Project: Automation and Testing of Angular web application using Cypress

Overview

This project is to set up and perform end-to-end (E2E) testing on an Angular web application using Cypress. Cypress is a JavaScript based testing framework that simplifies writing, running, and debugging front-end tests. This project will focus on using Cypress to automate tests for various components of the Angular application, specifically for forms and user interactions.

Prerequisites :

1. **Node.js** and **npm** need to be installed on your machine.
2. **Angular WebApp code** that is accessible from Github repo and can be served on localhost.
3. **Cypress** must be installed as a testing tool for end-to-end tests.

Steps to Set Up and Execute the Project 1. Set Up Your Environment

- **Install Node.js and npm**: Download and install Node.js from https://nodejs.org/. This will also install npm, which is required for installing Cypress and other dependencies.
- **Clone the Project Repository**: In your terminal, clone the Angular application repository to your local machine.

bash Copy code git clone https://github.com/Simplilearn-

Edu/ATE_PEP1_Testing_Using_Cypress.git cd ATE_PEP1_Testing_Using_Cypress

2. Install Project Dependencies

- Inside the project directory, install dependencies to set up the Angular application:

  npm install

3. Install Cypress

- Install Cypress as a development dependency:

  npm install cypress --save-dev

4. Configure and Run the Angular Application

- Start the Angular application locally so it’s accessible to Cypress:

  npm start

- Verify that the application is running at http://localhost:4200 by visiting this URL in a browser.

5. Configure Cypress for the Project

- Open the cypress.config.js file in the root of your project. Set the baseUrl to point to the URL where your Angular app is running. In this case, set it to http://localhost:4200/pages:

  module.exports = {

```
  e2e: {
    baseUrl: 'http://localhost:4200/pages',  // Points Cypress to the Angular app's URL
specPattern: 'cypress/e2e/**/*.cy.js'    // Specifies location of test files
  }
};
```

6. Create Cypress Test Files

- Inside the cypress/e2e folder, create a test file, e.g., Test.spec.js, to write your test cases.

7. Run Cypress Test Runner

- Open Cypress Test Runner to execute tests:

```
npx cypress open
```

- In the Cypress window, select your test file (e.g., Test.spec.js) to run the tests.

Troubleshooting Common Issues

1. **Cypress Cannot Find the Application (404 Error)**:
   - Ensure that the Angular application is running and accessible at http://localhost:4200/pages.
   - Verify that baseUrl is correctly set to http://localhost:4200/pages in cypress.config.js.
2. **Timeout Error: Element Not Found**:
   - If Cypress times out while trying to find an element, increase the timeout in cy.get() or use cy.wait() to delay until the element is visible. o Check that the element has the correct attribute, e.g., data-cy="signInButton". If not, update the selector in your test or modify the HTML.
3. **Conditional Loading of Elements**:
   - If elements load based on conditions (e.g., after clicking a button), use cy.contains() or cy.wait() to wait for the element to appear before interacting with it.

Conclusion

This project provided a structured way to use Cypress for E2E testing on an Angular application. By setting up Cypress, configuring the baseUrl, and writing tests, we've established a repeatable testing process. Cypress allows us to simulate user interactions like clicking buttons, filling out forms, and validating content on the page, ensuring our Angular application behaves as expected.

Cypress's real-time feedback and debugging tools made it easy to detect and fix issues, enhancing the reliability of our application. With Cypress, we have a robust tool to maintain test coverage and quickly identify regressions during development.

Let me know if you need further customization or additional test scenarios for this project!