng-star-inserted">
      <div _ngcontent-tak-c118 class="container-fluid vh-100">
        <div _ngcontent-tak-c118 class="row" style="height: inherit;">
          <div _ngcontent-tak-c118 id="bg" class="col-12 col-sm-6 d-flex flex-column justify-content-center align-items-start px-4" style="background-color: #f0f0f0; height: 100%; position: relative; width: 100%; z-index: 1;">
            <div _ngcontent-tak-c118 class="col-12 col-sm-6 d-flex flex-column justify-content-center align-items-center" style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: #fff; border-radius: 10px; padding: 20px; z-index: 2;">
              <div _ngcontent-tak-c118 class="card w-75 shadow-sm" style="border: 1px solid #ccc; border-radius: 10px; padding: 15px; margin-bottom: 10px;">
                <div _ngcontent-tak-c118 class="card-header bg-white border-0" style="border: none; padding: 0; margin: 0; font-size: 1.2em; font-weight: bold; color: #333; text-align: center;">
                  <h3 _ngcontent-tak-c118 class="text-center m-0">Register</h3>
                </div>
                <div _ngcontent-tak-c118 class="card-body" style="border: none; padding: 0; margin: 0; font-size: 0.9em; color: #666; text-align: center;">
                  <form _ngcontent-tak-c118 novalidate ng-reflect-form="[object Object]" class="ng-untouched ng-pristine ng-invalid">
                    <div _ngcontent-tak-c118 class="form-group col-12 mt-2" style="border: none; margin: 0; padding: 0; position: relative;">
                      <label _ngcontent-tak-c118 for="email" class="text-sm">Name</label>
                      <input _ngcontent-tak-c118 placeholder="eg: John Peterson" formcontrolname="fullName" name="name" type="text" required class="form-control ng-untouched ng-pristine ng-invalid" ng-reflect-name="fullName" ng-reflect-required>
                    </div>
                    <div _ngcontent-tak-c118 class="form-group col-12 mt-2" style="border: none; margin: 0; padding: 0; position: relative;">
                      <label _ngcontent-tak-c118 for="email" class="text-sm">Email</label>
                      <input _ngcontent-tak-c118 placeholder="eg: abc@example.com" formcontrolname="email" name="email" type="email" required class="form-control ng-untouched ng-pristine ng-invalid" ng-reflect-name="email" ng-reflect-required>
                    </div>
                    <div _ngcontent-tak-c118 class="form-group col-12 mt-2" style="border: none; margin: 0; padding: 0; position: relative;">
                      <!--bindings={>
                        <!--ng-reflect-ng-if: "false"-->
                    </div>
                  </form>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </app-register>
  </app-auth-layout>
</app-root>
```

# 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**



The screenshot shows a dark-themed VS Code interface. On the left is the Explorer sidebar with project files like .angular, .vscode, cypress, e2e, testcases, and various configuration files. The main editor area displays a Cypress test file named register-spec.cy.ts. The code is written in TypeScript and uses the Cypress framework. It defines two scenarios: one for checking labels and another for registering with specific data.

