

### Test Automation using:

- Java
- React
- Node.JS
- Auth0
- Visual Studio Code
- Vitest (framework)
- Libraries (ie., @testing-library, jest-dom)
- Git and GitHub repository

### Document highlight:

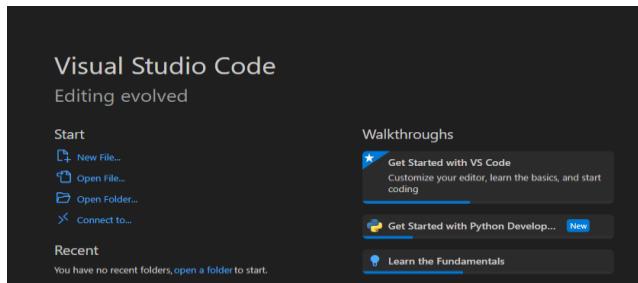
- Tools, packages, and libraries installation and setup
- Create and run test cases
  - Greet
  - User Account
  - User List
  - Product Image Gallery
  - Terms And Conditions
  - Expandable Text
- Simplifying the Tests
- Final end result

## 1. Base installation and setup

Ensure that the Visual Studio Code is setup...

Download:

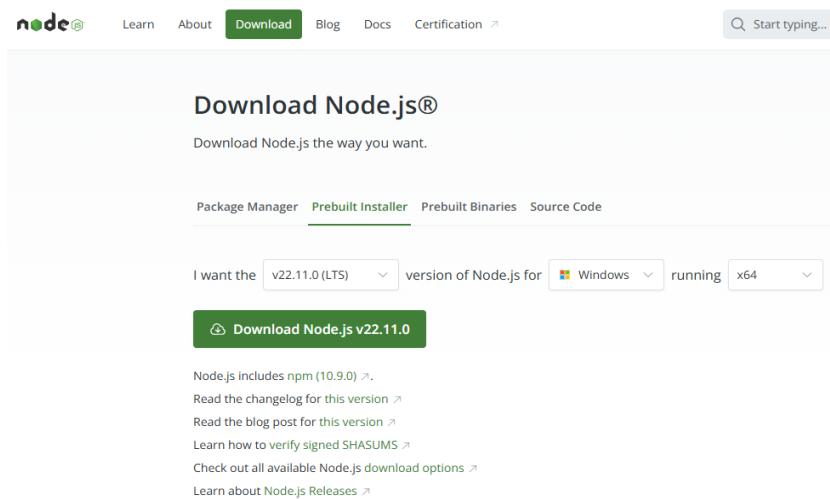
<https://code.visualstudio.com/download>



Ensure that the Node.JS is setup

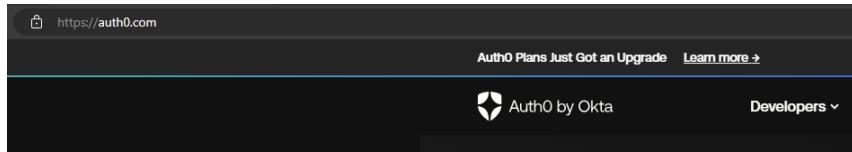
Download:

<https://nodejs.org/en/download/prebuilt-installer>



Ensure to have an Auth0 account:

<https://auth0.com/>



Update Auth0 config details under settings

Allowed Callback URLs

```
http://localhost:5173
```

After the user authenticates we will only call |  
specify multiple valid URLs by comma-separated  
different environments like QA or testing). Ma  
( https:// ) otherwise the callback may fail i  
of custom URI schemes for native clients, all  
https://. You can use [Organization URL](#) ↗ |

Allowed Logout URLs

```
http://localhost:5173
```

Comma-separated list of allowed logout URLs:  
You can use wildcards at the subdomain level  
and hash information are not taken into account.  
[Learn more about logout](#) ↗

Allowed Web Origins

```
http://localhost:5173
```

Install node components:

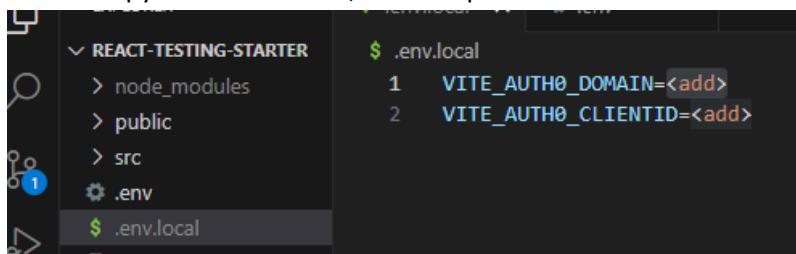
```
PS C:\Users\njml0\react-testing-starter> npm i
added 538 packages, and audited 539 packages in 28s
```

```
PS C:\Users\njml0\react-testing-starter> npm -v
10.9.0
```

Open in VS Code:

```
PS C:\Users\njml0\react-testing-starter> code .
```

Make a copy of the “env” file, in the copied file fill in the details off of the Auth0 app:



Start npm:

```
PS C:\Users\njml\react-testing-starter> npm start
```

```
> react-testing-vite@0.0.0 start
> concurrently "npm run server" "npm run dev"

(node:15388) [DEP0060] DeprecationWarning: The 'util._extend' API is deprecated. Please use Object.assign() instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
[1]
[1] > react-testing-vite@0.0.0 dev
[1] > vite
[1]
[8]
[8] > react-testing-vite@0.0.0 server
[8] > json-server --watch src/data/db.json --delay 500
[8]
[8]
[8] \{^_}/ hi!
[8]
[8] Loading src/data/db.json
[8] Done
[1]
[1] VITE v5.4.11 ready in 515 ms
[1]
[1] → Local: http://localhost:5173/
[1] → Network: use --host to expose
[8]
[8] Resources
[8] http://localhost:3000/categories
[8] http://localhost:3000/products
[8]
[8] Home
[8] http://localhost:3000
[8]
[8] Type s + enter at any time to create a snapshot of the database
[8] Watching...
[8]
```

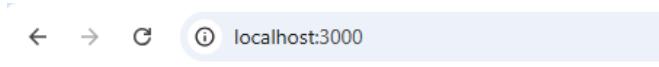
Front-end url:

<http://localhost:5173>



Back-end url:

<http://localhost:3000>



**JSON Server is running!**

## 2. Install the testing framework Vitest:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\njml0\react-testing-starter>npm i -D vitest

added 30 packages, removed 1 package, changed 3 packages, and audited 570 packages in 8s

found 0 vulnerabilities

C:\Users\njml0\react-testing-starter>
```

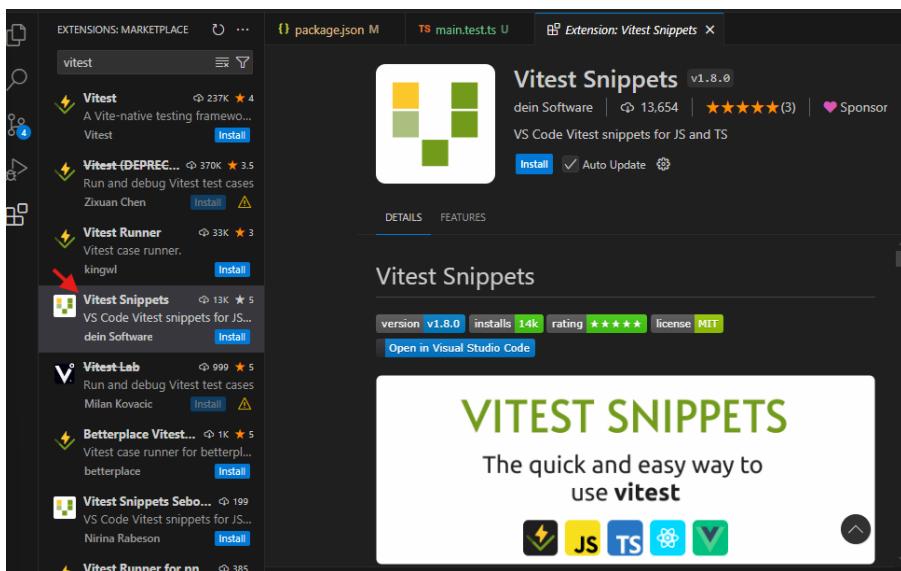
```
C:\Users\njml0\react-testing-starter>npm vitest -v
10.9.0

C:\Users\njml0\react-testing-starter>[]
```

Update the package.json file and include (under the section of scripts objects) the new dependency:

```
package.json M | TS main.test.ts U
package.json > {} dependencies > @hookform/resolvers
1  {
2    "name": "react-testing-vite",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    ▷ Debug
7    "scripts": {
8      "dev": "vite",
9      "build": "tsc && vite build",
10     "lint": "eslint . --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
11     "preview": "vite preview",
12     "server": "json-server --watch src/data/db.json --delay 500",
13     "start": "concurrently \"npm run server\" \"npm run dev\"",
14     "test": "vitest",
15     "test:ui": "vitest --ui"
16   }
17 }
```

Install the needed extension to be able to use the essential functions of Vitest:



To test that the setup was set properly and runs as expected:

Create a folder and file



Code a test suite and run it:

```
{} package.json M ts main.test.ts U X
tests > ts main.test.ts > ...
1 import { it, expect, describe } from 'vitest'
2
3 describe('group', () => {
4   it('should', () => {
5     expect(1).toBeTruthy();
6   })
7 })
```

In the terminal window, run test npm t:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
C:\Users\njml0\react-testing-starter>npm t

> react-testing-vite@0.0.0 test
> vitest

[DEV] v2.1.5 C:/Users/njml0/react-testing-starter
✓ tests/main.test.ts (1)
  ✓ group (1)
    ✓ should

Test Files 1 passed (1)
Tests 1 passed (1)
Start at 10:07:00
Duration 1.07s (transform 66ms, setup 0ms, collect 39ms, tests 3ms, environment 0ms, prepare 652ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```

In the terminal window, install @vitest/ui:

```
C:\Users\njml0\react-testing-starter>npm install @vitest/ui
added 9 packages, changed 1 package, and audited 579 packages in 4s
found 0 vulnerabilities

C:\Users\njml0\react-testing-starter>
C:\Users\njml0\react-testing-starter>npm @vitest/ui -v
10.9.0
C:\Users\njml0\react-testing-starter>
```

In the terminal window, run test npm test ui:

```
C:\Users\njml0\react-testing-starter>npm run test:ui
> react-testing-vite@0.0.0 test:ui
> vitest --ui

[DEV] v2.1.5 C:/Users/njml0/react-testing-starter
UI started at http://localhost:51204/_vitest_/

✓ tests/main.test.ts (1)
  ✓ group (1)
    ✓ should

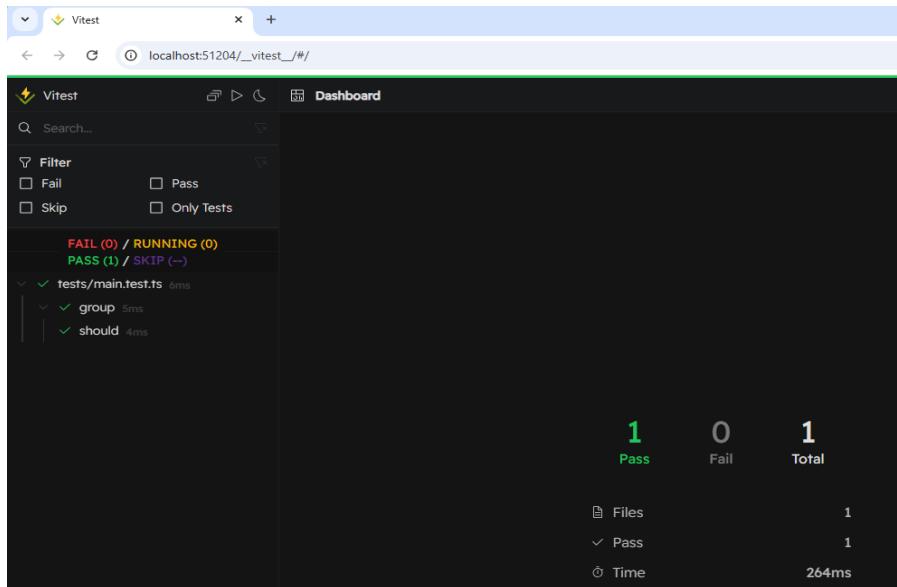
Test Files 1 passed (1)
  Tests 1 passed (1)
Start at 10:23:10
Duration 1.35s (transform 146ms, setup 0ms, collect 73ms, tests 6ms, environment 0ms, prepare 262ms)

PASS Waiting for file changes...
press h to show help, press q to quit

C:\Users\njml0\react-testing-starter>npm test:ui -v
10.9.0

C:\Users\njml0\react-testing-starter>
```

Test UI:



### 3. Setup React testing libraries

Install libraries @testing-library/react and @testing-library/dom :

```
C:\Users\njml0\react-testing-starter>npm i -D @testing-library/react @testing-library/dom
added 16 packages, and audited 595 packages in 5s
found 0 vulnerabilities

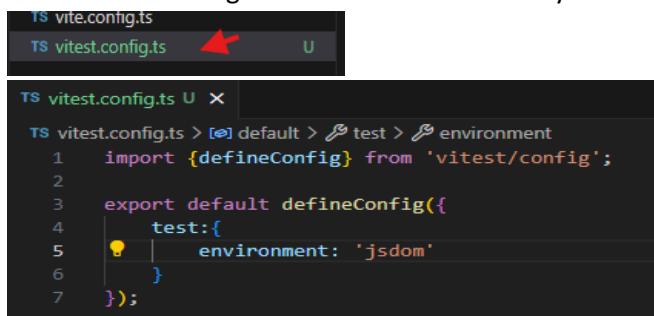
C:\Users\njml0\react-testing-starter>
```

```
C:\Users\njml0\react-testing-starter>npm @testing-library/react -v  
10.9.0  
C:\Users\njml0\react-testing-starter>
```

Install library jsdom:

```
C:\Users\njml0\react-testing-starter>npm i -D jsdom  
added 28 packages, and audited 623 packages in 5s  
found 0 vulnerabilities  
C:\Users\njml0\react-testing-starter>npm jsdom -v  
10.9.0  
C:\Users\njml0\react-testing-starter>
```

Create a new config file from the root directory:



To test that all installed are working at this point:

```
C:\Users\njml0\react-testing-starter>npm run test:ui  
> react-testing-vite@0.0.0 test:ui  
> vitest --ui  
  
[DEV] v2.1.5 C:/Users/njml0/react-testing-starter  
UI started at http://localhost:51204/_vitest_/  
  
