**Basics**

Jest Vs RTL:

**Jest** : it is a javascript testing framework . it is a test runner that finds tests, runs the tests, determines whether the test passed or failed and reports it back in human readable manner.

**RTL**: It is a Javascript testing utility that provides virtual DOM for testing react component.

React Testing Library provides a virtual DOM which can be used to interact with and verify the behaviour of a react component.

Testing library is infect a family of packages which helps test UI components.

The core library is called DOM Testing library and RTL is simply a wrapper around this core library to test React applications in an easier way.

**Test Components**

it or test: describes the test itself. It takes as parameters the name of the test and a function that holds the tests.

expect: the condition that the test needs to pass. It will compare the received parameter to a matcher.

a matcher: a function that is applied to the expected condition.

Create testingapp project

Add below dependencies

**npm install --save-dev @testing-library/react @testing-library/jest-dom @testing-library/user-event web-vitals react-router-dom**

Go to App component add <h1> tag to test the text inside h1

import logo from "./logo.svg";

import "./App.css";

function App() {

return (

<div className="App">

<h1>First Text</h1>

</div>

);

}

export default App;

Go to app.test .js and add testing code for h1 tag text

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

import App from "./App";

test("renders learn react link", () => {

render(<App />);

const linkElement = screen.getByText("first text");

//const linkElement = screen.getByText(/first text/i);

expect(linkElement).toBeInTheDocument();

});

Go to cmd and write npm test—it will fail because text does not match the case so add following code

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

import App from "./App";

test("renders learn react link", () => {

render(<App />);

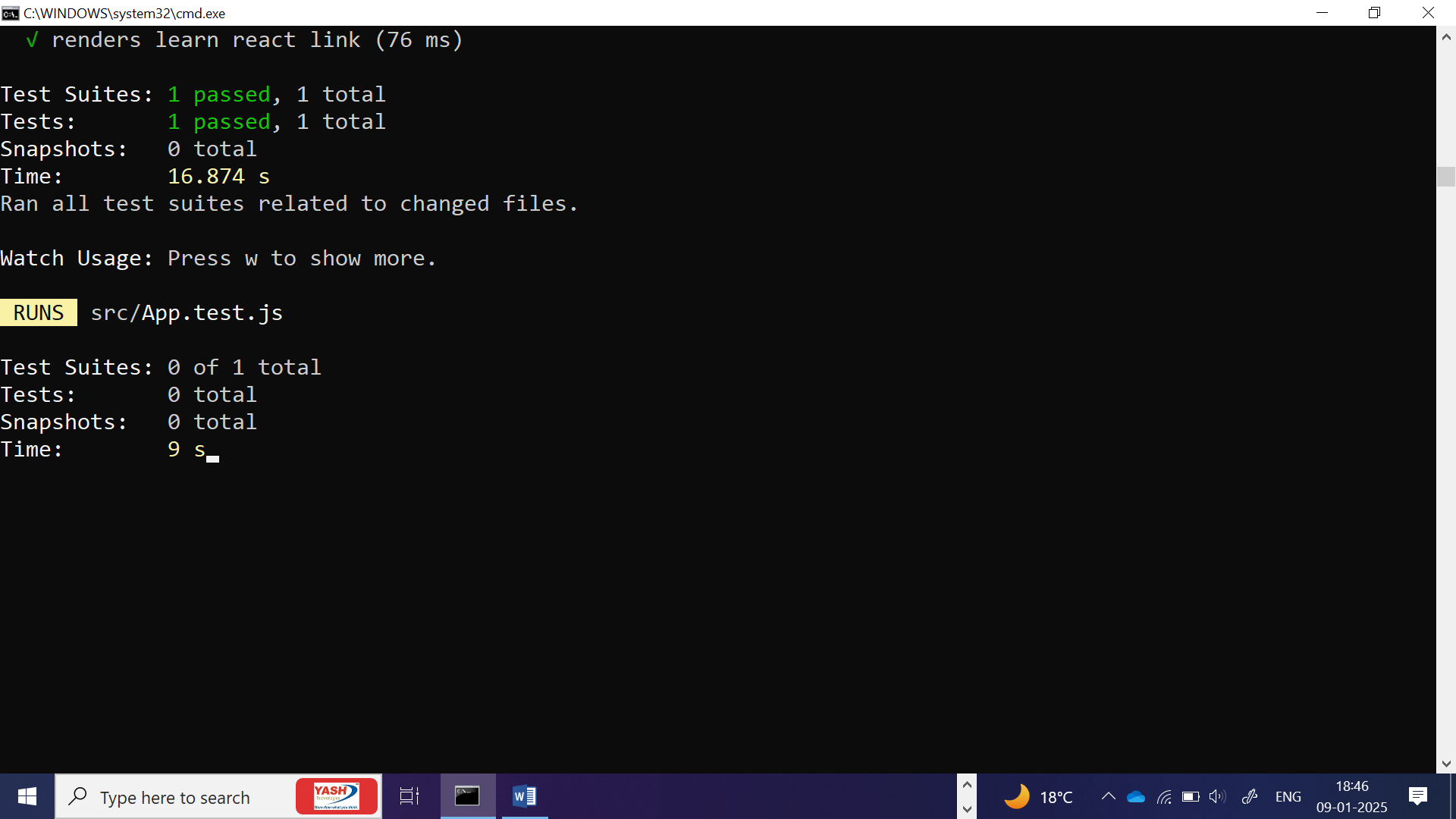
//const linkElement = screen.getByText("first text");