```
register-spec.cy.ts — foodinc-end-user-app
cypress > e2e > testcases > TS register-spec.cy.ts > ⚡ describe('Admin Web App') callback
  1  describe('Admin Web App', () => {
  2
  3    beforeEach(()=>{
  4      cy.visit('/register')
  5    })
  6
  7    it('should have correct labels', () => {
  8      cy.contains('label', 'Name')
  9      cy.contains('label', 'Email')
 10      cy.contains('label', 'Password')
 11    })
 12
 13    it('should register with name, email and password', function() {
 14      cy.get('input[formControlName=fullName]').type("Shawn")
 15      cy.get('input[formControlName=email]').type("shawn@example.com")
 16      cy.get('input[formControlName=password]').type("shawn123")
 17      cy.get('#registerButton').click()
 18      cy.get('.navbar-nav li').should('have.length', 5)
 19    });
 20
 21  })
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 698
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 898
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 999
 1000
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1197
 1198
 1199
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1258
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1278
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1297
 1298
 1299
 1299
 1300
 1301
 1302

```

# Write Test Case for Register Page

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

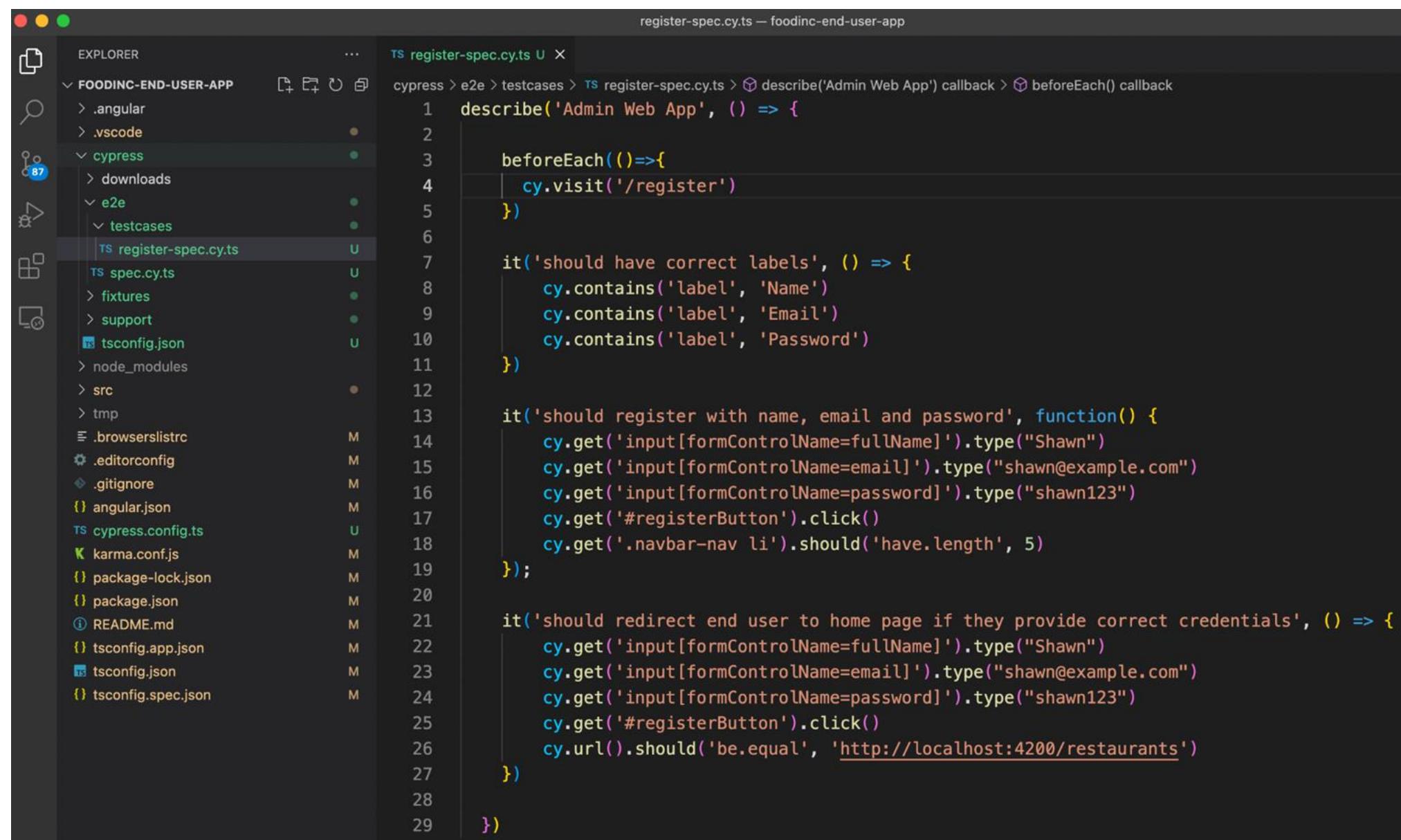
We will now create a scenario **should redirect end user to home page if they provide correct credentials**. In order to send data to input fields, we will use formControlName Locator. Right click on your web app and open elements to find the formControlName and The Button Id.

The screenshot shows a browser window with the URL `localhost:4200/register`. The page title is "FOODIE INC". The main content is a "Register" form with three input fields: "Name" (placeholder "eg: John Peterson"), "Email" (placeholder "eg: abc@example.com"), and "Password" (placeholder "\*\*\*\*\*"). Below the form is a red "Sign Up" button and a link "Already Have Account? [Sign In](#)". To the right of the browser window is the "Elements" tab of the Chrome DevTools, displaying the DOM structure of the page. The "Elements" tab shows the HTML code for the "app-root", "app-auth-layout", "app-register", and various form components like "div.form-group.col-12.mt-2", "label", and "input" with their respective attributes and class names. The "Email" input field is highlighted with a blue border, indicating it is the current element selected in the DevTools.

# 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

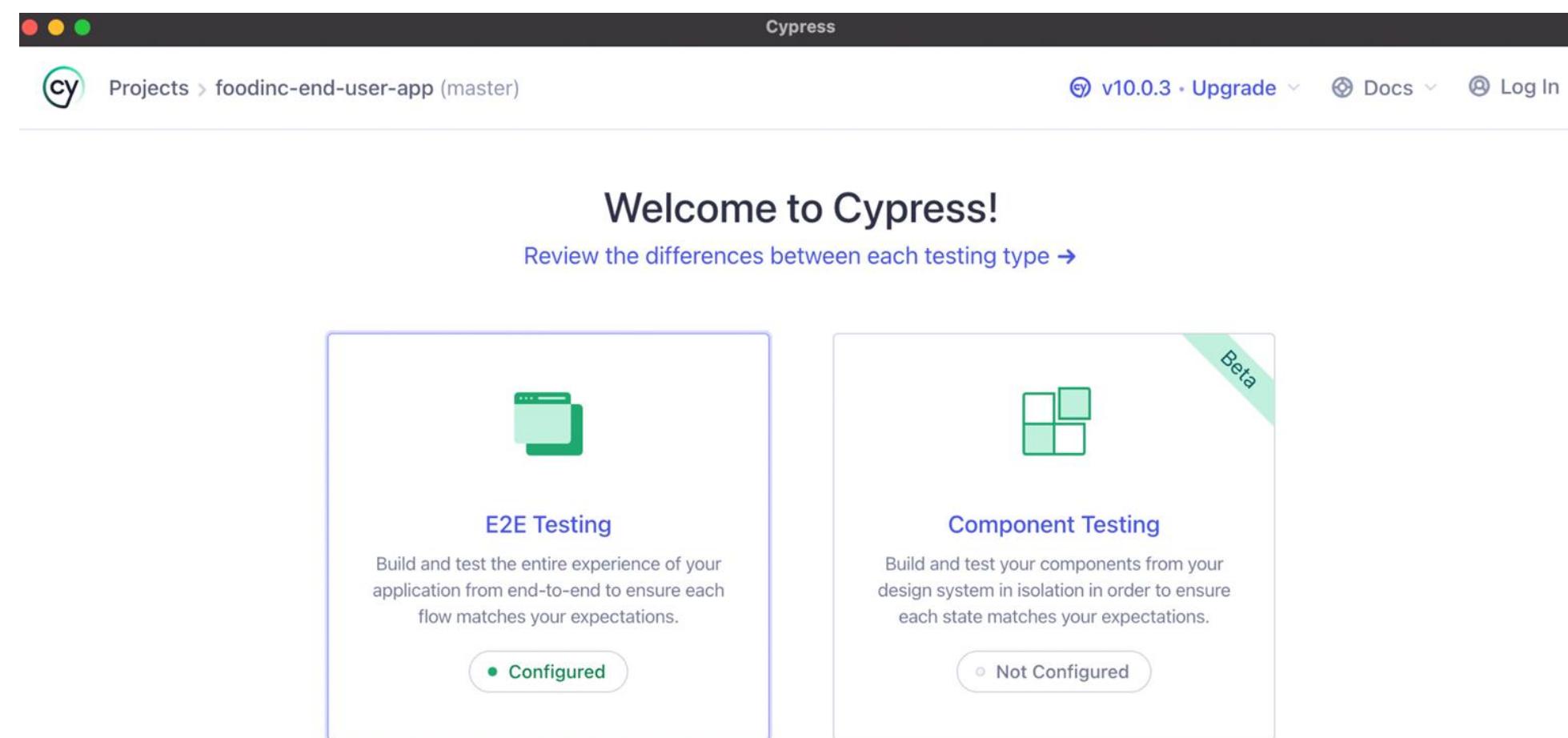


