Hands On Lab 1 - Asynchronous Testing

**Objective:** The objective of this lab exercise is to demonstrate the implementation of asynchronous testing in an Angular application. By the end of this exercise, participants should be able to:

- Understand the concept of asynchronous testing in Angular.

- Create and test asynchronous methods in Angular services.

- Write unit tests for Angular components that involve asynchronous operations.

- Use the Angular testing utilities to handle asynchronous code and assertions.

- Validate that asynchronous operations are functioning correctly in the application.

**Step 1: Set up a new Angular project**

1. Create a new Angular project using the Angular CLI:



1. Change into the project directory:



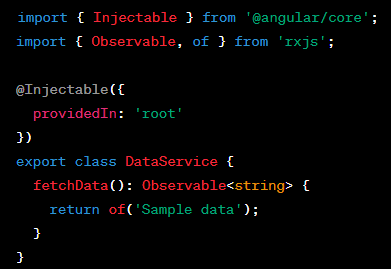
1. Open the project in your favorite code editor.

**Step 2: Create a service and test it asynchronously**

1. Generate a new service using the Angular CLI:



1. Open the data.service.ts file in the src/app directory.
2. Modify the DataService class to include a method that fetches data asynchronously. For example:



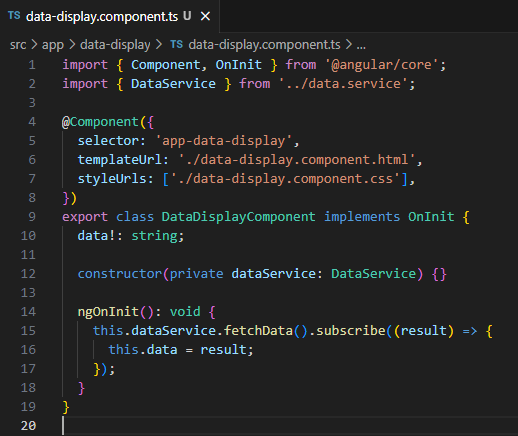
1. Save the changes to the data.service.ts file.

**Step 3: Create a component and test it asynchronously**

1. Generate a new component using the Angular CLI:



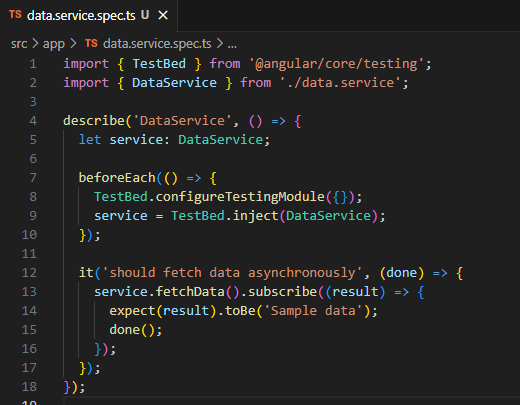
1. Open the data-display.component.ts file in the src/app/data-display directory.
2. Modify the DataDisplayComponent class to use the DataService and display the fetched data. For example:



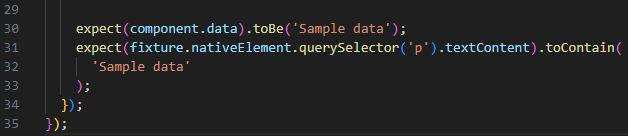
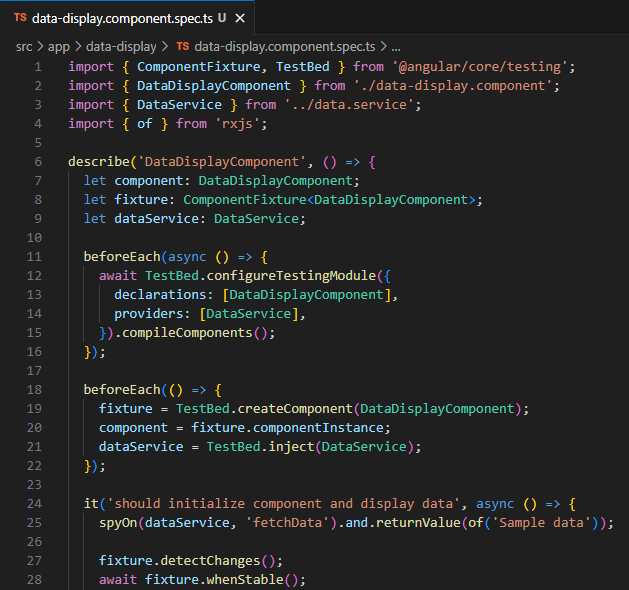
1. Save the changes to the data-display.component.ts file.

**Step 4: Write asynchronous tests**

1. Open the data.service.spec.ts file in the src/app directory.
2. Modify the existing test suite to include a test for the fetchData method. For example:



1. Open the data-display.component.spec.ts file in the src/app/data-display directory.
2. Modify the existing test suite to include a test for the component's initialization and data display. For example:



1. Save the changes to the test files.

**Step 5: Update data-display.component.html**

1. In order to test the component's template, we need to make a change to the data-display.component.html file. Let's update the test case and the component's template to ensure the expected content is displayed correctly.
2. In the **data-display.component.html** file (**src/app/data-display/data-display.component.html**), make sure it contains a **<p>** element to display the fetched data:

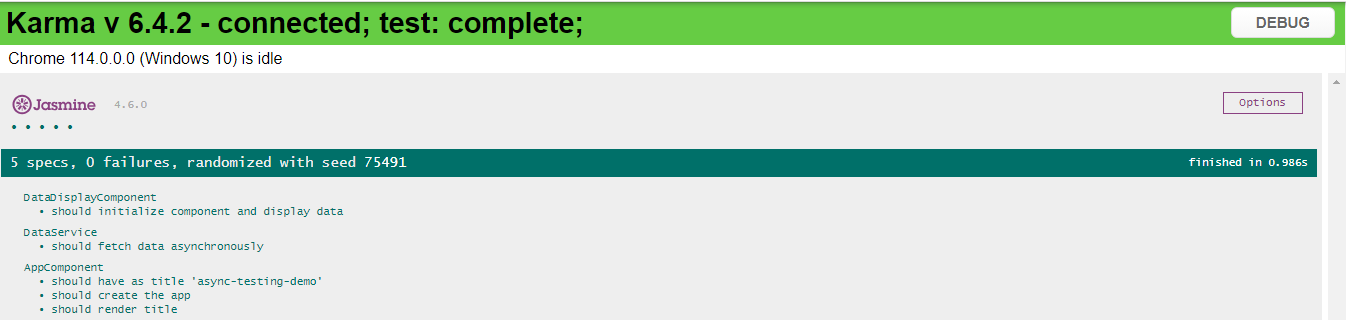


**Step 6: Run the tests**

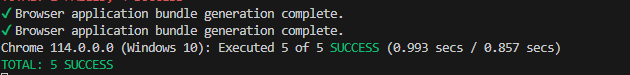
1. Open a terminal or command prompt.
2. Navigate to the project directory if you're not already there.
3. Run the following command to execute the tests



1. Angular's test runner will launch, and the tests will run.



1. Once the tests complete, you should see the test results in the terminal or command prompt.

 With the passing tests, you can be confident that the asynchronous behavior of the component and service is working as intended. This demonstrates the usage of asynchronous testing in Angular, which is crucial for handling asynchronous operations such as API calls or async data manipulation.

That's it! You have successfully created a step-by-step lab exercise to demonstrate Asynchronous Testing in Angular.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*