Testing the react application:

* Manual testing
  + Testing with search functionality without test cases
  + It is not efficient ..if I change one module am I going to test entire application again?? ..>this wont work
  + When application grow big react application has many components and changing code in one component may effect other components
  + Even a single line of code will make impact of code
* **Write test cases to test application**
  + It will test application automatically

There are 3 types of testing in react

Types of testing

1.unit testing

2.integration testing

3.end to end testing

**1.unit testingg** : Test react component in isolation ..eg:if we need to test headder component .. I will unit test header component alone to check whehter it is rendered properly or not -->one component in isolation

**2.integration testing** : testing integration of component . component talk to each other .. there are so many component colabrated to form a module .. we will develop a flow to test it..eg:if application have card and if we search the input there are many component involved so we will test

**3.End to end testing :** We will test from start to end .. login to logout .. similate the user flow

-->it requires different type of tools like sypus etc

we will focus of unit testing and integration testing

libraries used for testing

1.react testing library -->it is given by react to test application .. it has many react features

along with create react app --> testing library comes with package --> it uses jest behind the screen

2.jest --> it is javascript testing library --> it uses babel behind scene

#setting up testing

1.@testing-library/react

2.install jest

3.babel with dependency🡪https://jestjs.io/docs/getting-started

4.config babel 🡪 babel.config.js 🡪copy past form jest

5.disable babel with parcel to make it work with jest through .parcel config file--https://parceljs.org/languages/javascript/

6.setup command to run test caase

7.writing test configuration .. need to create new configuration file for jest configuration🡪 jest –init

8.jsdom library 🡪 this act as browser to test🡪separate library 🡪npm install --save-dev jest-environment-jsdom

9.we need to add

"@babel/preset-react - is trying to convert jsx to html for testing

10.

We need babel for react testing library but we already we babel for parcel which will create conflict with it 🡪 we are trying to add extra configuration to babel so we need to change parcel to make it compactable with jest

Solution:

<https://parceljs.org/languages/javascript/>

1.babel section -> usage with other tools

1.we can disable babel with parcel which wont impact the application and help babel make work with jest

what is jsdom?

1.When we run test cases it wont run in browser .. these test cases will need environment to make testing exect

2.jsdom is like browser to make html code work with jsdom ..

3.  the goal of the project is to emulate enough of a subset of a web browser to be useful for testing and scraping real-world web applications.

Unit testing:

Taking up one component and testing it

Sample unit testing

import { Provider } from "react-redux";

import Header from "../Header";

import appStore from "../../utils/appStore";

import { BrowserRouter } from "react-router-dom";

import { fireEvent, render, screen } from "@testing-library/react";

import "@testing-library/jest-dom";

it("Should have login button", () => {

  render(

    <BrowserRouter>

      <Provider store={appStore}>

        <Header />

      </Provider>

    </BrowserRouter>

  );

  //   if there are many buttons in form and need only button with name:login

  const loginButton = screen.getByRole("button", { name: "Login" });

  expect(loginButton).toBeInTheDocument();

  //   we can use regular expression to test whether the card is present

  const cartItem = screen.getByText(/Cart/);

  expect(cartItem).toBeInTheDocument();

  //   when external library are useed there will be issue with it

});

it("Should have logout when button clicked", () => {

  render(

    <BrowserRouter>

      <Provider store={appStore}>

        <Header />

      </Provider>

    </BrowserRouter>

  );

  //   if there are many buttons in form and need only button with name:login

  const loginButton = screen.getByRole("button", { name: "Login" });

  expect(loginButton).toBeInTheDocument();

  fireEvent.click(loginButton);

  const logoutButton = screen.getByRole("button", { name: "Logout" });

  expect(logoutButton).toBeInTheDocument();

  //   when external library are useed there will be issue with it

});

We are using browserRouter since Link is used in header

When we run test cases it wont run directly in browser .. it runs in jest-dom and the api from browser like fetch are from native browser so it wont work in testing so how to make it work?

