**CYPRESS:**

To perform end-to-end (E2E) testing in a React application, you can use testing frameworks and libraries like Cypress, Puppeteer, or WebDriverIO. Here's a general outline of how you can set up and use Cypress for React E2E testing:

1. Project Setup:

Start by creating a React project or use an existing one. You can use tools like Create React App to quickly set up a new project. Ensure you have Node.js installed on your system.

2. Install Cypress:

You can install Cypress as a development dependency using npm

**npm install cypress –save-dev**

3. Cypress Configuration:

Cypress provides an interactive CLI for initial setup and configuration. Run the following command in your project directory:

**npx cypress open**

This will create a `cypress` directory in your project and open the Cypress Test Runner. You can use the Test Runner to manage and run your tests.

4. Write Test Scripts:

Cypress allows you to write E2E tests in JavaScript or TypeScript. By default, it creates an example test script that you can find in the `cypress/integration` directory. You can create new test files in this directory or modify the existing ones.

Here's a simple example of a Cypress test for a React application:

**javascript**

**// cypress/integration/sample.spec.js**

**describe('Sample Test', () => {**

**it('Visits the app root URL', () => {**

**cy.visit('/');**

**cy.contains('Welcome to React'); // Replace with your app-specific test**

**});**

**});**

**Register.cy:**

describe('template spec', () => {

    it('passes', () => {

      cy.visit('[http://localhost:3000/UserRegistrationForm'](http://localhost:3000/UserRegistrationForm%27))

      cy.title().should('eq','Movie App')

      cy.get('[id="loginId"]').should('exist')

      cy.get('[id="firstName"]').should('exist')

      cy.get('[id="lastName"]').should('exist')

      cy.get('[id="email"]').should('exist')

    })

    it('passes', () => {

      cy.visit('[http://localhost:3000/UserRegistrationForm'](http://localhost:3000/UserRegistrationForm%27))

      cy.title().should('eq','Movie App')

      cy.get('[id="loginId"]').type('jothi123')

      cy.get('[id="firstName"]').type('jothi')

      cy.get('[id="lastName"]').type('mani')

      cy.get('[id="email"]').type('jothi@gmail.com')

    })

  })

**Login.cy:**

describe('template spec', () => {

  it('passes', () => {

    cy.visit('[http://localhost:3000/'](http://localhost:3000/%27))

    cy.title().should('eq','Movie App')

    cy.contains('Login').click();

    cy.get('[id="loginId"]').should('exist')

    cy.get('[id="password"]').should('exist')

  })

  it('passes', () => {

    cy.visit('[http://localhost:3000/'](http://localhost:3000/%27))

    cy.title().should('eq','Movie App')

    cy.contains('Login').click();

    cy.get('[id="loginId"]').type('sudha123')

    cy.get('[id="password"]').type('abc123')

    cy.get('[type="submit"]').click();

  })

})

**AddMovie.cy:**

describe('template spec', () => {

    it('passes', () => {

      cy.visit('[http://localhost:3000/AddMovieForm'](http://localhost:3000/AddMovieForm%27))

      cy.title().should('eq','Movie App')

      cy.contains('Submit').click();

      cy.get('[id="\_id"]').should('exist')

      cy.get('[id="movieName"]').should('exist')

      cy.get('[id="theatreName"]').should('exist')

      cy.get('[id="noOfTicketsAvailable"]').should('exist')

      cy.get('[id="ticketsStatus"]').should('exist')

    })

    it('passes', () => {

      cy.visit('[http://localhost:3000/AddMovieForm'](http://localhost:3000/AddMovieForm%27))

      cy.title().should('eq','Movie App')

      cy.contains('Submit').click();

      cy.get('[id="\_id"]').type('111')

      cy.get('[id="movieName"]').type('Leo')

      cy.get('[id="theatreName"]').type('inox')

      cy.get('[id="noOfTicketsAvailable"]').type('126')

      cy.get('[id="ticketsStatus"]').type('Booked ASAP')

      cy.get('[type="submit"]').click();

    })

  })

**BookTicket.cy:**

describe('template spec', () => {

    it('passes', () => {

      cy.visit('<http://localhost:3000/DeleteMovie'>)

      cy.title().should('eq','Movie App')

      cy.contains('Submit').click();

      cy.get('[id="\_id"]').should('exist')

      cy.get('[id="movieName"]').should('exist')

      cy.get('[id="theatreName"]').should('exist')

      cy.get('[id="noOfTicketsAvailable"]').should('exist')

      cy.get('[id="ticketsStatus"]').should('exist')

    })

    it('passes', () => {

      cy.visit('<http://localhost:3000/BookTicket'>)

      cy.title().should('eq','Movie App')

      cy.contains('Submit').click();

      cy.get('[id="loginId"]').type('111')

      cy.get('[id="movieName"]').type('Leo')

      cy.get('[id="theatreName"]').type('inox')

      cy.get('[id="noOfTickets"]').type('1')

      cy.get('[id="seat\_no"]').type('c1')

      cy.get('[type="submit"]').click();

    })

  })

**DeleteMovie.cy:**

describe('template spec', () => {

    it('passes', () => {

      cy.visit('<http://localhost:3000/DeleteMovie'>)

      cy.title().should('eq','Movie App')

      cy.contains('Submit').click();

      cy.get('[id="\_id"]').should('exist')

      cy.get('[id="movieName"]').should('exist')

      cy.get('[id="theatreName"]').should('exist')

      cy.get('[id="noOfTicketsAvailable"]').should('exist')

      cy.get('[id="ticketsStatus"]').should('exist')

    })

    it('passes', () => {

      cy.visit('<http://localhost:3000/DleteMovie'>)

      cy.title().should('eq','Movie App')

      cy.contains('Submit').click();

      cy.get('[id="\_id"]').type('111')

      cy.get('[id="movieName"]').type('Leo')

      cy.get('[id="theatreName"]').type('inox')

      cy.get('[id="noOfTicketsAvailable"]').type('126')

      cy.get('[id="ticketsStatus"]').type('Booked ASAP')

      cy.get('[type="submit"]').click();

    })

  })

This test script visits your application's root URL and checks if it contains a specific element.

5. Running Tests:

You can run your Cypress tests using the Test Runner by running the following command in your project directory:

**npx cypress open**

This will open the Cypress Test Runner, where you can select and run your test files. You can also run tests in headless mode using the CLI:

**npx cypress run**

6. Assertions and Interactions:

Cypress provides a rich set of commands for interacting with your application, making assertions, and simulating user behaviour. You can use these commands to test various aspects of your React application, such as form submissions, navigation, and UI elements.

7. Continuous Integration:

To incorporate E2E tests into your CI/CD pipeline, you can use the `cypress run` command with CI options. This allows you to run tests in a headless environment and generate reports for integration with tools like Jenkins, Travis CI, or GitHub Actions.

8. Custom Configuration:

You can customize Cypress configuration in the `cypress.json` file or by using plugins to extend its functionality. For example, you can set up custom commands or integrate with other testing libraries and tools.

9. Maintaining and Expanding Tests:

As your React application evolves, you should keep your E2E tests up to date. You may need to add new tests for new features or update existing tests to reflect changes in your application.

10. Review and Reporting:

Review the test results and generate reports to identify issues and regressions. Cypress provides built-in support for generating reports in various formats.

Remember that E2E testing is just one part of a comprehensive testing strategy. It's essential to combine it with unit testing and integration testing to ensure the overall reliability of your React application.