✓ tests/main.test.ts (1)  
  ✓ group (1)  
    ✓ should  
  
Test Files 1 passed (1)  
  Tests 1 passed (1)  
Start at 11:25:43  
Duration 2.69s (transform 74ms, setup 0ms, collect 50ms, tests 4ms, environment 1.18s, prepare 207ms)  
  
PASS Waiting for file changes...  
press h to show help, press q to quit
```

Install library jest-dom:

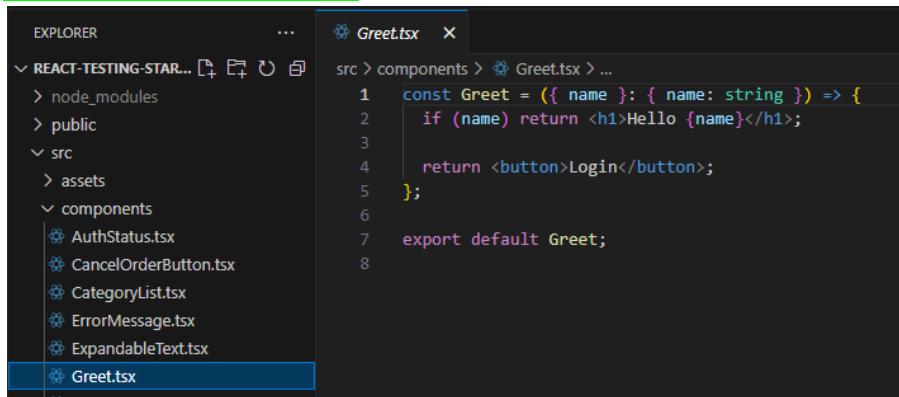
```
C:\Users\njml0\react-testing-starter>npm i -D @testing-library/jest-dom
added 14 packages, and audited 637 packages in 3s
found 0 vulnerabilities

C:\Users\njml0\react-testing-starter>npm @testing-library/jest-dom -v
10.9.0

C:\Users\njml0\react-testing-starter>
```

#### 4. Create and run test cases

##### Component to test: Greet.tsx



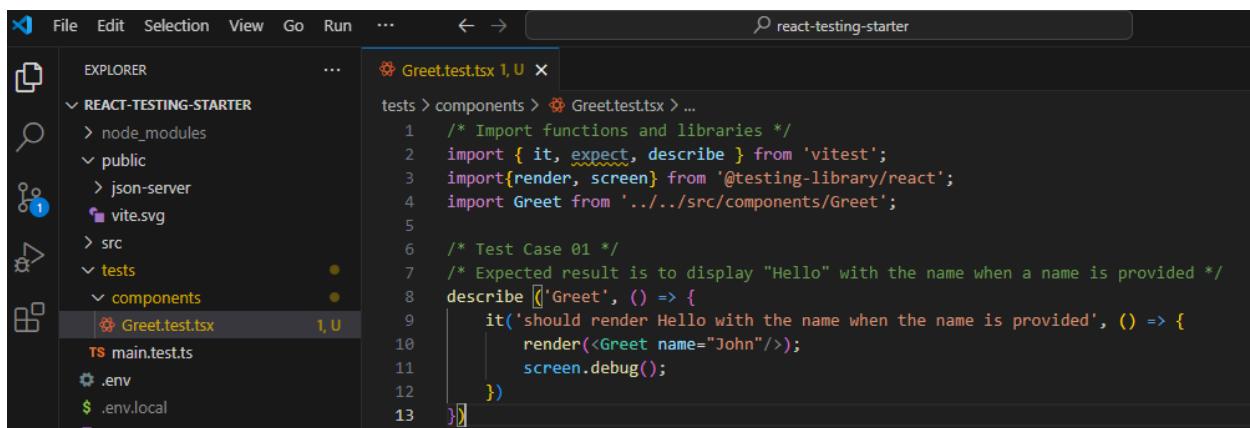
The screenshot shows the VS Code interface. On the left is the Explorer sidebar with project files like node\_modules, public, and src. Under src, there's a components folder containing several files: AuthStatus.tsx, CancelOrderButton.tsx, CategoryList.tsx, ErrorMessage.tsx, ExpandableText.tsx, and Greet.tsx. Greet.tsx is currently selected and highlighted with a blue bar at the bottom. The main editor area shows the code for Greet.tsx:

```
const Greet = ({ name }: { name: string }) => {
  if (name) return <h1>Hello {name}</h1>;
  return <button>Login</button>;
};

export default Greet;
```

##### Test case 01:

Create a test file and test case...



The screenshot shows the VS Code interface with a different view. The Explorer sidebar now shows a tests folder under src, which contains a components folder. Inside components, Greet.test.tsx is selected and highlighted. The main editor area shows the test code for Greet.tsx:

```
/* Import functions and libraries */
import { it, expect, describe } from 'vitest';
import { render, screen } from '@testing-library/react';
import Greet from '../src/components/Greet';

/* Test Case 01 */
/* Expected result is to display "Hello" with the name when a name is provided */
describe('Greet', () => {
  it('should render Hello with the name when the name is provided', () => {
    render(<Greet name="John"/>);
    screen.debug();
  });
});
```

```
RERUN x1

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
<body>
  <div>
    <h1>
      Hello
      John
    </h1>
  </div>
</body>

✓ tests/components/Greet.test.tsx (1)
  ✓ Greet (1)
    ✓ should render Hello with the name when the name is provided

Test Files 1 passed (1)
  Tests 1 passed (1)
  Start at 12:15:53
  Duration 320ms

PASS Waiting for file changes...
press h to show help, press q to quit
```

Vitest

tests/components/Greet.test.tsx

All tests passed in this file

FAIL (0) / RUNNING (0)  
PASS (2) / SKIP (-)

tests/components/Greet.test.tsx 29ms

Greet 28ms

should render Hello with the name when the name is provided 27ms

Vitest

tests/components/Greet.test.tsx

12:15:54 PM | Greet > should render Hello with the name when the name is provided | stdout

```
<body>
  <div>
    <h1>
      Hello
      John
    </h1>
  </div>
</body>
```

FAIL (0) / RUNNING (0)  
PASS (2) / SKIP (-)

tests/components/Greet.test.tsx 29ms

Greet 28ms

should render Hello with the name when the name is provided 27ms

tests/main.test.ts 6ms

To find the <body> <div> <h1> in the dom and make accession, use testing library query methods:

The screenshot shows the 'About Queries' page of the Testing Library documentation. The left sidebar has a tree view with 'About Queries' selected. The main content area has a heading 'About Queries' and a section titled 'Overview'. It explains that queries are methods to find elements on the page, mentioning 'get', 'find', and 'query' types, and notes that the query will throw an error if no element is found or return a Promise and retry if it will.

To use jest-dom custom matchers:

The screenshot shows the GitHub page for the '@testing-library/jest-dom' package. It includes the package name, version (6.6.3), license (MIT), and a 'Readme' tab which is currently selected. Other tabs include 'Code', 'Beta', '7 Dependencies', and '21,02:'. The main content area features a logo of an owl and the text 'Custom jest matchers to test the state of the DOM'.

The screenshot shows the npmjs.com page for the '@testing-library/jest-dom' package. The top navigation bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', and '...'. The main content area lists 'Custom matchers' with items like 'toBeDisabled', 'toBeEnabled', 'toBeEmptyDOMElement', 'toBeInTheDocument', and 'toBeInvalid'. Below this is a code snippet from a test file.

After modifying the test case, rerun the test:

The screenshot shows a code editor interface with a dark theme. On the left is an 'EXPLORER' sidebar showing project files like 'node\_modules', 'public', 'src', 'tests', and 'components'. In the center, a file named 'Greet.test.tsx' is open, displaying Jest test code. The right side shows a 'TERMINAL' tab with the output of a command-line run. The terminal output shows the test passed with 1 test file and 1 test passed, starting at 12:46:09 and taking 472ms. The status bar at the bottom indicates 'PASS' and 'Waiting for file changes...'.

Vitest

tests/components/Greet.test.tsx

FAIL (0) / RUNNING (0)  
PASS (2) / SKIP (-)

tests/components/Greet.test.tsx 65ms

- Greet 64ms
  - should render Hello with the name when the name is provided 64ms

Vitest

tests/components/Greet.test.tsx

12:39:07 PM | Greet > should render Hello with the name when the name is provided | stdout  
PASSED as expected.

FAIL (0) / RUNNING (0)  
PASS (2) / SKIP (-)

tests/components/Greet.test.tsx 65ms

- Greet 64ms
  - should render Hello with the name when the name is provided 64ms

Running a negative test (fail the test case):

```

src > components > Greet.tsx > Greet
1 const Greet = ({ name }: { name: string }) => {
2   // if (name) return <h1>Hello {name}</h1>;
3   return <button>Login</button>;
4 };
5
6
7 export default Greet;
8

```

Test Files 1 failed (1)  
Tests 1 failed (1)  
Start at 12:40:45  
Duration 587ms

**FAIL** Tests failed. Watching for file changes...  
press h to show help, press q to quit

Vitest

tests/components/Greet.test.tsx

FAIL (1) / RUNNING (0)  
PASS (0) / SKIP (-)

tests/components/Greet.test.tsx 114ms

- Greet 113ms
  - should render Hello with the name when the name is provided 112ms

**Greet**

should render Hello with the name when the name is provided  
TestingLibraryElementError: Unable to find an accessible element with the role "heading"

Here are the accessible roles:

```

button:
  Name "Login":
    <button />
  
```

-----

Ignored nodes: comments, script, style

```

<body>
  <div>
    <button>
      Login
    </button>
  </div>

```

## Test case 02:

What we are testing...

The screenshot shows two code editor tabs. The left tab is Greet.tsx, containing the following code:

```
src > components > Greet.tsx > Greet > name
1 const Greet = ({ name }: { name: string }) => {
2   if (name) return <h1>Hello {name}</h1>;
3
4   return <button>Login</button>;
5 }
6
7 export default Greet;
```

## Test case and outcome...

The screenshot shows the VS Code interface with the Greet.test.tsx file open in the editor. The file contains the following test cases:

```
tests > components > Greet.test.tsx > describe('Greet') callback > it('should render login button when a name is not provided') callback
6
7  /* Test Case 01 */
8  /* Expected result is to display "Hello" with the name when a name is provided */
9  describe ('Greet', () => {
10    it('should render Hello with the name when the name is provided', () => {
11      render(<Greet name="John"/>);
12      const heading = screen.getByRole('heading');
13      expect(heading).toBeInTheDocument();
14      expect(heading).toHaveTextContent(/Hello John/i);
15      console.log('PASSED as expected.');
16    })
17
18  /* Test Case 02 */
19  /* Expected result is to display login button when a name is not provided */
20  it('should render login button when a name is not provided', () => {
21    render(<Greet/>);
22    const button = screen.getByRole('button');
23    expect(button).toBeInTheDocument();
24    expect(button).toHaveTextContent(/Login/i);
25    console.log('PASSED as expected.');
26  })
27 })
```

The terminal below shows the test results:

```
stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

✓ tests/components/Greet.test.tsx (2)
  ✓ Greet (2)
    ✓ should render Hello with the name when the name is provided
    ✓ should render login button when a name is not provided
```

A red arrow points to the second test case in the code editor, and another red arrow points to the second test case in the terminal output.

The screenshot shows a browser window running a Vitest test runner. The left sidebar shows the test tree:

- Vitest
- Filter: Fail, Pass, Skip, Only Tests
- FAIL (0) / RUNNING (0)
- PASS (2) / SKIP (-)
- tests/components/Greet.test.tsx 69ms
  - Greet 68ms
    - should render Hello with the name when the name is provided 59ms
    - should render login button when a name is not provided 8ms

The right panel shows the test logs:

```
1:47:13 PM | Greet > should render Hello with the name when the name is provided | stdout
PASSED as expected.

1:47:13 PM | Greet > should render login button when a name is not provided | stdout
PASSED as expected.
```

Running a negative test (fail the test case):

The screenshot shows a VS Code interface with two tabs open: `Greet.tsx` and `Greet.test.tsx`. The `Greet.tsx` file contains the following code:

```
1 const Greet = ({ name }: { name?: string }) => {
2   if (name) return <h1>Hello {name}</h1>;
3   // return <button>Login</button>;
4 };
5
6
7 export default Greet;
8
```

A red arrow points to the third line of the component's body, which is currently commented out. The `Greet.test.tsx` file contains the following test code:

```
1 import { render } from '@testing-library/react';
2 import { screen } from '@testing-library/dom';
3 import Greet from './Greet';
4
5 describe('Greet', () => {
6   it('should render login button when a name is not provided', () => {
7     render(<Greet/>);
8     const button = screen.getByRole('button');
9     expect(button).toBeInTheDocument();
10    expect(button).toHaveTextContent(/login/i);
11  });
12});
```

The terminal window at the bottom shows the test results:

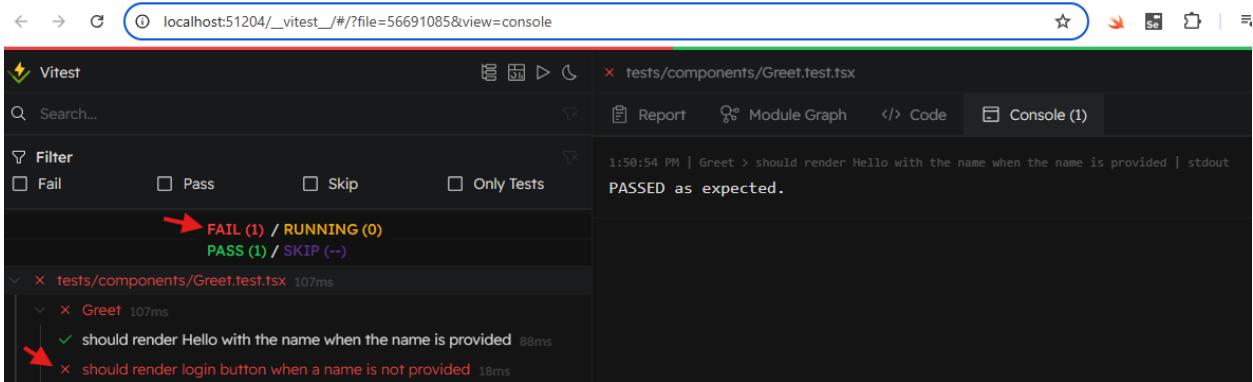
```
FAIL tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
TestingLibraryElementError: Unable to find an accessible element with the role "button"

Here are the accessible roles:
heading:
Name "Hello John":
<h1 />

-----
Ignored nodes: comments, script, style
<body>
  <div>
    <h1>
      Hello
      John
    </h1>
  </div>
  <div />
</body>
> Object.getPrototypeOf(node).constructor.name:config.js:37:19
> node_modules/@testing-library/dom/dist/query-helpers.js:76:38
> node_modules/@testing-library/dom/dist/query-helpers.js:52:17
> node_modules/@testing-library/dom/dist/query-helpers.js:95:19
> tests/components/Greet.test.tsx:22:31
20|   it('should render login button when a name is not provided', () => {
21|     render(<Greet/>);
22|     const button = screen.getByRole('button');
23|     expect(button).toBeInTheDocument();
24|     expect(button).toHaveTextContent(/login/i);

Test Files 1 failed (1)
Tests 1 failed | 1 passed (2)
Start at 13:50:52
Duration 568ms

FAIL Tests failed. Watching for file changes...
press h to show help, press q to quit
```



## 5. Simplifying the test setup

In the vitest.config.ts file, setting the globals for Vitest to true...