const linkElement = screen.getByText(/first text/i);

expect(linkElement).toBeInTheDocument();

});

The test case will pass



In the App.test add testing for another text and image

import logo from "./logo.svg";

import "./App.css";

function App() {

return (

<div className="App">

<h1>First Text</h1>

<p>Welcome to Testing</p>

<img

title="First image"

src="https://www.shutterstock.com/search/background"

/>

</div>

);

}

export default App;

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

import App from "./App";

test("renders learn react link", () => {

render(<App />);

//const linkElement = screen.getByText("first text");

const linkElement = screen.getByText(/first text/i);

const linkElement2 = screen.getByText(/welcome to testing/i);

const image = screen.getByTitle(/first image/i);

expect(linkElement).toBeInTheDocument();

expect(linkElement2).toBeInTheDocument();

expect(image).toBeInTheDocument();

});

Add counter increment and decrement to the app.js

import logo from "./logo.svg";

import "./App.css";

import React, { useState } from "react";

const App = () => {

const [count, setCount] = useState(0);

const increment = () => setCount(count + 1);

const decrement = () => setCount(count - 1);

return (

<div className="App">

<h1>First Text</h1>

<p>Welcome to Testing</p>

<img

title="First image"

src="https://www.shutterstock.com/search/background"

/>

<p>

Count: <span data-testid="count-value">{count}</span>

</p>

<button onClick={increment} data-testid="increment-btn">

Increment

</button>

<button onClick={decrement} data-testid="decrement-btn">

Decrement

</button>

</div>

);

};

export default App;

Add below test for counter

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

import App from "./App";

test("renders learn react link", () => {

render(<App />);

//const linkElement = screen.getByText("first text");

const linkElement = screen.getByText(/first text/i);

const linkElement2 = screen.getByText(/welcome to testing/i);

const image = screen.getByTitle(/first image/i);

// Check if the initial count is rendered as 0

const countValue = screen.getByTestId("count-value");

expect(countValue).toHaveTextContent("0");

// Get the count and increment button

const incrementButton = screen.getByTestId("increment-btn");

// Click the increment button

fireEvent.click(incrementButton);

expect(linkElement).toBeInTheDocument();

expect(linkElement2).toBeInTheDocument();

expect(image).toBeInTheDocument();

});

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

import App from "./App";

import userEvent from "@testing-library/user-event";

test("renders learn react link", () => {

render(<App />);

//const linkElement = screen.getByText("first text");

const linkElement = screen.getByText(/first text/i);

const linkElement2 = screen.getByText(/welcome to testing/i);

const image = screen.getByTitle(/first image/i);

// Check if the initial count is rendered as 0

const countValue = screen.getByTestId("count-value");

expect(countValue).toHaveTextContent("0");

// Get the count and increment button

const incrementButton = screen.getByTestId("increment-btn"); //event testing

// Click the increment button

fireEvent.click(incrementButton);

const submitElement = screen.getByText(/submit/i);

// Click the submit button without typing anything

userEvent.click(submitElement);

expect(linkElement).toBeInTheDocument();

expect(linkElement2).toBeInTheDocument();

expect(image).toBeInTheDocument();

});