```
register-spec.cy.ts — foodinc-end-user-app
EXPLORER          TS register-spec.cy.ts U X
FOODINC-END-USER-APP
cypress > e2e > testcases > TS register-spec.cy.ts > describe('Admin Web App') callback > beforeEach() callback
1  describe('Admin Web App', () => {
2
3    beforeEach(()=>{
4      cy.visit('/register')
5    })
6
7    it('should have correct labels', () => {
8      cy.contains('label', 'Name')
9      cy.contains('label', 'Email')
10     cy.contains('label', 'Password')
11   })
12
13   it('should register with name, email and password', function() {
14     cy.get('input[formControlName=fullName]').type("Shawn")
15     cy.get('input[formControlName=email]').type("shawn@example.com")
16     cy.get('input[formControlName=password]').type("shawn123")
17     cy.get('#registerButton').click()
18     cy.get('.navbar-nav li').should('have.length', 5)
19   });
20
21   it('should redirect end user to home page if they provide correct credentials', () => {
22     cy.get('input[formControlName=fullName]').type("Shawn")
23     cy.get('input[formControlName=email]').type("shawn@example.com")
24     cy.get('input[formControlName=password]').type("shawn123")
25     cy.get('#registerButton').click()
26     cy.url().should('be.equal', 'http://localhost:4200/restaurants')
27   })
28
29 })
```

# 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.

The screenshot shows the Cypress Test Runner interface. The left panel displays the test structure and code:

```
Specs
  register-spec.cy.ts
    Admin Web App
      ✓ should have correct labels
      ✓ should register with name, email and password
      ✓ should redirect end user to home page if they provide correct credentials
    BEFORE EACH
      1 visit '/register'
    TEST BODY
      1 get input[formControlName=fullName]
      2 -type Shawn
      3 get input[formControlName=email]
      4 -type shawn@example.com
      5 get input[formControlName=password]
      6 -type shawn123
      7 get #registerButton
      8 -click
      (xhr) POST 200
      http://localhost:8080/users/add
      9 url
      10 -assert expected
          http://localhost:4200/restaurants to equal
          http://localhost:4200/restaurants
```

The right panel shows the browser window with the registration page for "FOODIE INC". The page includes fields for Name, Email, and Password, and a "Sign in" link at the bottom.

# 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

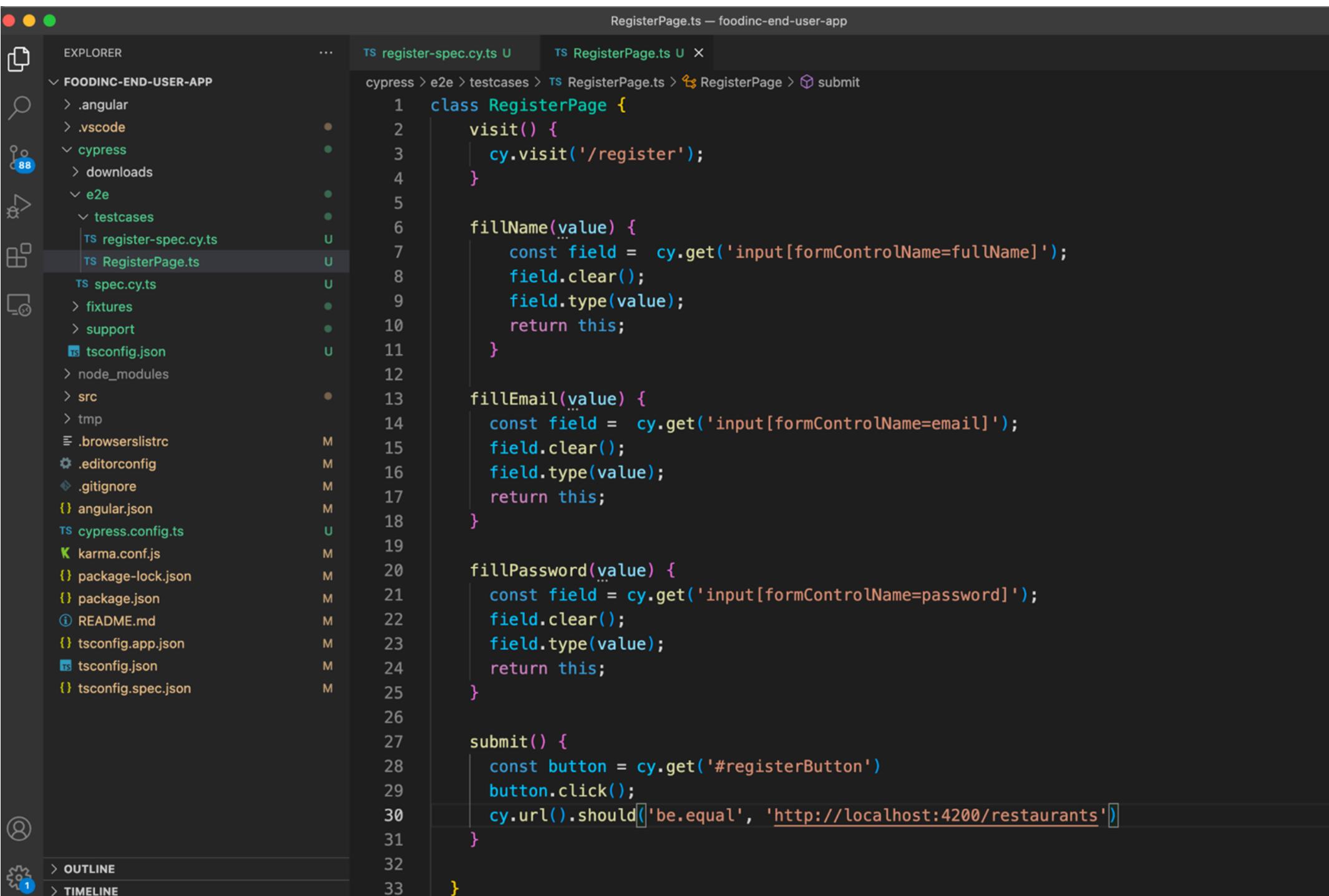
# Page Object Model for Register Page

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



The screenshot shows a dark-themed instance of VS Code. The left sidebar displays the project structure under 'FOODINC-END-USER-APP'. The 'testcases' folder contains three files: 'register-spec.cy.ts', 'RegisterPage.ts', and 'spec.cy.ts'. The 'RegisterPage.ts' file is open in the main editor. It defines a class 'RegisterPage' with methods for visiting the register page, filling name, email, and password fields, and submitting the form. The code uses Cypress commands like 'cy.visit', 'cy.get', and 'cy.url'. The status bar at the bottom indicates there is one uncommitted change.