```
vitest.config.ts M | Greet.tsx M | Greet.test.tsx U
vitest.config.ts > [?] default
1 import {defineConfig} from 'vitest/config';
2
3 export default defineConfig({
4   test: {
5     environment: 'jsdom',
6     // Red arrow points here
7     globals: true
8   });

```

So no need to have this line in every test file...

```
vitest.config.ts M | Greet.test.tsx U | Greet.tsx M
tests > components > Greet.test.tsx > describe('Greet') callback > it('should
1 /* Import functions and libraries */
2 // Red arrow points here
3 import { it, expect, describe } from 'vitest';
4 import {render, screen} from '@testing-library/react';
5 import Greet from '../../../../../src/components/Greet';
6 import '@testing-library/jest-dom/vitest';

```

After removing...

```
vitest.config.ts M | Greet.test.tsx 7, U | Greet.tsx M
tests > components > Greet.test.tsx ...
1 /* Import functions and libraries */
2 import {render, screen} from '@testing-library/react';
3 import Greet from '../../../../../src/components/Greet';
4 import '@testing-library/jest-dom/vitest';
5

```

Another item or error to address...

```
vitest.config.ts M | Greet.test.tsx 7, U | tsconfig.json M | Greet.tsx M
tests > components > Greet.test.tsx > describe('Greet') callback > it('should render Hello with the name when the name is provided') callback
1 /* Import functions and libraries */
2 import {render, screen} from '@testing-library/react';
3 // Red box highlights this line
4 Cannot find name 'describe'. Do you need to install type definitions for a test runner? Try
5 `npm i --save-dev @types/jest` or `npm i --save-dev @types/mocha`. ts(2582)
6 any
7 View Problem (Alt+F8) Quick Fix... (Ctrl+.)
8 describe ('Greet', () => {
9   // Red arrow points here
10   it('should render Hello with the name when the name is provided', () => [

```

Update tsconfig.json file...

```
ts vitest.config.ts M Greet.test.tsx U tsconfig.json M Greet.tsx M
tsconfig.json > {} compilerOptions
1 {
2   "compilerOptions": {
3     "target": "ES2020",
4     "useDefineForClassFields": true,
5     "lib": ["ES2020", "DOM", "DOM.Iterable"],
6     "module": "ESNext",
7     "skipLibCheck": true,
8     "types": ["vitest/globals"], ↴
```

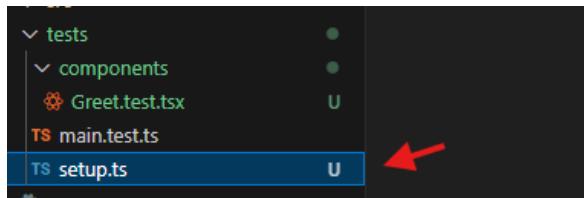
After updating the config file...

```
TS vitest.config.ts M Greet.test.tsx U tsconfig.json M Greet.tsx M
tests > components > Greet.test.tsx > describe('Greet') callback > it('should render Hello with the name when the name is provided', () => {
1   /* Import functions and libraries */
2   import {render, screen} from '@testing-library/react';
3   import Greet from '../../../../../src/components/Greet';
4   import '@testing-library/jest-dom/vitest';
5
6   /* Test Case 01 */
7   /* Expected result is to display "Hello" with the name when a name is provided */
8   describe ('Greet', () => {
9     it('should render Hello with the name when the name is provided', () => {
10       render(<Greet name="John"/>);
11       const heading = screen.getByRole('heading');
12       expect(heading).toBeInTheDocument();
13       expect(heading).toHaveTextContent('Hello John!'); ↴
```

Another import line to move into global setting...

```
TS vitest.config.ts M Greet.test.tsx U TS setup.ts U tsconfig.json M
tests > components > Greet.test.tsx > describe('Greet') callback > it('should render Hello with the name when the name is provided', () => {
1   /* Import functions and libraries */
2   import {render, screen} from '@testing-library/react';
3   import Greet from '../../../../../src/components/Greet';
4   import '@testing-library/jest-dom/vitest'; ↴
```

So, in test folder, create a setup.ts file...



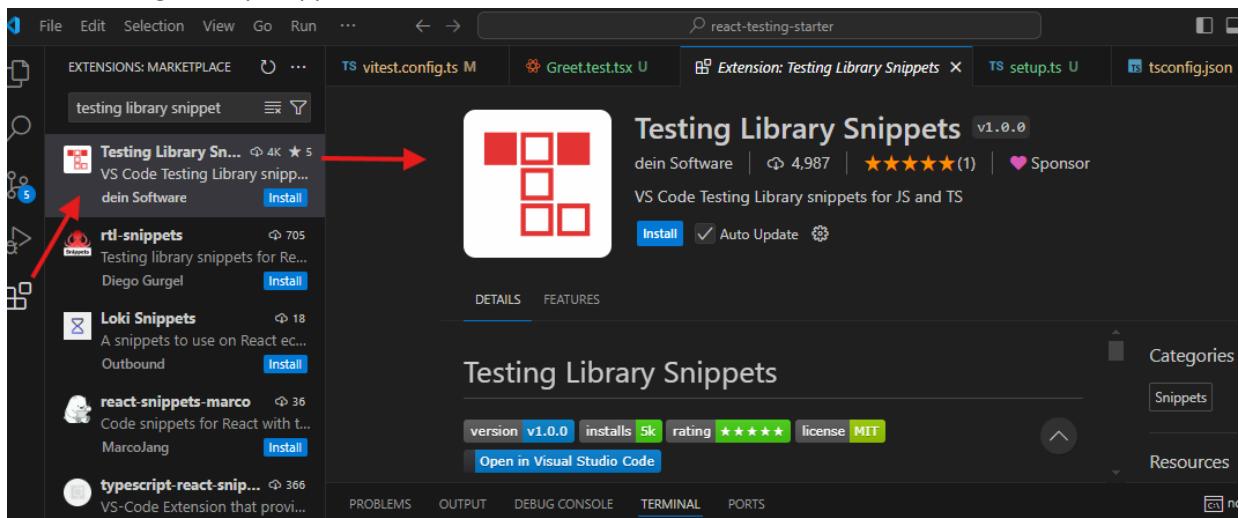
Move the import line into the setup file...

```
TS vitest.config.ts M Greet.test.tsx U TS setup.ts U
tests > TS setup.ts
1 import '@testing-library/jest-dom/vitest';
```

Reference the file from the config file...

```
TS vitest.config.ts M Greet.test.tsx U TS setup.ts U tsconfig.json M
vitest.config.ts > default
1 import {defineConfig} from 'vitest/config';
2
3 export default defineConfig({
4   test: {
5     environment: 'jsdom',
6     globals: true,
7     setupFiles: 'tests/setup.ts', ↴
8   },
9});
```

## Install Testing Library Snippets extension...



Now, instead of typing this long line...

```
> components > ❁ Greet.test.tsx > ...
/* Import functions and libraries */
import { render, screen } from '@testing-library/react' ↑
import Greet from '../../../../../src/components/Greet';
```

Simply key in "itr" and hit enter...

```
tests > components > ❁ Greet.test.tsx > ...
1  /* Import functions and libraries */
2  itr ↑
3  imp □ itr import.testing-library.react
4  □ itrh import.testing-library.render-hook
< components > ❁ greetest.tsx > ...
/* Import functions and libraries */
import { render, screen } from '@testing-library/react' ↑
import Greet from '../../../../../src/components/Greet';
```

To ensure that all the changes are properly setup and test cases still passed, run the test:

```
C:\Users\njmllo\react-testing-starter>npm run test:ui

> react-testing-vite@0.0.0 test:ui
> vitest --ui

[DEV] v2.1.5 C:/Users/njmllo/react-testing-starter
UI started at http://localhost:51204/_vitest_/

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)

Test Files 2 passed (2)
  Tests 3 passed (3)
  Start at 14:27:12
  Duration 3.49s (transform 135ms, setup 649ms, collect 376ms, tests 162ms, environment 2.42s, prepare 431ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```

Vitest

Dashboard

Search...

Filter

FAIL (0) / RUNNING (0)

PASS (2) / SKIP (0)

tests/components/Greet.test.tsx 157ms

Greet 156ms

should render Hello wit... 135ms

should render login but... 21ms

tests/main.test.ts 5ms

3 Pass

0 Fail

3 Total

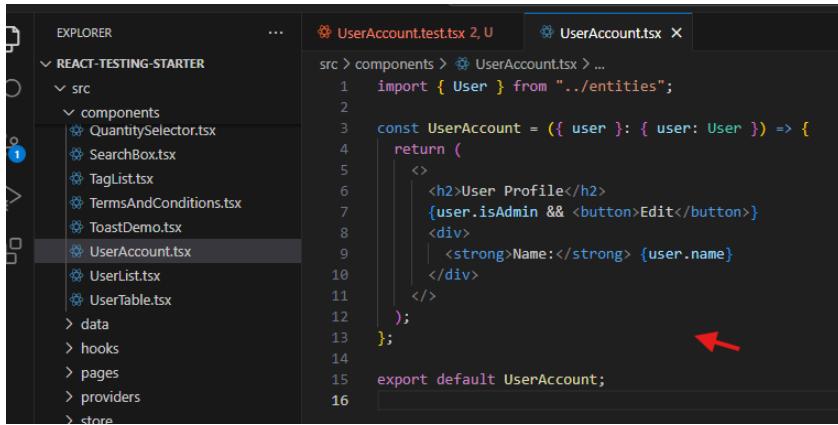
Files 2

Pass 2

Time 4.04s

## 6. Testing User Account

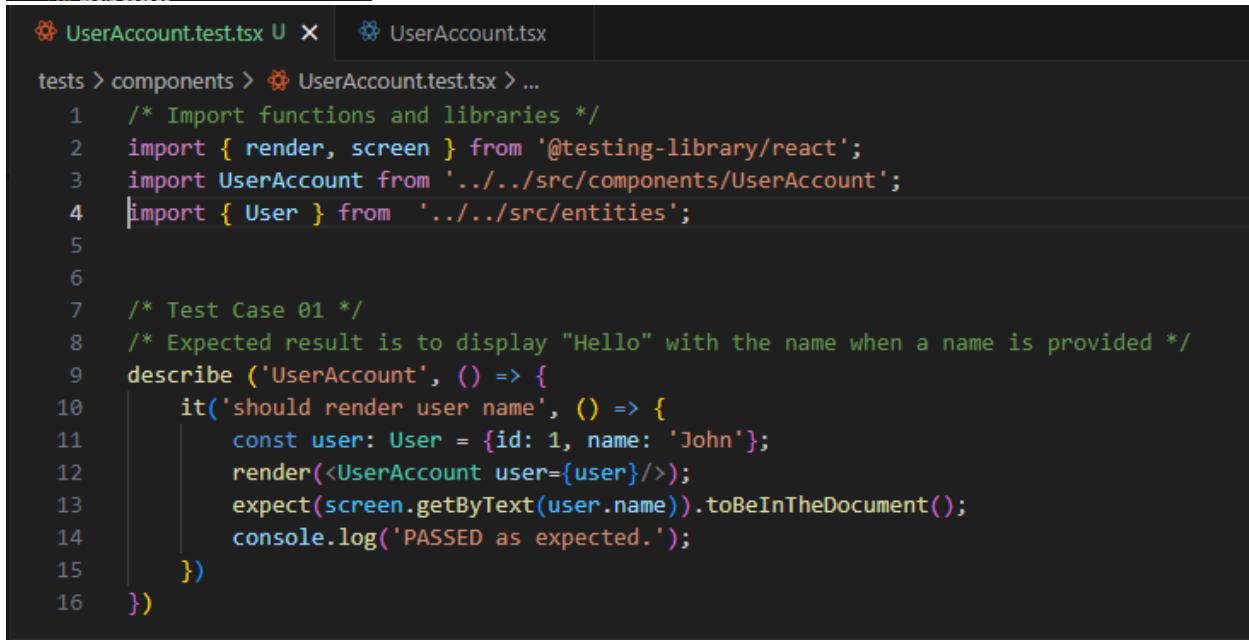
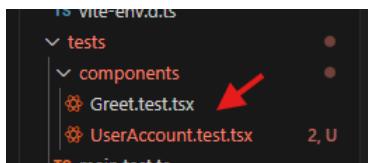
### Component to test: UserAccount.tsx



```
src > components > UserAccount.tsx ...
1 import { User } from "../entities";
2
3 const UserAccount = ({ user }: { user: User }) => {
4   return (
5     <>
6       <h2>User Profile</h2>
7       {user.isAdmin && <button>Edit</button>}
8       <div>
9         <strong>Name:</strong> {user.name}
10        </div>
11      </>
12    );
13  };
14
15 export default UserAccount;
```

### Test case 01:

Create a test file...



```
UserAccount.test.tsx U X UserAccount.tsx
tests > components > UserAccount.test.tsx ...
1  /* Import functions and libraries */
2  import { render, screen } from '@testing-library/react';
3  import UserAccount from '../../../../../src/components/UserAccount';
4  import { User } from '../../../../../src/entities';

5
6
7  /* Test Case 01 */
8  /* Expected result is to display "Hello" with the name when a name is provided */
9  describe ('UserAccount', () => {
10    it('should render user name', () => {
11      const user: User = {id: 1, name: 'John'};
12      render(<UserAccount user={user}/>);
13      expect(screen.getByText(user.name)).toBeInTheDocument();
14      console.log('PASSED as expected.');
15    })
16  })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Microsoft Windows [Version 10.0.22631.4460]  
(c) Microsoft Corporation. All rights reserved.