We need to make moke function for fetch to make fetch work

We cannot make a api call directly always so we just create moke fetch or any function to use

Mock fetch to use with test case

  global.fetch = jest.fn(() => {

    return Promise.resolve({

      json: () => Promise.resolve(MOCK\_DATA),

    });

  });

As we use state with component we need to wrap it with act() to make to work with component

At comes from react-testing-library

it("Should search for particular item", async () => {

  await act(async () =>

    render(

      <BrowserRouter>

        <Body />

      </BrowserRouter>

    )

  );

Automatic update of test case update(hmr) through below code

 "watch-test": "jest --watch"

**Integration testing**

* Test the whole module which involves different component or functionality together
* **Fetch,e.target wont work with tets since ther test cases are not run in actual browser**
* **So we use mock for fetch 🡪mock\_data is the json o/p of fetch**

global.fetch = jest.fn(() => {

  return Promise.resolve({

    json: () => Promise.resolve(MOCK\_DATA),

  });

});

Integration testing example

global.fetch = jest.fn(() => {

  return Promise.resolve({

    json: () => Promise.resolve(MOCK\_DATA),

  });

});

it("Should search for particular item", async () => {

  await act(async () =>

    render(

      <BrowserRouter>

        <Body />

      </BrowserRouter>

    )

  );

  const searchBtn = screen.getByRole("button", { name: "search" });

  //   to get input that dont hav ename or placeholder or any unique value we can use data-testid

  const inputElement = screen.getByTestId("search");

  fireEvent.change(inputElement, { target: { value: "burger" } }); // in second parameter we are setting trigger value manually since e is from browser api

  fireEvent.click(searchBtn);

  const cards = screen.getAllByTestId("resCard");

  expect(cards.length).toBe(1);

});

it("Should get top rated restro", async () => {

  await act(async () =>

    render(

      <BrowserRouter>

        <Body />

      </BrowserRouter>

    )

  );

  const searchBtn = screen.getByRole("button", { name: /top rated restorent/ });

  //   to get input that dont hav ename or placeholder or any unique value we can use data-testid

  const cardsBeforeFilter = screen.getAllByTestId("resCard");

  expect(cardsBeforeFilter.length).toBe(8);

  fireEvent.click(searchBtn);

  const cards = screen.getAllByTestId("resCard");

  expect(cards.length).toBe(3);

});

There are few functions given by test to run before all test case is run and after each test case is run

beforeAll(() => {

  console.log("before all test case run");

});

beforeEach(() => {

  console.log("before each test case run");

});

afterAll(() => {

  console.log("after all test case run");

});

afterEach(() => {

  console.log("after each test case run");

});

Note:

import "@testing-library/jest-dom"; //this holds toBeInTheDocument()

| **fireEvent** | **userEvent** |
| --- | --- |
| Fires an event directly | Simulates realistic user actions |
| Immediate (sync) | More real, often async |
| Lower-level | Higher-level abstraction |
| Good for small, simple triggers | Better for real user flows (typing, clicking, tabbing) |

Sample test case

import { act, fireEvent, render, screen } from "@testing-library/react";

import RestaurantMenu from "../RestaurantMenu";

import MOCK\_DATA from "../mock/cartMock.json";

import "@testing-library/jest-dom"; //this holds toBeInTheDocument()

import { Provider } from "react-redux";

import appStore from "../../utils/appStore";

import Header from "../Header";

import { BrowserRouter } from "react-router-dom";

import Cart from "../Cart";

global.fetch = jest.fn(() => {

  return Promise.resolve({

    json: () => Promise.resolve(MOCK\_DATA),

  });

});

it("Should load restaurant menu component", async () => {

  await act(() =>

    render(

      <BrowserRouter>

        <Provider store={appStore}>

          <Header />

          <RestaurantMenu />

          <Cart />

        </Provider>

      </BrowserRouter>

    )

  );

  const restroName = screen.getByText("A2B - Adyar Ananda Bhavan");

  expect(restroName).toBeInTheDocument();

  const accordHeader = screen.getByText("Recommended (20)");

  fireEvent.click(accordHeader);

  // asset whether there are 20 items in recomended

  const items = screen.getAllByTestId("items");

  expect(items.length).toBe(20);

  //   this throw error but need getallbyrole

  const addBtn = screen.getAllByRole("button", { name: "ADD" });

  expect(screen.getByText("Cart - (0 items)")).toBeInTheDocument();

  fireEvent.click(addBtn[0]);

  //   after clicking button we need to check whether header changed -- we can add header component too

  expect(screen.getByText("Cart - (1 items)")).toBeInTheDocument();

  fireEvent.click(addBtn[1]);

  expect(screen.getByText("Cart - (2 items)")).toBeInTheDocument();

  const clearCart = screen.getByRole("button", { name: "Clear Cart" });

  fireEvent.click(clearCart);

  expect(screen.getByText("Cart - (0 items)")).toBeInTheDocument();

});