```
class RegisterPage {
  visit() {
    cy.visit('/register');
  }

  fillName(value) {
    const field = cy.get('input[formControlName=fullName]');
    field.clear();
    field.type(value);
    return this;
  }

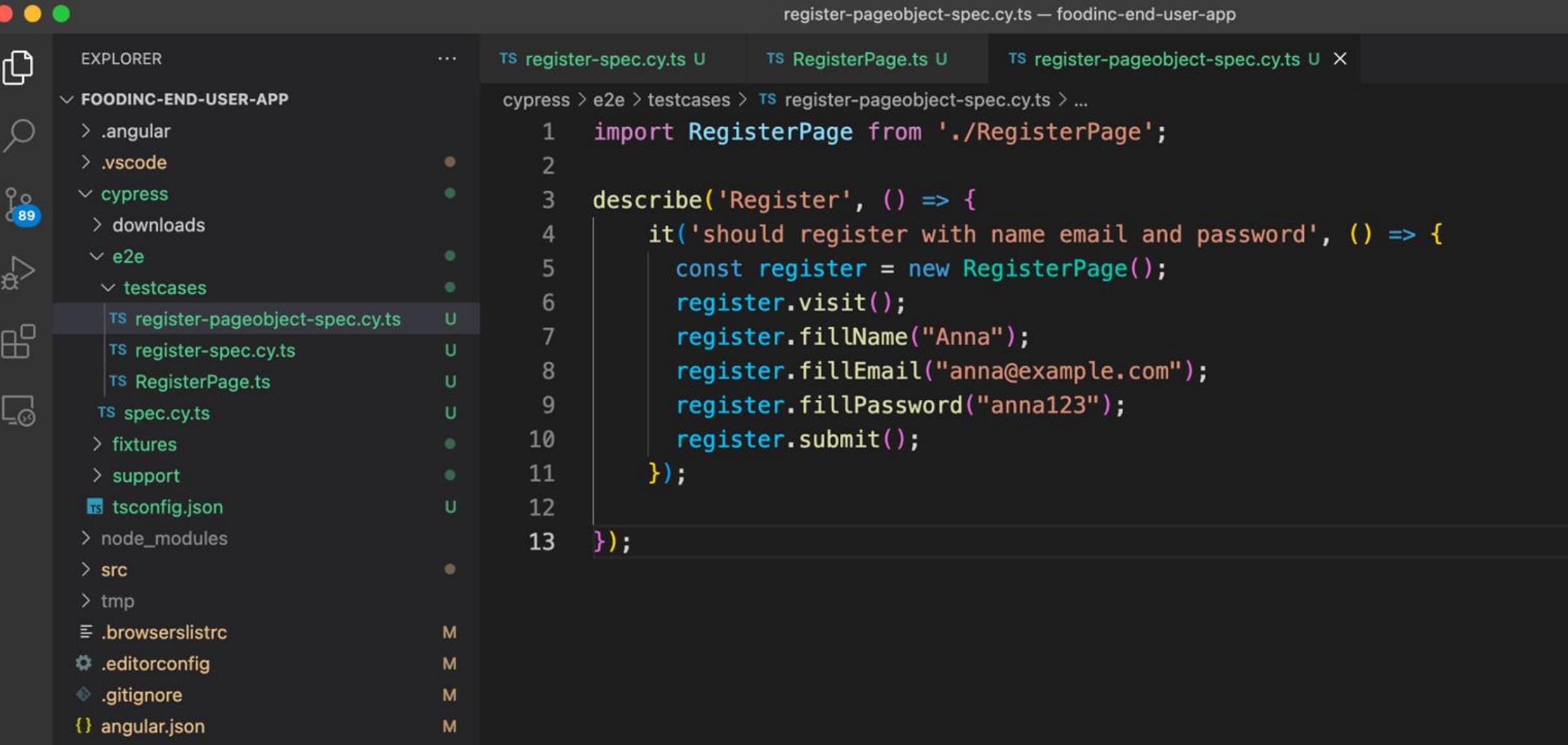
  fillEmail(value) {
    const field = cy.get('input[formControlName=email]');
    field.clear();
    field.type(value);
    return this;
  }

  fillPassword(value) {
    const field = cy.get('input[formControlName=password]');
    field.clear();
    field.type(value);
    return this;
  }

  submit() {
    const button = cy.get('#registerButton')
    button.click();
    cy.url().should('be.equal', 'http://localhost:4200/restaurants')
  }
}
```

# Page Object Model for Register Page

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



```
register-pageobject-spec.cy.ts — foodinc-end-user-app
import RegisterPage from './RegisterPage';