```
C:\Users\njml0\react-testing-starter>npm run test:ui
> react-testing-vite@0.0.0 test:ui
> vitest --ui

DEV v2.1.5 C:/Users/njml0/react-testing-starter
UI started at http://localhost:51204/_vitest_/

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render user name
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/UserAccount.test.tsx (1)

Test Files 3 passed (3)
Tests 4 passed (4)
Start at 16:29:21
Duration 3.65s (transform 179ms, setup 988ms, collect 722ms, tests 166ms, environment 4.05s, prepare 839ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```

← → ⌂ localhost:51204/\_vitest\_/#/ star Bookmark Share Copy

**Vitest** Dashboard

Search... ▼

Filter ▼

Fail  Pass  
 Skip  Only Tests

**FAIL (0) / RUNNING (0)**  
**PASS (3) / SKIP (-)**

- > ✓ tests/components/Greet.test.tsx 95ms
- ✓ tests/components/UserAccount.test.tsx 54ms
  - ✓ UserAccount 52ms
    - ✓ should render user name 51ms ↗
- > ✓ tests/main.test.ts 4ms

**4** **0** **4**  
Pass Fail Total

Files 3  
Pass 3  
Time 6.35s

Running a negative test (fail the test case):

The screenshot shows the VS Code interface with two tabs open: "UserAccount.test.tsx" and "UserAccount.tsx". The "UserAccount.test.tsx" tab contains the following code:

```
src > components > UserAccount.tsx > UserAccount
1 import { User } from "../entities";
2
3 const UserAccount = ({ user }: { user: User }) => {
4   return (
5     <>
6       <h2>User Profile</h2>
7       {user.isAdmin && <button>Edit</button>}
8       <div>
9         /* <strong>Name:</strong> {user.name} */
10        </div>
11      </>
12    );
13  };
14
```

A red arrow points to the line containing the comment `/\* <strong>Name:</strong> {user.name} \*/` in the code editor.

The "Terminal" tab shows the output of the test run:

```
Failed Tests 1
```

**FAIL** tests/components/UserAccount.test.tsx > UserAccount > should render user name  
TestingLibraryElementError: Unable to find an element with the text: John. This could be because the text is broken up by multiple elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

Ignored nodes: comments, script, style

```
<body>
  <div>
    <h2>
      User Profile
    </h2>
    <div />
  </div>
</body>
> Object.getPrototypeOf(node).name
> node_modules/@testing-library/dom/dist/query-helpers.js:76:38
> node_modules/@testing-library/dom/dist/query-helpers.js:52:17
> node_modules/@testing-library/dom/dist/query-helpers.js:95:19
> tests/components/UserAccount.test.tsx:13:23
  11|   const user = {id: 1, name: 'John'};
  12|   render(<UserAccount user={user}/>);
  13|   expect(screen.getByText(user.name)).toBeInTheDocument();
  14|   console.log('PASSED as expected.');
  15| })
```

[1/1]-

Test Files 1 failed (1)  
Tests 1 failed (1)  
Start at 16:31:49  
Duration 482ms

**FAIL** Tests failed. Watching for file changes...  
press h to show help, press q to quit

The screenshot shows the Vitest UI in a browser window at [localhost:51204/\\_vitest\\_/#?file=tests/components/UserAccount.test.tsx](http://localhost:51204/_vitest_/#?file=tests/components/UserAccount.test.tsx). The left sidebar shows a tree view of test files and their status. The right panel displays the test results for "UserAccount".

**UserAccount**

should render user name  
TestingLibraryElementError: Unable to find an element with the text: John. This could be because the text is broken up by multiple elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

Ignored nodes: comments, script, style

```
<body>
  <div>
    <h2>
      User Profile
    </h2>
    <div />
  </div>
</body>
```

TestingLibraryElementError: Unable to find an element with the text: John. This could be because the text is broken up by multiple elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

## Test case 02:

VS Code Explorer showing the project structure of REACT-TESTING-STARTER. The UserAccount.test.tsx file is open in the editor. The code contains two test cases: 'should render user name' and 'should render edit button if user is admin'. The second test case is highlighted with a red arrow.

```
/* Import functions and libraries */
import { render, screen } from '@testing-library/react';
import UserAccount from '../../../../../src/components/UserAccount';
import { User } from '../../../../../src/entities';

/* Test Case 01 */
/* Expected result is it should render user name */
describe ('UserAccount', () => {
  it('should render user name', () => {
    const user = {id: 1, name: 'John'};
    render(<UserAccount user={user}>);
    expect(screen.getByText(user.name)).toBeInTheDocument();
    console.log('PASSED as expected.');
  })
}

/* Test Case 02 */
/* Expected result is it should render edit button if user is admin */
it('should render edit button if user is admin', () => {
  const user = {id: 1, name: 'John', isAdmin: true};
  render(<UserAccount user={user}>);
  const button = screen.getByRole('button');
  expect(button).toBeInTheDocument();
  expect(button).toHaveTextContent(/edit/i);
  console.log('PASSED as expected.');
})
```

Vitest UI dashboard showing the test results. The results table indicates 5 Passes, 0 Failures, and a total of 5 tests. The 'Dashboard' section shows the test tree with two test files: Greet.test.tsx and UserAccount.test.tsx. The UserAccount test has two passing test cases: 'should render user name' and 'should render edit button if user is admin'. A red arrow points to the second test case in the UserAccount test tree.

Pass	Fail	Total
5	0	5

Files: 3  
Pass: 3  
Time: 9.28s

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

press **h** to show help, press **q** to quit

```
C:\Users\njml0\react-testing-starter>npm run test:ui

> react-testing-vite@0.0.0 test:ui
> vitest --ui

DEV v2.1.5 C:/Users/njml0/react-testing-starter
UI started at http://localhost:51204/_vitest_/

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render user name
PASSED as expected.

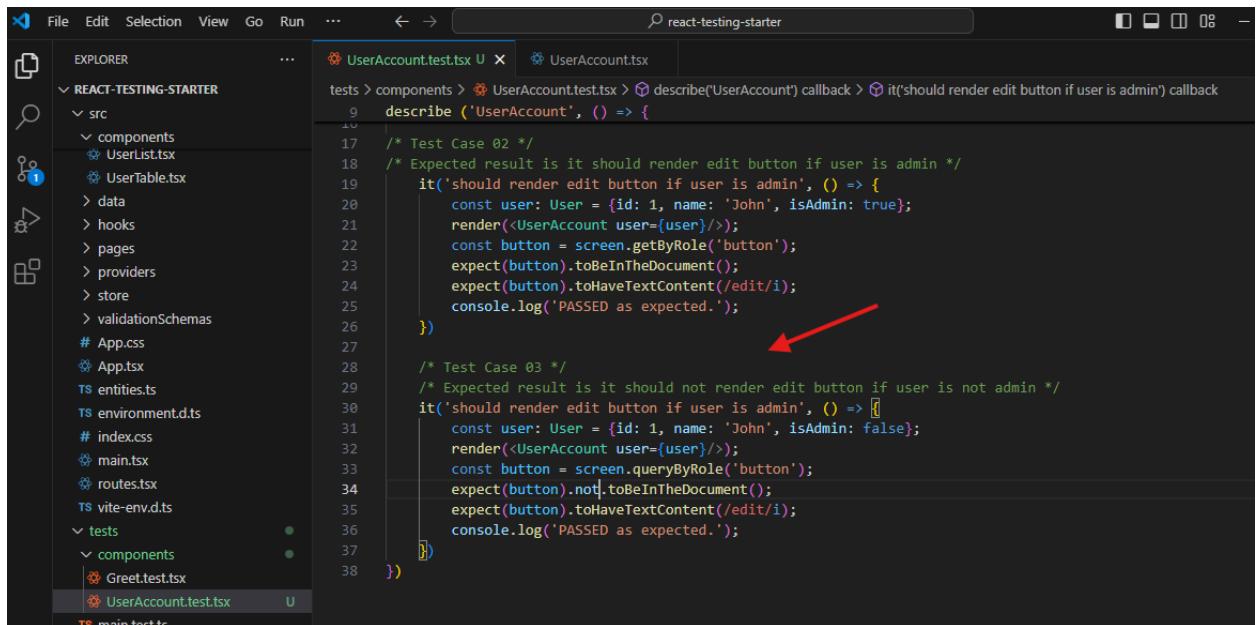
stdout | tests/components/UserAccount.test.tsx > UserAccount > should render edit button if user is admin
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/UserAccount.test.tsx (2)

Test Files 3 passed (3)
Tests 5 passed (5)
Start at 16:44:01
Duration 3.75s (transform 260ms, setup 921ms, collect 723ms, tests 221ms, environment 4.21s, prepare 3.20s)

PASS Waiting for file changes...
press h to show help, press q to quit
```

### Test case 03:



```
File Edit Selection View Go Run ... react-testing-starter
EXPLORER REACT-TESTING-STARTER ...
src components UserList.tsx UserTable.tsx
data hooks pages providers store validationSchemas
App.css App.tsx entities.ts environment.d.ts
index.css main.tsx routes.tsx vite-env.d.ts
tests components Greet.test.tsx UserAccount.test.tsx
main.test.ts

UserAccount.test.tsx (U) 17 /* Test Case 02 */
18 /* Expected result is it should render edit button if user is admin */
19 it('should render edit button if user is admin', () => {
20   const user = {id: 1, name: 'John', isAdmin: true};
21   render(<UserAccount user={user}>);
22   const button = screen.getByRole('button');
23   expect(button).toBeInTheDocument();
24   expect(button).toHaveTextContent(/edit/i);
25   console.log('PASSED as expected.');
26 }
27
28 /* Test Case 03 */
29 /* Expected result is it should not render edit button if user is not admin */
30 it('should render edit button if user is admin', () => [
31   const user = {id: 1, name: 'John', isAdmin: false};
32   render(<UserAccount user={user}>);
33   const button = screen.queryByRole('button');
34   expect(button).not.toBeInTheDocument();
35   expect(button).not.toHaveTextContent(/edit/i);
36   console.log('PASSED as expected.');
37 ])
38 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(c) Microsoft Corporation. All rights reserved.

```
C:\Users\njmlo\react-testing-starter>npm run test:ui
> react-testing-vite@0.0.0 test:ui
> vitest --ui

DEV v2.1.5 C:/Users/njmlo/react-testing-starter
UI started at http://localhost:51204/_vitest_/

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render user name
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render edit button if user is admin
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should not render edit button if user is not admin
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/UserAccount.test.tsx (3)

Test Files 3 passed (3)
Tests 6 passed (6)
Start at 16:56:56
Duration 3.70s (transform 179ms, setup 952ms, collect 766ms, tests 255ms, environment 3.90s, prepare 785ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```

Vitest

localhost:51204/\_vitest\_/#/

Filter

Fail     Pass     Skip     Only Tests

FAIL (0) / RUNNING (0)  
PASS (3) / SKIP (-)

- tests/components/Greet.test.tsx 125ms
  - Greet 124ms
- tests/components/UserAccount.test.tsx 125ms
  - UserAccount 123ms
    - should render user name 55ms
    - should render edit button if user is admin 63ms
    - should not render edit button if user is not admin 4ms
- tests/main.test.ts 6ms

6 Pass    0 Fail    6 Total

Files	3
Pass	3
Time	6.65s

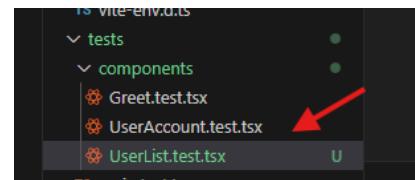
## 7. Testing Lists

### Component to test: UserList.tsx

```
❶ userList.tsx X
src > components > userList.tsx > ...
1  import { User } from "../entities";
2
3  const userList = ({ users }: { users: User[] }) => {
4    if (users.length === 0) return <p>No users available.</p>;
5
6    return (
7      <ul>
8        {users.map((user) => (
9          <li key={user.id}>
10            <a href={`/users/${user.id}`}>{user.name}</a>
11          </li>
12        )));
13      </ul>
14    );
15  };
16
17  export default userList;
18
```

### Test case 01:

Create a test file...



```
❷ userList.test.tsx U X
tests > components > userList.test.tsx > ...
1  import { render, screen } from '@testing-library/react';
2  import userList from '../../../../../src/components/UserList';
3
4  /* Test Case 01 */
5  /* Expected result is it should render no users when the users array is empty */
6  describe ('UserList', () => {
7    it('should render no users when the users array is empty', () => {
8      render(<UserList users={[ ]}/>);
9      expect(screen.getByText('No users')).toBeInTheDocument();
10     console.log('PASSED as expected.');
11   });
12})
```

```

> react-testing-vite@0.0.0 test:ui
> vitest --ui

[DEV] v2.1.5 C:/Users/njmlo/react-testing-starter
UI started at http://localhost:51204/_vitest_/

stdout | tests/components/UserList.test.tsx > UserList > should render no users when the users array is empty
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render user name
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render edit button if user is admin
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should not render edit button if user is not admin
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/UserAccount.test.tsx (3)
✓ tests/components/UserList.test.tsx (1)

Test Files 4 passed (4)
Tests 7 passed (7)
Start at 11:30:58
Duration 4.04s (transform 202ms, setup 1.51s, collect 1.36s, tests 381ms, environment 6.14s, prepare 1.18s)

PASS Waiting for file changes...
press h to show help, press q to quit

```

Vitest

Dashboard

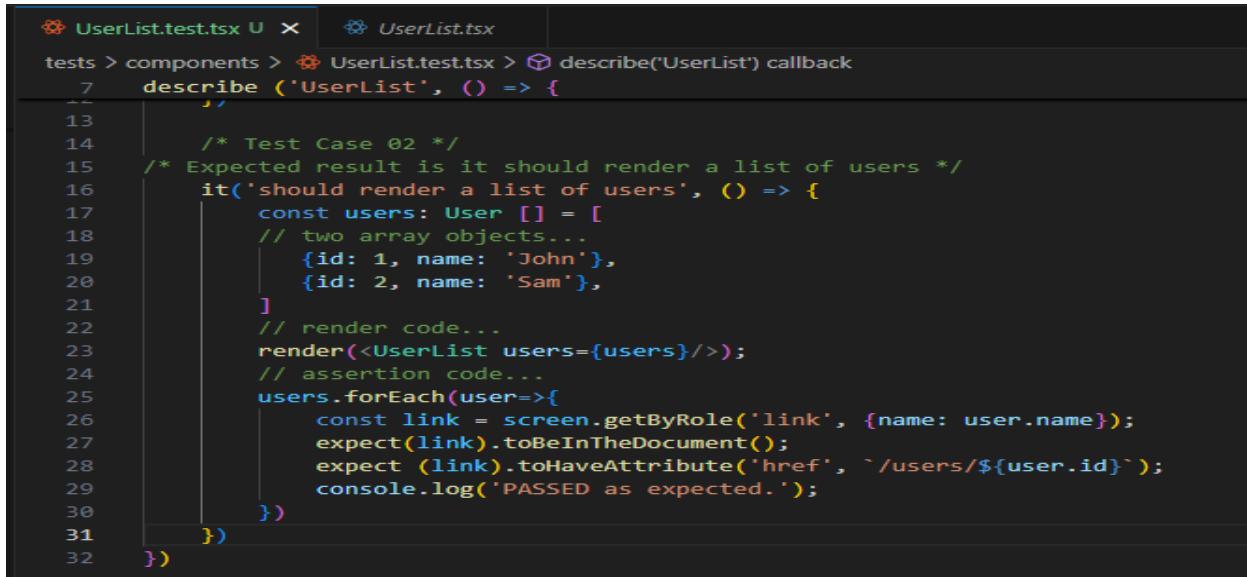
FAIL (0) / RUNNING (0)  
PASS (4) / SKIP (-)

- > ✓ tests/components/Greet.test.tsx 124ms
- > ✓ tests/components/UserAccount.test.tsx 186ms
- ✓ tests/components/UserList.test.tsx 64ms
  - ✓ UserList 63ms
    - ✓ should render no users when the users array is empty 62ms
- > ✓ tests/main.test.ts 6ms

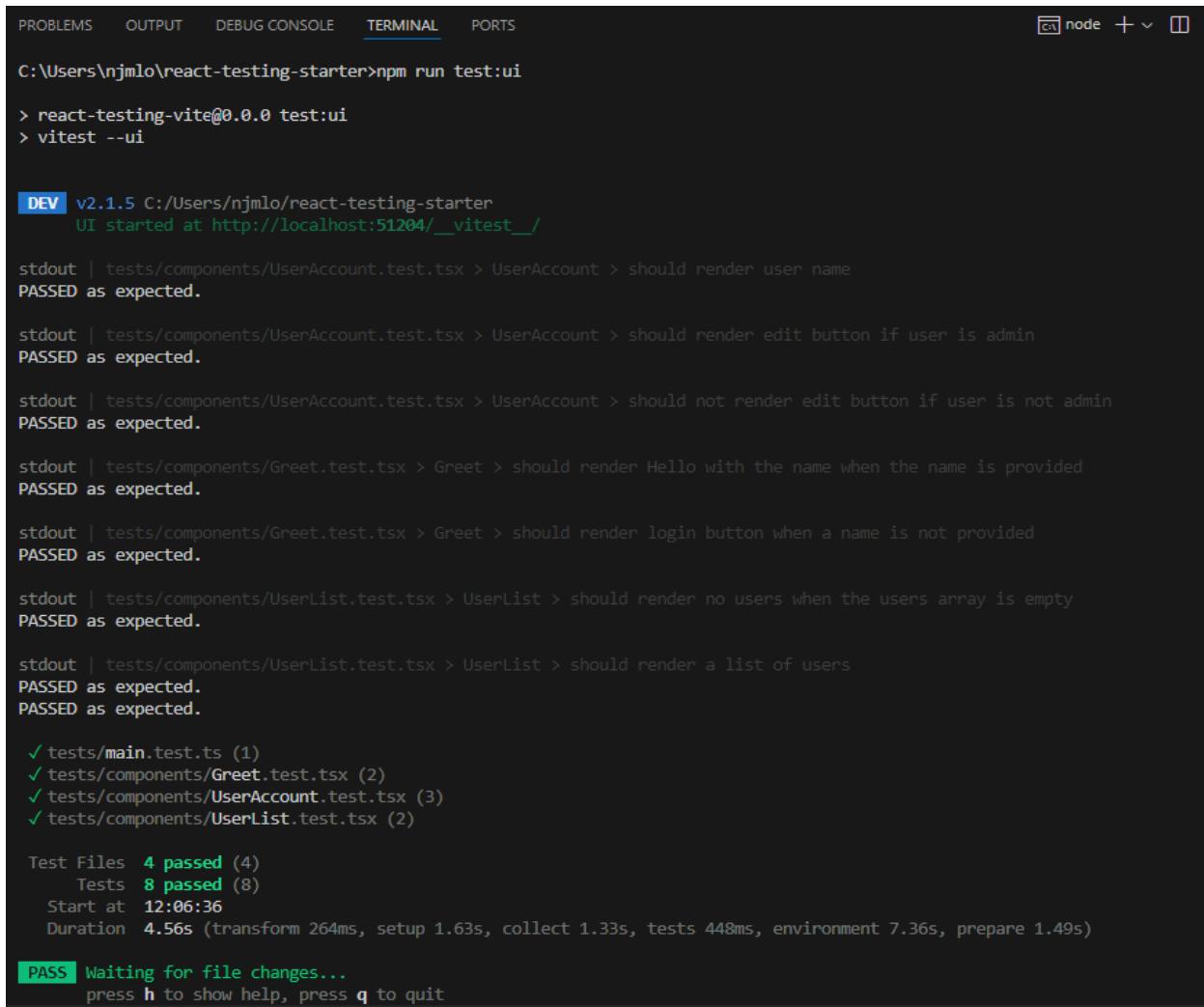
Pass	Fail	Total
7	0	7

Files: 4  
Pass: 4  
Time: 10.57s

## Test case 02:



```
tests > components > UserList.test.tsx > describe('UserList') callback
  7   describe (''UserList'', () => {
  8     /* Test Case 02 */
  9     /* Expected result is it should render a list of users */
 10     it('should render a list of users', () => {
 11       const users: User [] = [
 12         // two array objects...
 13         {id: 1, name: 'John'},
 14         {id: 2, name: 'Sam'},
 15       ]
 16       // render code...
 17       render(<UserList users={users}/>);
 18       // assertion code...
 19       users.forEach(user=>{
 20         const link = screen.getByRole('link', {name: user.name});
 21         expect(link).toBeInTheDocument();
 22         expect (link).toHaveAttribute('href', `/users/${user.id}`);
 23         console.log('PASSED as expected.');
 24       })
 25     })
 26   })
 27 })
 28 })
 29 })
 30 })
 31 })
 32 })
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
C:\Users\njml0\react-testing-starter>npm run test:ui
```

```
> react-testing-vite@0.0.0 test:ui
> vitest --ui
```

```
DEV v2.1.5 C:/Users/njml0/react-testing-starter
UI started at http://localhost:51204/_vitest_/_
```

```
stdout | tests/components/UserAccount.test.tsx > UserAccount > should render user name
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should render edit button if user is admin
PASSED as expected.

stdout | tests/components/UserAccount.test.tsx > UserAccount > should not render edit button if user is not admin
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

stdout | tests/components/UserList.test.tsx > UserList > should render no users when the users array is empty
PASSED as expected.

stdout | tests/components/UserList.test.tsx > UserList > should render a list of users
PASSED as expected.
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/UserAccount.test.tsx (3)
✓ tests/components/UserList.test.tsx (2)

Test Files 4 passed (4)
  Tests 8 passed (8)
  Start at 12:06:36
  Duration 4.56s (transform 264ms, setup 1.63s, collect 1.33s, tests 448ms, environment 7.36s, prepare 1.49s)
```

```
PASS Waiting for file changes...
press h to show help, press q to quit
```

Vitest

Search...

Filter

FAIL (0) / RUNNING (0)  
PASS (4) / SKIP (-)

- tests/components/Greet.test.tsx 143ms
- tests/components/UserAccount.test.tsx 136ms
- tests/components/UserList.test.tsx 164ms
  - UserList 163ms
    - should render no users when the users array is empty 64ms
    - should render a list of users 98ms (red arrow)
- tests/main.test.ts 5ms

**8** Pass    **0** Fail    **8** Total

Files	4
Pass	4
Time	12.27s

Running a negative test (fail the test case):

```

UserList.test.tsx U UserListtsx M X
src > components > UserList.tsx > UserList > users.map() callback
1 import { User } from "../entities";
2
3 const UserList = ({ users }: { users: User[] }) => {
4   if (users.length === 0) return <p>No users available.</p>;
5
6   return (
7     <ul>
8       <li key={user.id}>
9         <a href={`/users/${user.id}`}>{user.name}</a>
10        /*<a href={`/users/${user.id}`}>{user.name}</a> */
11      </li>
12    </ul>
13  );
14}
15
16 export default UserList;
17
18

```

## Failed Tests 1

**FAIL** tests/components/UserList.test.tsx > userList > should render a list of users  
**TestingLibraryElementError**: Unable to find an accessible element with the role "link" and name "John"

Here are the accessible roles:

list:

```
Name "":
<ul />
```

listitem:

```
Name "":
<li />
```

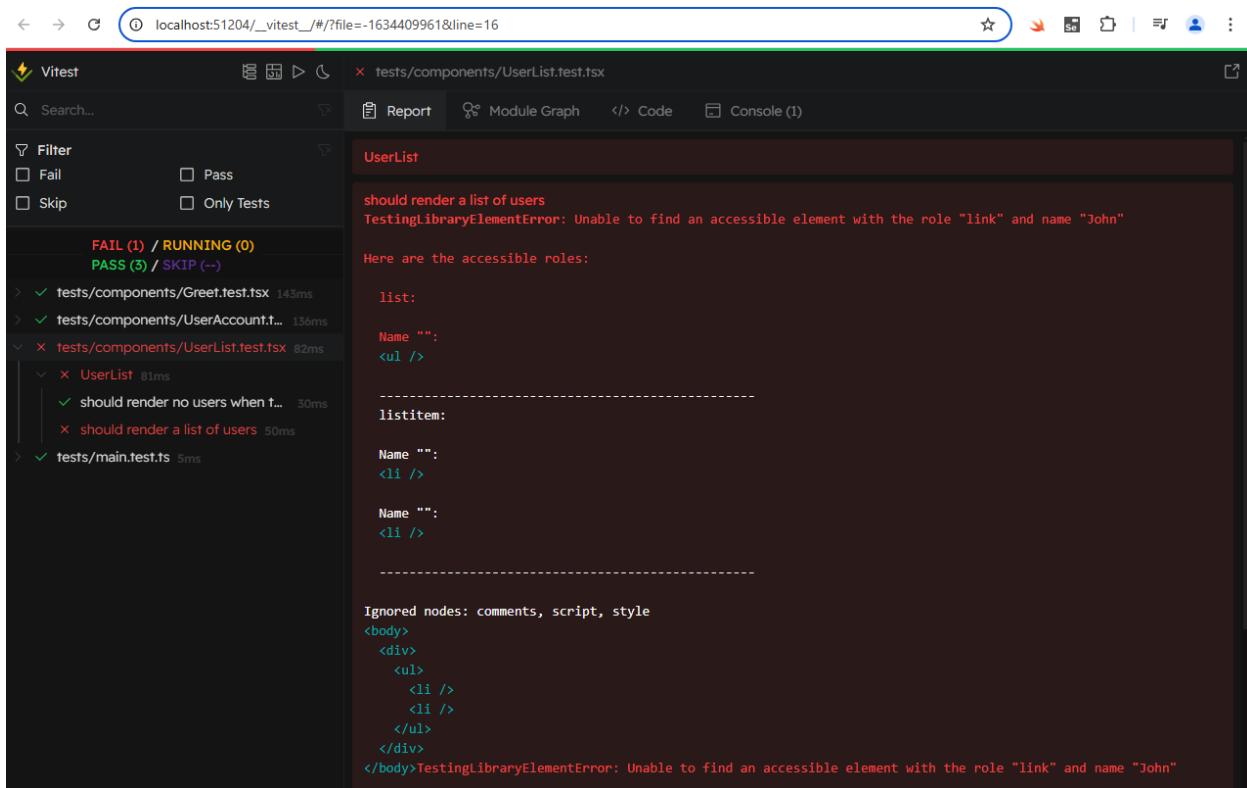
```
Name "":
<li />
```

Ignored nodes: comments, script, style

```
<body>
  <div>
    <ul>
      <li />
      <li />
    </ul>
  </div>
</body>
> Object.getElementError node_modules@testing-library/dom/dist/config.js:37:19
> node_modules@testing-library/dom/dist/query-helpers.js:76:38
> node_modules@testing-library/dom/dist/query-helpers.js:52:17
> node_modules@testing-library/dom/dist/query-helpers.js:95:19
> tests/components/UserList.test.tsx:26:33
  24|       // assertion code...
  25|       users.forEach(user=>{
  26|         const link = screen.getByRole('link', {name: user.name});
  27|           ^
  28|           expect(link).toBeInTheDocument();
  29|           expect(link).toHaveAttribute('href', `/users/${user.id}`);
> tests/components/UserList.test.tsx:25:15
```

Test Files **1 failed** (1)  
Tests **1 failed** | **1 passed** (2)  
Start at 12:08:52  
Duration 520ms

**FAIL** Tests failed. Watching for file changes...  
press **h** to show help, press **q** to quit



## 8. Testing ProductImageGallery

Component to test: ProductImageGallery.tsx

EXPLORER

REACT-TESTING-STARTER

- src
- components
- AuthStatus.tsx
- CancelOrderButton.tsx
- CategoryList.tsx
- ErrorMessage.tsx
- ExpandableText.tsx
- Greet.tsx
- Label.tsx
- LanguageSelector.tsx
- LoginButton.tsx
- LogoutButton.tsx
- NavBar.tsx
- Onboarding.tsx
- OrderStatusSelector.tsx
- PrivateRoutes.tsx
- ProductDetail.tsx
- ProductForm.tsx
- ProductImageGallery.tsx