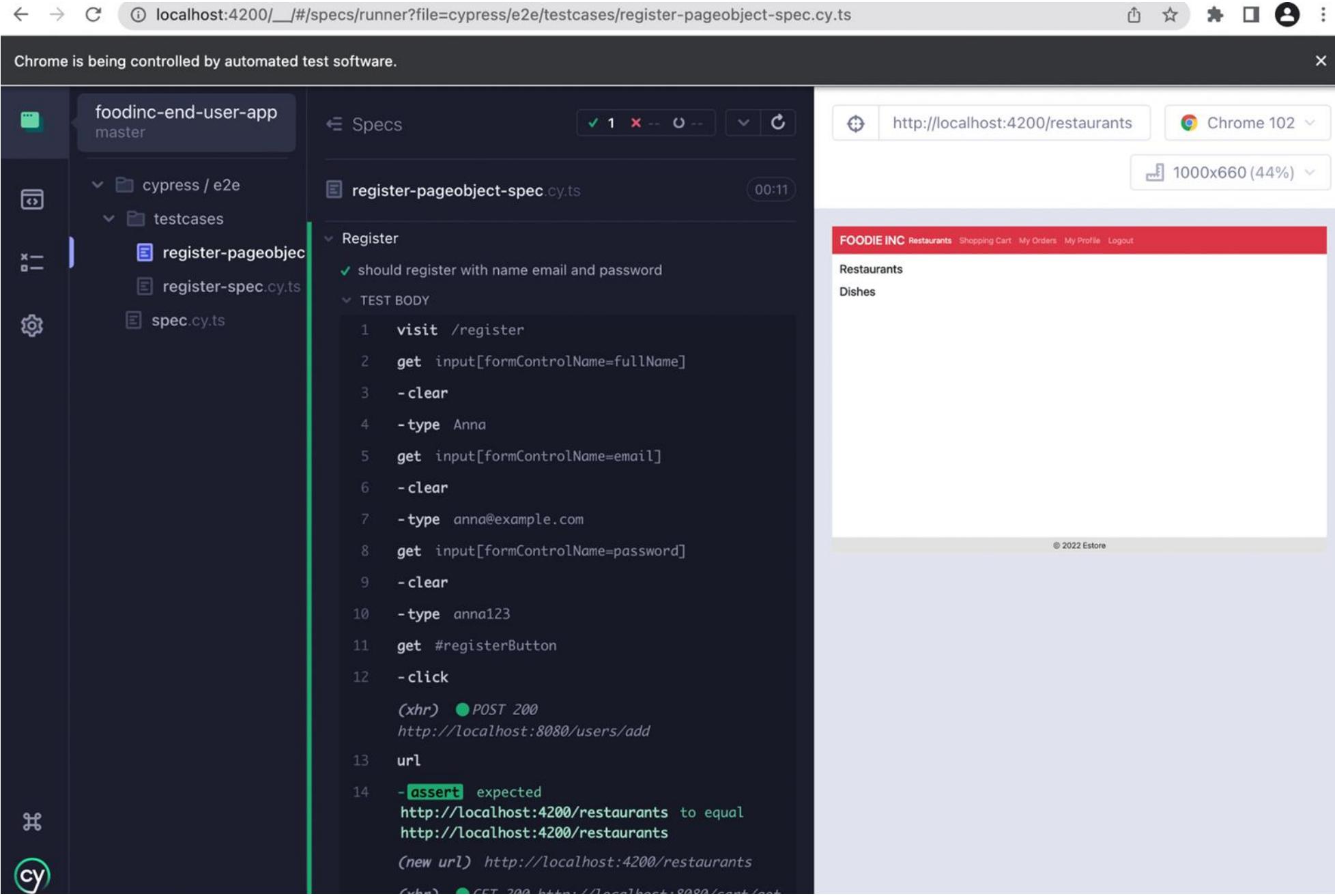
describe('Register', () => {
  it('should register with name email and password', () => {
    const register = new RegisterPage();
    register.visit();
    register.fillName("Anna");
    register.fillEmail("anna@example.com");
    register.fillPassword("anna123");
    register.submit();
  });
});
```

The screenshot shows a dark-themed VS Code interface. On the left is the Explorer sidebar with project files like .angular, .vscode, cypress, e2e, testcases, and tsconfig.json. The file 'TS register-pageobject-spec.cy.ts' is selected in the Explorer. The main editor area displays the code for a Cypress test. The code imports the RegisterPage class, defines a describe block for 'Register', and contains an it block for 'should register with name email and password'. It uses the Page Object Model pattern by creating a new instance of RegisterPage and calling its methods like visit(), fillName(), fillEmail(), fillPassword(), and submit().

# 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.



The screenshot shows the Cypress Test Runner interface. On the left, there's a tree view of the project structure under the 'foodinc-end-user-app' repository. The 'cypress / e2e' folder contains 'testcases', which includes 'register-pageobject-spec.cy.ts' (the current file being run) and 'register-spec.cy.ts'. Below these are 'spec.cy.ts' and a gear icon for settings. The middle pane displays the test code in Cypress syntax:

```
Chrome is being controlled by automated test software.

Specs
register-pageobject-spec.cy.ts 00:11

Register
✓ should register with name email and password
  TEST BODY
    1  visit '/register'
    2  get input[formControlName=fullName]
    3  -clear
    4  -type Anna
    5  get input[formControlName=email]
    6  -clear
    7  -type anna@example.com
    8  get input[formControlName=password]
    9  -clear
   10 -type anna123
   11 get #registerButton
   12 -click
      (xhr) POST 200
      http://localhost:8080/users/add
   13 url
   14 -assert expected
      http://localhost:4200/restaurants to equal
      http://localhost:4200/restaurants
      (new url) http://localhost:4200/restaurants
      (cy) GET 200 https://localhost:8080/test/test
```

The right pane shows a screenshot of the 'FOODIE INC' application. The header has 'FOODIE INC Restaurants Shopping Cart My Orders My Profile Logout'. The sidebar has 'Restaurants' and 'Dishes' options. The main content area is mostly blank with a small footer note '© 2022 Estore'.

# 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.

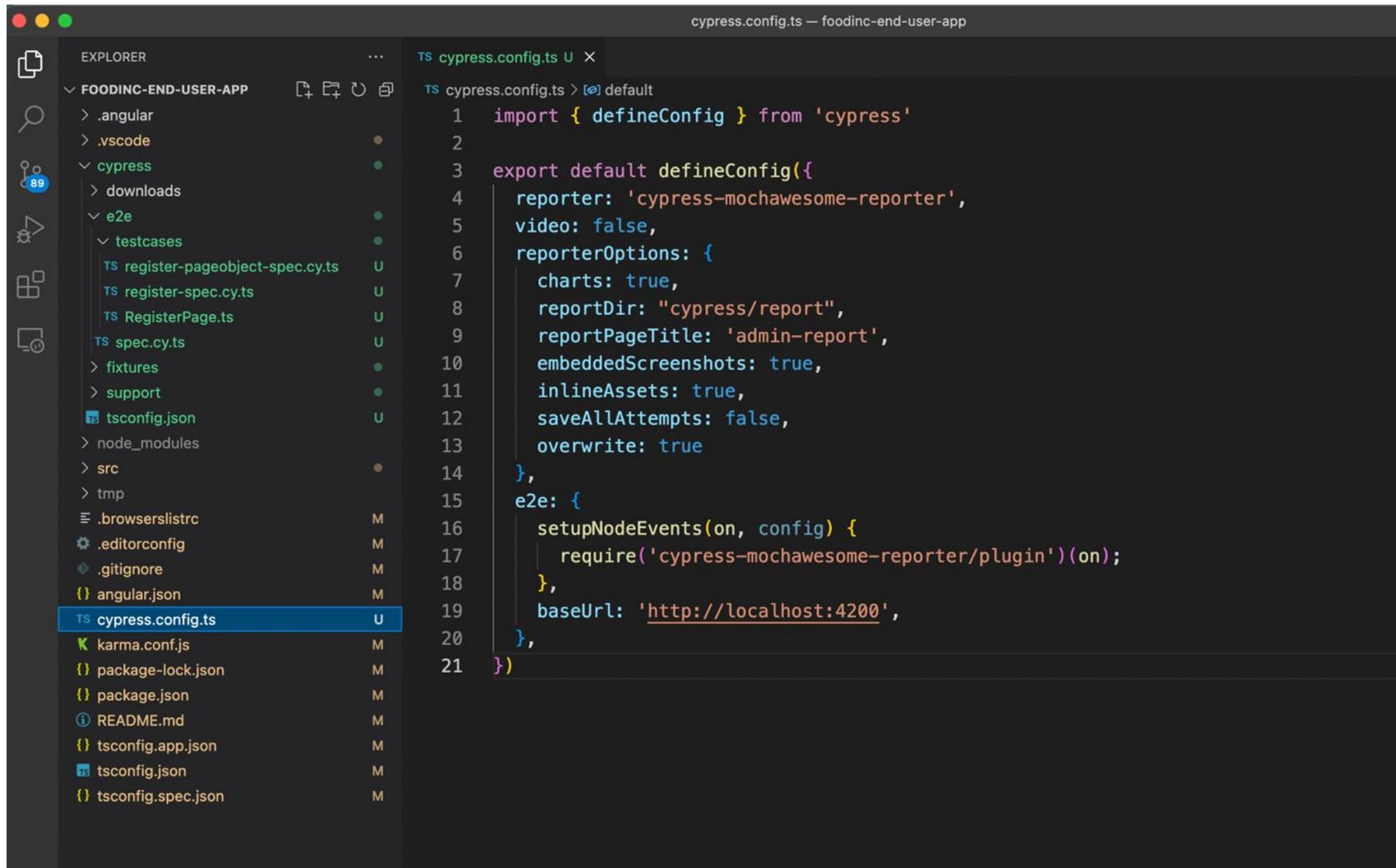
Use the below command:

**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

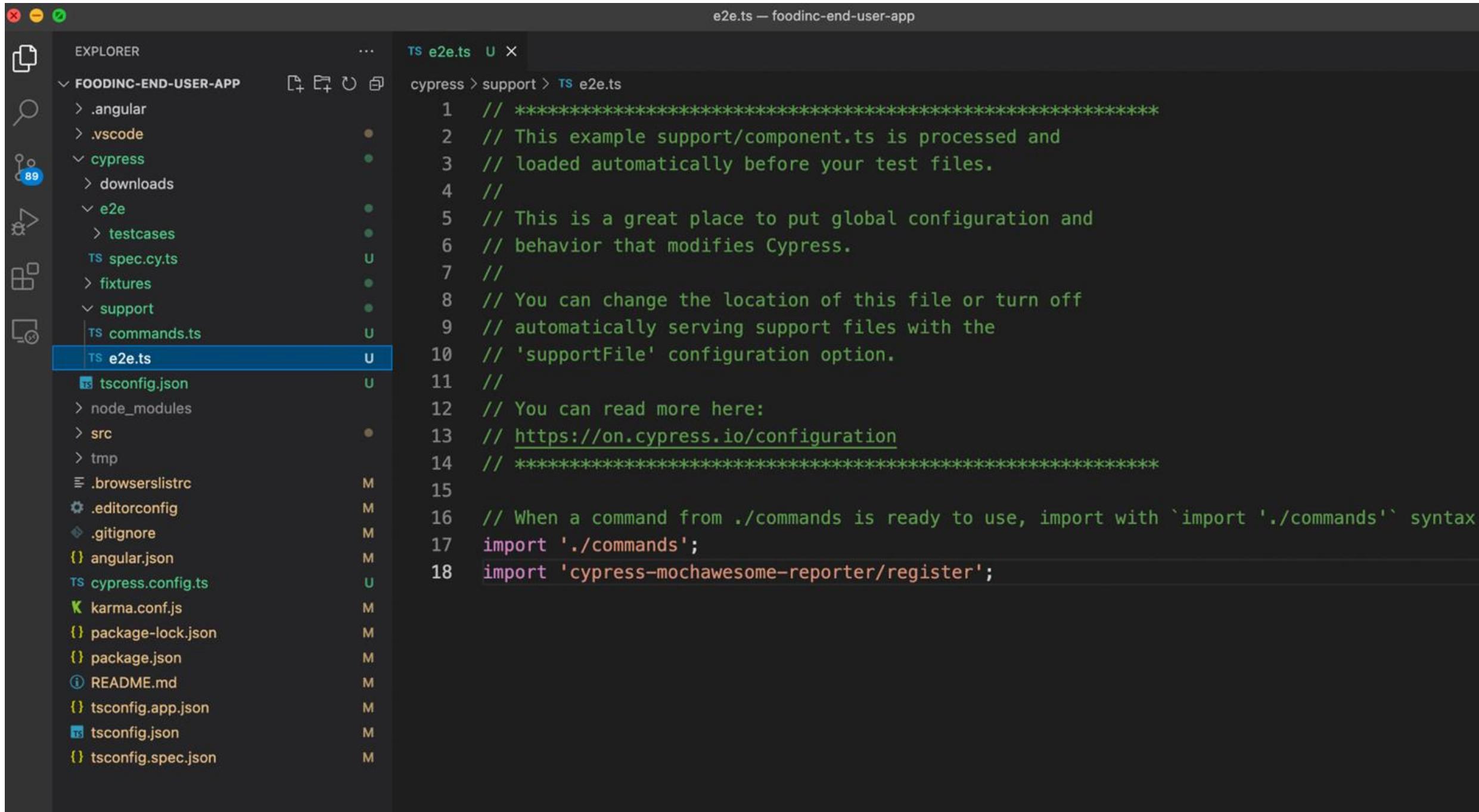
Now, configure reporter tool in your cypress.config.ts file. Add reporter options in the configuration as per your need.