ProductImageGallery.tsx

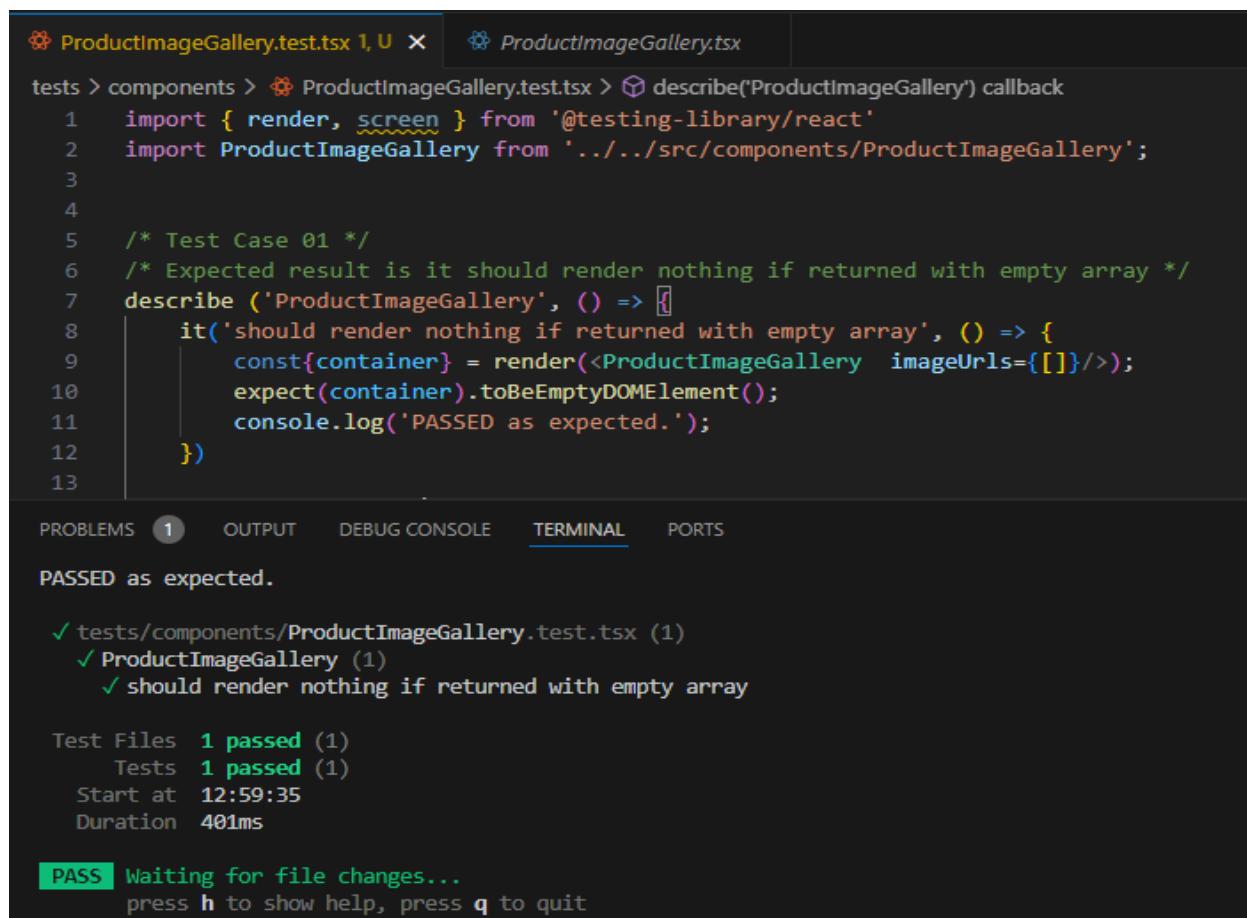
```
const ProductImageGallery = ({ imageUrls }: { imageUrls: string[] }) => {
  if (imageUrls.length === 0) return null;

  return (
    <ul>
      {imageUrls.map((url) => (
        <li key={url}>
          <img src={url} />
        </li>
      ))}
    </ul>
  );
};

export default ProductImageGallery;
```

## Test case 01:

Create a test file...



```
ProductImageGallery.test.tsx 1, U ✘ ProductImageGallery.tsx
tests > components > ProductImageGallery.test.tsx > describe('ProductImageGallery') callback
  1 import { render, screen } from '@testing-library/react'
  2 import ProductImageGallery from '../../../../../src/components/ProductImageGallery';
  3
  4
  5 /* Test Case 01 */
  6 /* Expected result is it should render nothing if returned with empty array */
  7 describe ('ProductImageGallery', () => [
  8   it('should render nothing if returned with empty array', () => {
  9     const{container} = render(<ProductImageGallery imageUrls={[ ]}>);
10     expect(container).toBeEmptyDOMElement();
11     console.log('PASSED as expected.');
12   })
13 ])
```

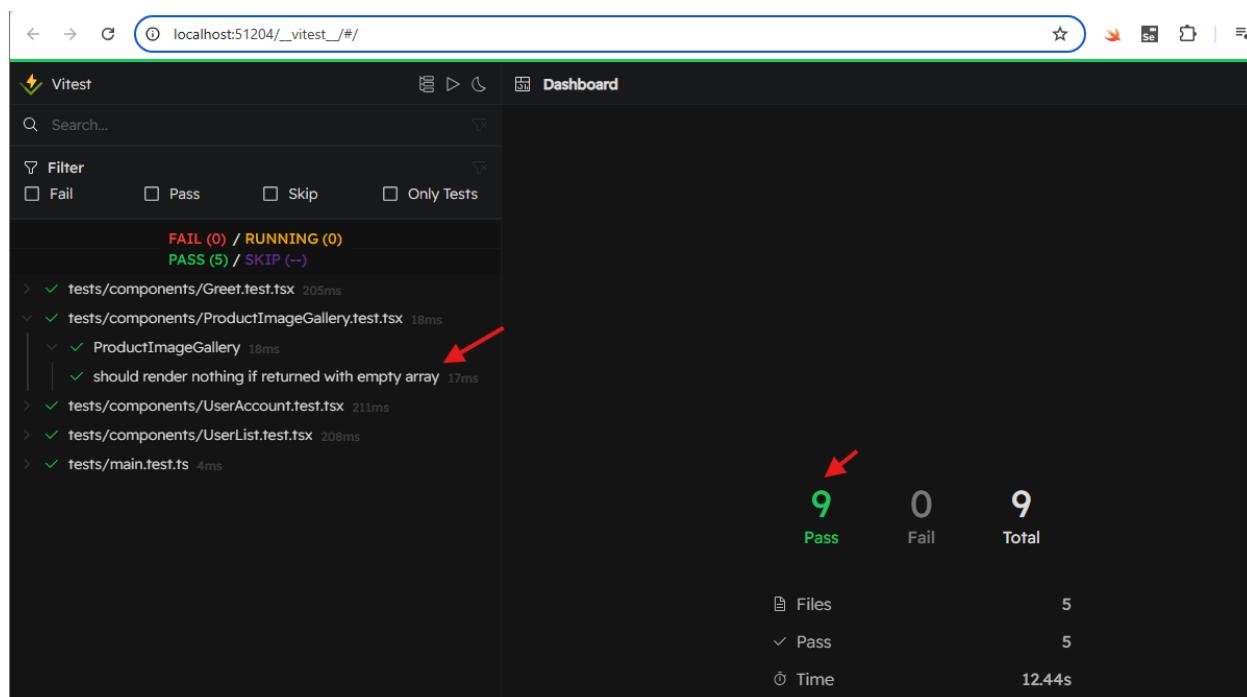
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PASSED as expected.

✓ tests/components/ProductImageGallery.test.tsx (1)  
  ✓ ProductImageGallery (1)  
    ✓ should render nothing if returned with empty array

Test Files 1 passed (1)  
Tests 1 passed (1)  
Start at 12:59:35  
Duration 401ms

PASS Waiting for file changes...  
press h to show help, press q to quit



Vitest

localhost:51204/\_vitest\_/#

FAIL (0) / RUNNING (0)  
PASS (5) / SKIP (-)

- > ✓ tests/components/Greet.test.tsx 205ms
- ✓ tests/components/ProductImageGallery.test.tsx 18ms
  - ✓ ProductImageGallery 18ms
    - ✓ should render nothing if returned with empty array 17ms
- > ✓ tests/components/UserAccount.test.tsx 211ms
- > ✓ tests/components/UserList.test.tsx 208ms
- > ✓ tests/main.test.ts 4ms

9 Pass 0 Fail 9 Total

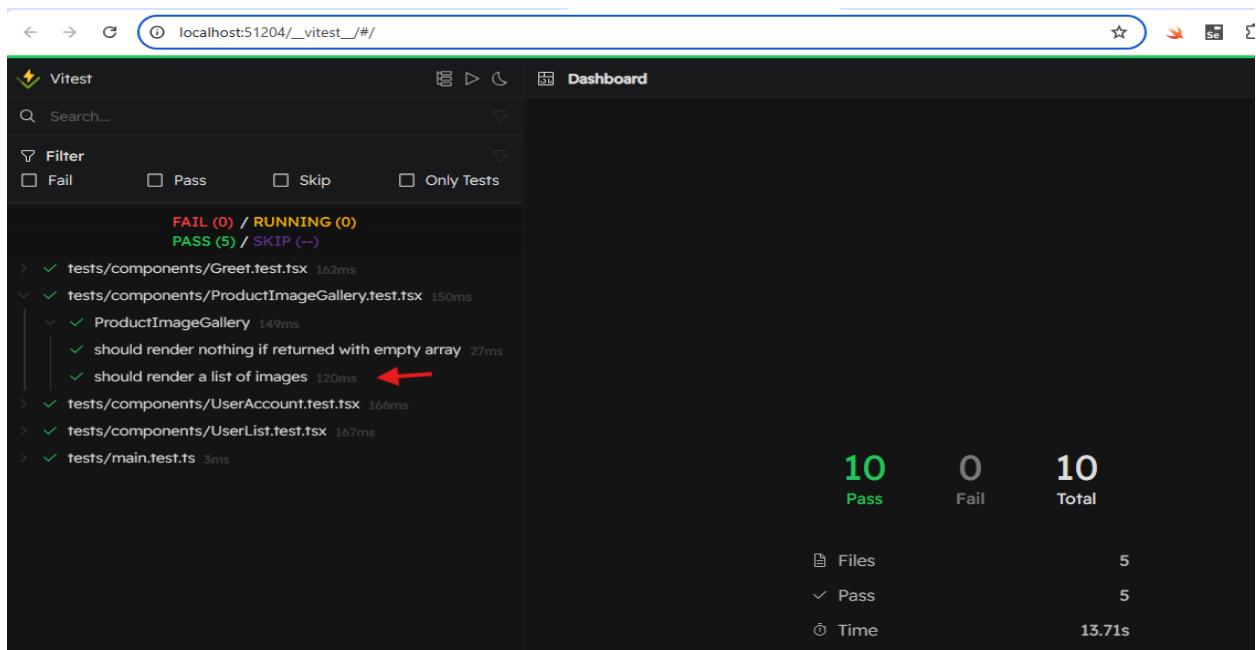
Files 5  
Pass 5  
Time 12.44s

## Test case 02:

The screenshot shows the VS Code interface with the following details:

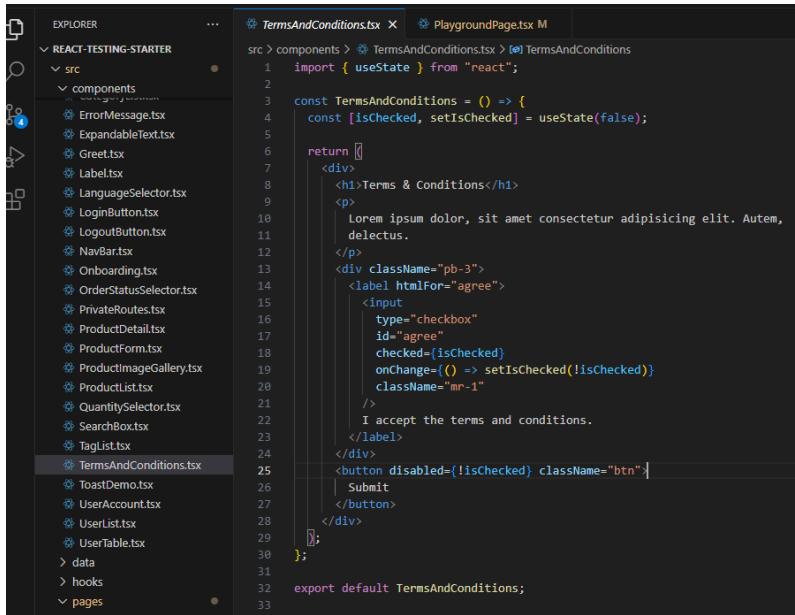
- Code Editor:** The active file is `ProductImageGallery.test.tsx`. The code contains a test for the `ProductImageGallery` component, specifically for "Test Case 02". It includes assertions for rendering a list of images with URLs `'url1'` and `'url2'`.
- Terminal:** The terminal shows the execution of the test. It outputs the message `PASSED as expected.`, followed by a summary of test results:
  - ✓ tests/components/ProductImageGallery.test.tsx (2)
  - ✓ ProductImageGallery (2)
  - ✓ should render nothing if returned with empty array
  - ✓ should render a list of images

Test Files 1 passed (1)  
Tests 2 passed (2)  
Start at 13:36:03  
Duration 465ms
- Status Bar:** The status bar indicates "PASS Waiting for file changes..." and provides keyboard shortcuts (`h` for help, `q` to quit).