```
TS cypress.config.ts — foodinc-end-user-app
TS cypress.config.ts U X
TS cypress.config.ts > [o] default
1 import { defineConfig } from 'cypress'
2
3 export default defineConfig({
4   reporter: 'cypress-mochawesome-reporter',
5   video: false,
6   reporterOptions: {
7     charts: true,
8     reportDir: "cypress/report",
9     reportPageTitle: 'admin-report',
10    embeddedScreenshots: true,
11    inlineAssets: true,
12    saveAllAttempts: false,
13    overwrite: true
14  },
15  e2e: {
16    setupNodeEvents(on, config) {
17      require('cypress-mochawesome-reporter/plugin')(on);
18    },
19    baseUrl: 'http://localhost:4200',
20  },
21})
```

# Cypress Mochawesome Reporter

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



The screenshot shows a dark-themed instance of Visual Studio Code. In the left sidebar (EXPLORER), there is a tree view of a project named "FOODINC-END-USER-APP". The "cypress" folder contains "support", which in turn contains "commands.ts" and "e2e.ts". The "e2e.ts" file is currently selected and open in the main editor area. The code in "e2e.ts" is as follows:

```
// ****
// This example support/component.ts is processed and
// loaded automatically before your test files.
//
// This is a great place to put global configuration and
// behavior that modifies Cypress.
//
// You can change the location of this file or turn off
// automatically serving support files with the
// 'supportFile' configuration option.
//
// You can read more here:
// https://on.cypress.io/configuration
// ****
// When a command from ./commands is ready to use, import with `import './commands'` syntax
import './commands';
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