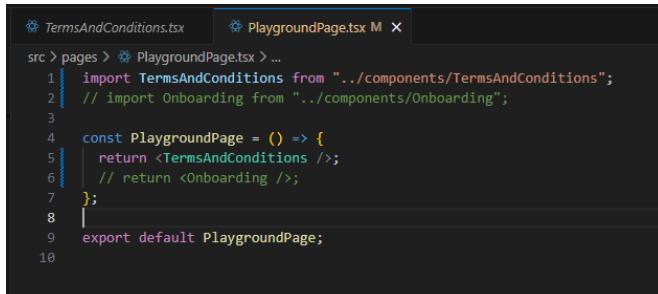
## 9. Testing User Interactions (TermsAndConditions)

### Component to test: TermsAndConditions.tsx



```
src > components > TermsAndConditions.tsx M
src > components > TermsAndConditions.tsx > TermsAndConditions
1 import { useState } from "react";
2
3 const TermsAndConditions = () => {
4   const [isChecked, setIsChecked] = useState(false);
5
6   return [
7     <div>
8       <h1>Terms & Conditions</h1>
9       <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Autem, delectus.</p>
10      <div className="pb-3">
11        <label htmlFor="agree">
12          <input type="checkbox" id="agree" checked={isChecked} onChange={() => setIsChecked(!isChecked)} className="mr-1" />
13          I accept the terms and conditions.
14        </label>
15      </div>
16      <button disabled={!isChecked} className="btn">
17        Submit
18      </button>
19    </div>
20  ];
21}
22
23 export default TermsAndConditions;
```

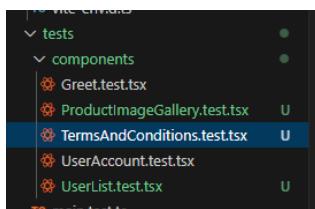
To display component in UI...



```
src > pages > PlaygroundPage.tsx M
src > pages > PlaygroundPage.tsx > ...
1 import TermsAndConditions from "../components/TermsAndConditions";
2 // import Onboarding from "../components/Onboarding";
3
4 const PlaygroundPage = () => {
5   return <TermsAndConditions />;
6   // return <Onboarding />;
7 };
8
9 export default PlaygroundPage;
10
```

### Test case 01:

Create a test file...



```
tests
  components
    Greet.test.tsx
    ProductImageGallery.test.tsx
    TermsAndConditions.test.tsx
    UserAccount.test.tsx
    UserList.test.tsx
```

EXPLORER

REACT-TESTING-STARTER

- src
  - components
  - data
  - hooks
  - pages
  - providers
  - store
  - validationSchemas
- # App.css
- App.tsx
- entities.ts
- environment.d.ts
- # index.css
- main.tsx
- routes.tsx
- vite-env.d.ts
- tests
  - components
  - Greet.test.tsx
  - ProductImageGallery.test.tsx
  - TermsAndConditions.test.tsx
  - UserAccount.test.tsx
  - UserList.test.tsx
- main.test.ts
- setup.ts
- .env
- .env.local
- .eslintrc.cjs
- .gitignore
- index.html
- json-server.json
- package-lock.json
- package.json

TERMINAL

```
tests > components > TermsAndConditions.test.tsx ✘ TermsAndConditions.tsx
  tests > components > TermsAndConditions.test.tsx > describe('TermsAndConditions') callback
    4   /* Test case 01 */
    5   /* Expected result is it should render with correct text and initial state */
    6   describe ('TermsAndConditions', () => {
    7     it('should render with correct text and initial state', () => {
    8       render(<TermsAndConditions/>);
    9       const heading = screen.getByRole('heading');
   10       expect(heading).toBeInTheDocument();
   11       expect(heading).toHaveTextContent ('Terms & Conditions');
   12       // validate the checkbox's initial state is unchecked
   13       const checkbox = screen.getByRole ('checkbox');
   14       expect(checkbox).toBeInTheDocument();
   15       expect(checkbox).not.toBeChecked();
   16       // validate the Submit button is initially disabled
   17       const button = screen.getByRole ('button');
   18       expect(button).toBeInTheDocument();
   19       expect(button).toHaveTextContent (/submit/i);
   20       expect(button).toBeDisabled();
   21       console.log('PASSED as expected.');
   22     })
   23   })

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

RERUN tests/components/TermsAndConditions.test.tsx x96

stdout | tests/components/TermsAndConditions.test.tsx > TermsAndConditions > should render with correct text and initial state  
PASSED as expected.

✓ tests/components/TermsAndConditions.test.tsx (1)  
  ✓ TermsAndConditions (1)  
    ✓ should render with correct text and initial state

Test Files 1 passed (1)  
  Tests 1 passed (1)  
  Start at 14:24:52  
  Duration 505ms

PASS Waiting for file changes...  
press h to show help, press q to quit

localhost:51204/\_vitest\_/#/

Vitest

Dashboard

Search...

Filter

Fail     Pass     Skip     Only Tests

FAIL (0) / RUNNING (0)  
PASS (6) / SKIP (-)

- > ✓ tests/components/Greet.test.tsx 166ms
- > ✓ tests/components/ProductImageGallery.test.tsx 145ms
- > ✓ tests/components/TermsAndConditions.test.tsx 79ms
  - ↳ ✓ TermsAndConditions 78ms
    - ↳ ✓ should render with correct text and initial state 77ms ←
- > ✓ tests/components/UserAccount.test.tsx 154ms
- > ✓ tests/components/UserList.test.tsx 173ms
- > ✓ tests/main.test.ts 4ms

11	0	11
Pass	Fail	Total

Files	6
Pass	6
Time	13.65s

## Test case 02:

First is to install the user-event library...

The screenshot shows the Testing Library documentation website with the 'Installation' page selected. The page provides instructions for installing the library using npm or Yarn, with a note that it requires @testing-library/dom. Below this, a terminal window shows the command `npm install --save-dev @testing-library/user-event` being run successfully.

Code editor view:

```
25  /* Test Case 02 */
26  /* Expected result is it should enable the button when the checkbox is checked */
27  // Note: install user-event library from @testing-library (this is to simulate a user behavior from the UI)
28  it('should enable the button when the checkbox is checked', async () => {
29    //Arrange
30    render(<TermsAndConditions/>);
31    //Act
32    const checkbox = screen.getByRole('checkbox');
33    const user = userEvent.setup();
34    await user.click(checkbox);
35    //Assert
36    expect(screen.getByRole('button')).toBeEnabled();
37    console.log('PASSED as expected.');
38  })
39 }
```

Terminal output:

```
stdout | tests/components/TermsAndConditions.test.tsx > TermsAndConditions > should enable the button when the checkbox is checked
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/ProductImageGallery.test.tsx (2)
✓ tests/components/TermsAndConditions.test.tsx (2)
✓ tests/components/UserAccount.test.tsx (3)
✓ tests/components/UserList.test.tsx (2)

Test Files 6 passed (6)
Tests 12 passed (12)
Start at 14:40:21
Duration 6.80s (transform 371ms, setup 2.19s, collect 2.52s, tests 990ms, environment 9.34s, prepare 1.70s)
```

PASS Waiting for file changes...  
press h to show help, press q to quit

The screenshot shows the Vitest test runner interface at [localhost:51204/\\_vitest\\_/#/](http://localhost:51204/_vitest_/#/). The interface has a dark theme. At the top, there are navigation icons and a search bar. Below that is a filter section with checkboxes for Fail, Pass, Skip, and Only Tests. The main area displays a tree view of test files and their results. A red arrow points to a green checkmark next to a test case in the `tests/components/TermsAndConditions.test.tsx` file. The summary at the bottom shows 12 Pass, 0 Fail, and a total time of 16.74s.

Pass	Fail	Total
12	0	12
Pass	Fail	Total

Files	6
Pass	6
Time	16.74s

## 10. Testing Expandable Test

**Component to test: ExpandableText.tsx**

The screenshot shows the VS Code Explorer with the project structure. The `src` folder contains several components like `AuthStatus.tsx`, `CancelOrderButton.tsx`, `CategoryList.tsx`, `ErrorMessage.tsx`, `ExpandableText.tsx` (which is highlighted), `Greet.tsx`, `Label.tsx`, `LanguageSelector.tsx`, `LoginButton.tsx`, `LogoutButton.tsx`, `NavBar.tsx`, `Onboarding.tsx`, `OrderStatusSelector.tsx`, and `PrivateRoute.tsx`. In the center, the code editor shows the `ExpandableText.tsx` file:

```

import { useState } from "react";
const ExpandableText = ({ text }: { text: string }) => {
  const limit = 255;
  const [isExpanded, setExpanded] = useState(false);

  if (text.length <= limit) return <article>{text}</article>

  return (
    <div>
      {isExpanded ? (
        <article>{text}</article>
      ) : (
        <article>{text.substring(0, limit)}...</article>
      )}
      <button onClick={() => setExpanded(!isExpanded)}>
        {isExpanded ? "Show Less" : "Show More"}
      </button>
    </div>
  );
};

export default ExpandableText;

```

## Test case 01:

Create a test file...

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure with `tests` and `components` subfolders. Inside `components`, there are files `ExpandableText.test.tsx` (selected) and `Greet.test.tsx`.
- Terminal:** Displays the test code for `ExpandableText.test.tsx`:

```
tests > components > ExpandableText.test.tsx > describe('ExpandableText') callback
1  /* Import functions and libraries */
2  import { render, screen } from '@testing-library/react';
3  import ExpandableText from '../../../../../src/components/ExpandableText';
4
5  /* Test Case 01 */
6  /* Expected result is it should render the full text if less than 25 char */
7  describe ('ExpandableText', () => {
8      it('should render the full text if less than 25 char', () => {
9          const text = "Short text";
10         render(<ExpandableText text={text}/>);
11         expect(screen.getByText(text)).toBeInTheDocument();
12         console.log('PASSED as expected.');
13     })
14 }
15 )
```

- Output:** Shows the terminal output of the test run:

```
PASSED as expected.

stdout | tests/components/ExpandableText.test.tsx > ExpandableText > should render the full text if less than 25 char
PASSED as expected.

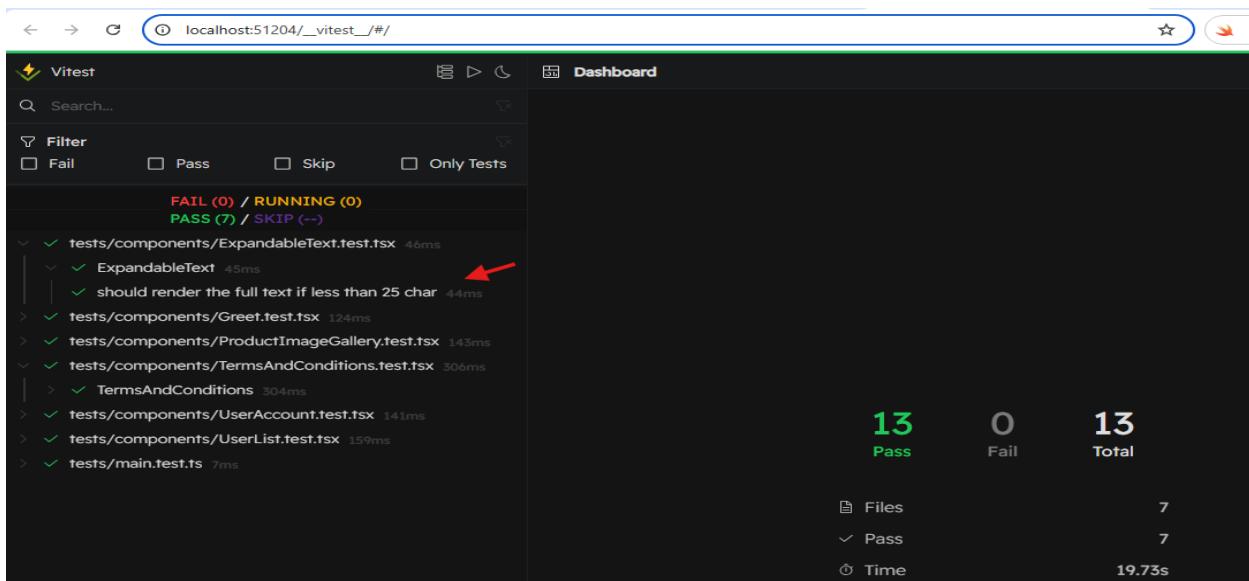
stdout | tests/components/Greet.test.tsx > Greet > should render Hello with the name when the name is provided
PASSED as expected.

stdout | tests/components/Greet.test.tsx > Greet > should render login button when a name is not provided
PASSED as expected.

✓ tests/main.test.ts (1)
✓ tests/components/ExpandableText.test.tsx (1)
✓ tests/components/Greet.test.tsx (2)
✓ tests/components/ProductImageGallery.test.tsx (2)
✓ tests/components/TermsAndConditions.test.tsx (2) 306ms
✓ tests/components/UserAccount.test.tsx (3)
✓ tests/components/UserList.test.tsx (2)

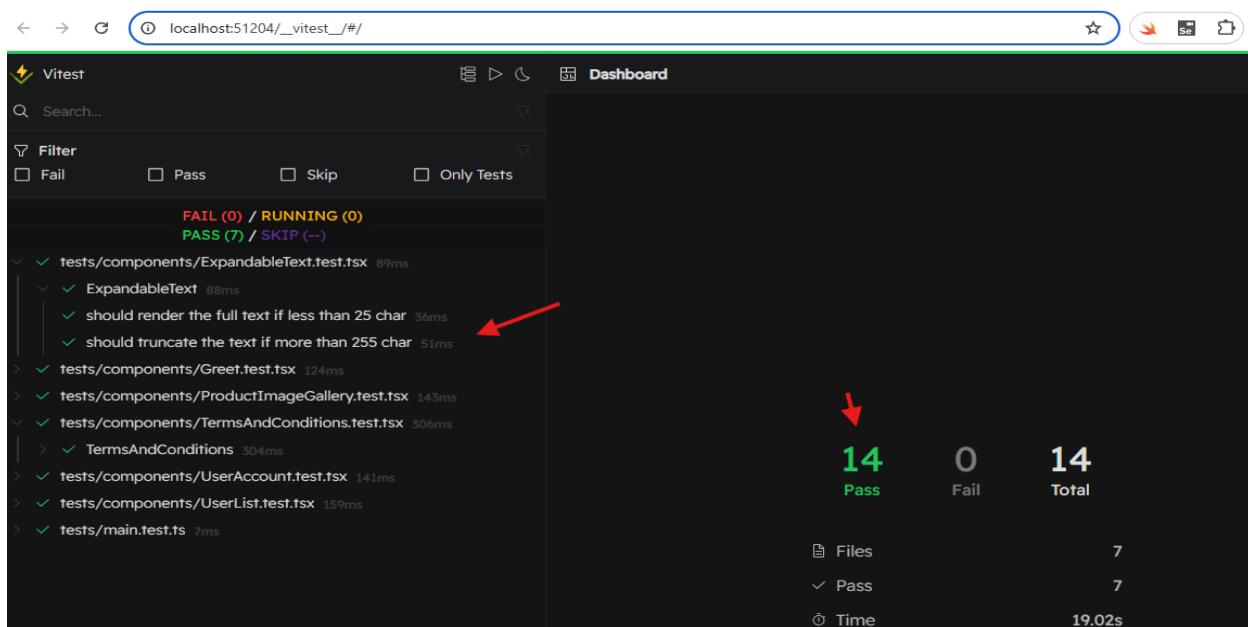
Test Files 7 passed (7)
Tests 13 passed (13)
Start at 15:32:41
Duration 7.07s (transform 378ms, setup 2.85s, collect 2.86s, tests 928ms, environment 10.88s, prepare 2.20s)
```

- Status Bar:** Shows "PASS Waiting for file changes..." and instructions to press **h** for help and **q** to quit.



## Test case 02:

```
ExpandableText.test.tsx M X
tests > components > ExpandableText.test.tsx > ...
1  /* Import functions and libraries */
2  import { render, screen } from '@testing-library/react';
3  import ExpandableText from '../../../../../src/components/ExpandableText';
4  // import { userEvent } from '@testing-library/user-event';
5
6  describe ('ExpandableText', () => {
7    const limit = 255;
8    const longText = 'a'.repeat (limit + 1);
9    const truncatedText = longText.substring (0, limit) + "...";
10
11   /* Test Case 01 */
12   /* Expected result is it should render the full text if less than 25 char */
13   it('should render the full text if less than 25 char', () => {
14     const text = "Short text";
15     render(<ExpandableText text={text}>/);
16     expect(screen.getByText(text)).toBeInTheDocument();
17     console.log('PASSED as expected.');
18   })
19
20   /* Test Case 02 */
21   /* Expected result is it should truncate the text if more than 255 char */
22   it('should truncate the text if more than 255 char', () => {
23     // this is to check the text if more than 255 char
24     render(<ExpandableText text={longText} />);
25     expect(screen.getByText(truncatedText)).toBeInTheDocument();
26     // this is to check the expected buttons
27     const button = screen.getByRole('button');
28     expect(button).toHaveTextContent(/more/i);
29     console.log('PASSED as expected.');
30   })
31
```



### Test case 03:

```
32  /* Test Case 03 */
33  /* Expected result is it should expand text when Show More button is clicked */
34  it('should expand text when Show More button is clicked', async () => [
35    // this is to check the text if more than 255 char...
36    render(<ExpandableText text={longText} />);
37    // this is to check when button is clicked...
38    const button = screen.getByRole('button');
39    const user = userEvent.setup();
40    await user.click(button);
41    // this is to check the expected buttons...
42    expect(screen.getByText(longText)).toBeInTheDocument();
43    | | | expect(button).toHaveTextContent(/less/i);
44    console.log('PASSED as expected.');
45  ]);
46 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

node + □ 🗑

```
stdout | tests/components/ExpandableText.test.tsx > ExpandableText > should expand text when Show More button is clicked
PASSED as expected.

✓ tests/components/ExpandableText.test.tsx (3)
  ✓ ExpandableText (3)
    ✓ should render the full text if less than 25 char
    ✓ should truncate the text if more than 255 char
    ✓ should expand text when Show More button is clicked

Test Files 1 passed (1)
Tests 3 passed (3)
Start at 16:44:32
Duration 667ms

PASS Waiting for file changes...
press h to show help, press q to quit
```

The screenshot shows the Vitest dashboard interface. At the top, there's a navigation bar with icons for back, forward, search, and refresh. The URL is localhost:51204/\_vitest\_/#/. Below the navigation is a search bar and a filter section with checkboxes for Fail, Pass, Skip, and Only Tests. The main area displays a tree view of test files and their results. A red arrow points to the 'should expand text when Show More button is clicked' test under 'ExpandableText'. To the right of the tree view is a summary statistics panel. A second red arrow points to the '15' in the 'Pass' column. The statistics are as follows:

Pass	Fail	Total
15	0	15

Below the statistics, there are three rows of metrics: Files (7), Pass (7), and Time (19.26s).

## Test case 04:

The screenshot shows a code editor with a test file named `ExpandableText.test.tsx`. The file contains a test suite for the `ExpandableText` component. A specific test case, labeled 'Test Case 04', is failing. The failure message is: `Expected result is it should collapse text when Show Less button is clicked`. The terminal output at the bottom shows the test results, indicating a failure for this specific test case.

```
tests > components > ExpandableText.test.tsx > describe('ExpandableText') callback > it('should collapse text when Show Less button is clicked') callback
  6   describe ('ExpandableText', () => {
47     /* Test Case 04 */
48     /* Expected result is it should collapse text when Show Less button is clicked */
49     it('should collapse text when Show Less button is clicked', async () => [
50       // this is to check the text if more than 255 char...
51       render(<ExpandableText text={longText} />);
52
53       // this is to check when button is clicked the first time (button is from Show More to Show Less)...
54       const showMoreButton = screen.getByRole('button', {name: /more/i});
55       const user = userEvent.setup();
56       await user.click(showMoreButton);
57       // this is to check when button is clicked the after full text is displayed and button is from Show Less to Show More again...
58       const showLessButton = screen.getByRole('button', {name: /less/i});
59       await user.click(showLessButton);
60
61       // this is to check the expected buttons...
62       expect(screen.getByText(truncatedText)).toBeInTheDocument();
63       expect(showMoreButton).toHaveTextContent(/more/i);
64       console.log('PASSED as expected.');
65     ])
66   })
```

PASSED as expected.

```
stdout | tests/components/ExpandableText.test.tsx > ExpandableText > should collapse text when Show Less button is clicked
PASSED as expected.
```

```
✓ tests/components/ExpandableText.test.tsx (4)
  ✓ ExpandableText (4)
    ✓ should render the full text if less than 25 char
    ✓ should truncate the text if more than 255 char
    ✓ should expand text when Show More button is clicked
    ✓ should collapse text when Show Less button is clicked

Test Files 1 passed (1)
Tests 4 passed (4)
Start at 16:54:56
Duration 761ms
```

PASS Waiting for file changes...  
press h to show help, press q to quit

The screenshot shows the Vitest UI interface. It displays the test results for the `ExpandableText` component. The test case 'should collapse text when Show Less button is clicked' is highlighted with a red arrow. The overall summary shows 16 Passes, 0 Failures, and 16 Total tests.

Vitest

localhost:51204/\_vitest\_/#/

Dashboard

FAIL (0) / RUNNING (0)  
PASS (7) / SKIP (-)

Filter

Pass Skip Only Tests

tests/components/ExpandableText.test.tsx 262ms

ExpandableText 260ms

should render the full text if less than 25 char 32ms  
should truncate the text if more than 255 char 43ms  
should expand text when Show More button is clicked 56ms  
should collapse text when Show Less button is clicked 129ms

tests/components/Greet.test.tsx 124ms

tests/components/ProductImageGallery.test.tsx 143ms

tests/components/TermsAndConditions.test.tsx 306ms

TermsAndConditions 304ms

tests/components/UserAccount.test.tsx 141ms

tests/components/UserList.test.tsx 159ms

tests/main.test.ts 7ms

16 Pass 0 Fail 16 Total

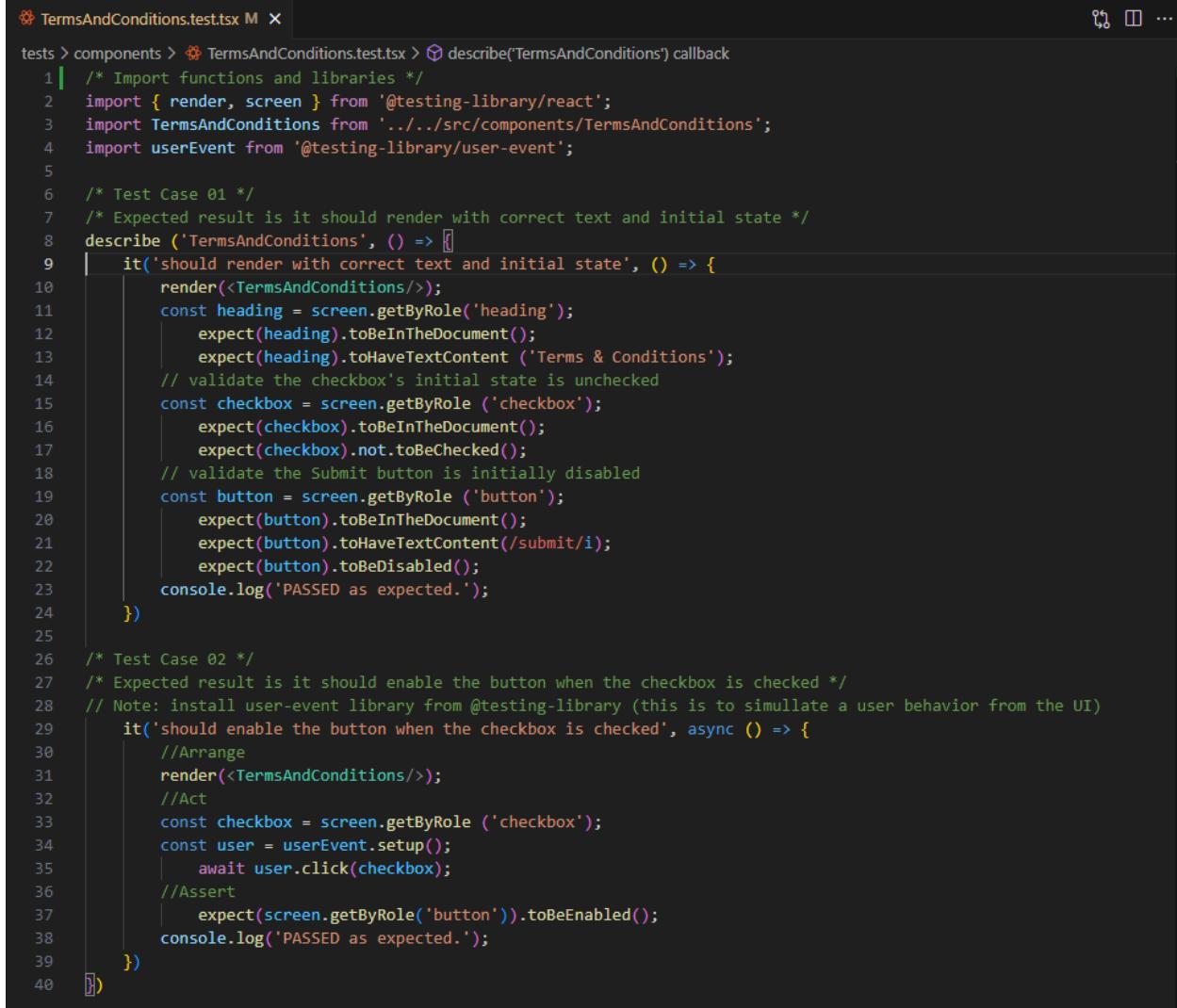
Files 7

Pass 7

Time 19.29s

## 11. Simplifying the Tests

Before cleanup...



```
tests > components > TermsAndConditions.test.tsx M X
tests > components > TermsAndConditions.test.tsx > describe('TermsAndConditions') callback
1 | /* Import functions and libraries */
2 | import { render, screen } from '@testing-library/react';
3 | import TermsAndConditions from '../../../../../src/components/TermsAndConditions';
4 | import userEvent from '@testing-library/user-event';
5 |
6 | /* Test Case 01 */
7 | /* Expected result is it should render with correct text and initial state */
8 | describe ('TermsAndConditions', () => [
9 |     it('should render with correct text and initial state', () => {
10 |         render(<TermsAndConditions/>);
11 |         const heading = screen.getByRole('heading');
12 |         expect(heading).toBeInTheDocument();
13 |         expect(heading).toHaveTextContent ('Terms & Conditions');
14 |         // validate the checkbox's initial state is unchecked
15 |         const checkbox = screen.getByRole ('checkbox');
16 |         expect(checkbox).toBeInTheDocument();
17 |         expect(checkbox).not.toBeChecked();
18 |         // validate the Submit button is initially disabled
19 |         const button = screen.getByRole ('button');
20 |         expect(button).toBeInTheDocument();
21 |         expect(button).toHaveTextContent(/submit/i);
22 |         expect(button).toBeDisabled();
23 |         console.log('PASSED as expected.');
24 |     })
25 |
26 |     /* Test Case 02 */
27 |     /* Expected result is it should enable the button when the checkbox is checked */
28 |     // Note: install user-event library from @testing-library (this is to simulate a user behavior from the UI)
29 |     it('should enable the button when the checkbox is checked', async () => {
30 |         //Arrange
31 |         render(<TermsAndConditions/>);
32 |         //Act
33 |         const checkbox = screen.getByRole ('checkbox');
34 |         const user = userEvent.setup();
35 |         await user.click(checkbox);
36 |         //Assert
37 |         expect(screen.getByRole('button')).toBeEnabled();
38 |         console.log('PASSED as expected.');
39 |     })
40 | ])
```

## After cleanup...

The screenshot shows a code editor in VS Code with a dark theme. The active file is `TermsAndConditions.test.tsx`. The code contains two test cases: `Test Case 01` and `Test Case 02`. `Test Case 01` checks the initial state of the component, including the heading text, checkbox status, and button enabledness. `Test Case 02` is an asynchronous test that checks if the button becomes enabled when the checkbox is checked. The terminal at the bottom shows a `PASS` message and a note about waiting for file changes.

```
tests > components > TermsAndConditions.test.tsx > ...
1  /* Import functions and libraries */
2  import { render, screen } from '@testing-library/react';
3  import TermsAndConditions from '../../../../../src/components/TermsAndConditions';
4  import userEvent from '@testing-library/user-event';
5
6  /* Test Case 01 */
7  /* Expected result is it should render with correct text and initial state */
8  describe ('TermsAndConditions', () => {
9      const renderComponent = () => {
10          render(<TermsAndConditions/>)
11          return{
12              heading: screen.getByRole('heading'),
13              checkbox: screen.getByRole ('checkbox'),
14              button: screen.getByRole ('button')
15          }
16      }
17      it('should render with correct text and initial state', () => {
18          const {heading, checkbox, button} = renderComponent();
19
20          expect(heading).toHaveTextContent ('Terms & Conditions');
21          // validate the checkbox's initial state is unchecked
22          expect(checkbox).not.toBeChecked();
23          // validate the Submit button is initially disabled
24          expect(button).toBeDisabled();
25          console.log('PASSED as expected.');
26      });
27
28  /* Test Case 02 */
29  /* Expected result is it should enable the button when the checkbox is checked */
30  // Note: install user-event library from @testing-library (this is to simulate a user behavior from the UI)
31  it('should enable the button when the checkbox is checked', async () => {
32      const {checkbox, button} = renderComponent();
33
34      //Arrange
35      //Act
36      const user = userEvent.setup();
37      await user.click(checkbox);
38      //Assert
39      expect(button).toBeEnabled();
40      console.log('PASSED as expected.');
41  })
42 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PASS Waiting for file changes...  
press h to show help, press q to quit

## 12. Final end result - all test cases PASSED

The screenshot shows the Vitest dashboard running at [localhost:51204/\\_vitest\\_/#/](http://localhost:51204/_vitest_/#/). The interface is dark-themed. On the left, a tree view lists test files and their results. The main summary on the right indicates 16 Pass, 0 Fail, and 16 Total tests. Below the summary, there are breakdowns by file, pass count, and total time.

Category	Count	Time
Files	7	
Pass	7	
Time	17.36s	

Test results from the tree view:

- tests/components/ExpandableText.test.tsx (262ms)
  - ✓ ExpandableText (260ms)
    - ✓ should render the full text if less than 25 char (52ms)
    - ✓ should truncate the text if more than 255 char (45ms)
    - ✓ should expand text when Show More button is clicked (56ms)
    - ✓ should collapse text when Show Less button is clicked (129ms)
- tests/components/Greet.test.tsx (124ms)
- tests/components/ProductImageGallery.test.tsx (143ms)
- tests/components/TermsAndConditions.test.tsx (187ms)
  - ✓ TermsAndConditions (186ms)
- tests/components/UserAccount.test.tsx (141ms)
- tests/components/UserList.test.tsx (159ms)
- tests/main.test.ts (7ms)

\*\*\*\*\*END\*\*\*\*\*