Writing Unit Tests in React with Vitest

1. Setup: Testing Environment

Before writing tests, make sure your environment is ready:

1. Open terminal in your project folder.
2. Install test dependencies:

```
npm install -D vitest jsdom @testing-library/react
@testing-library/jest-dom
```

3. Edit `vite.config.ts` and add test configuration:

```
/// <reference types="vitest" />
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  test: {
    globals: true,
    environment: 'jsdom'
  }
})
```

2. Writing Tests

All test files should be placed in the `src/tests/` folder. Here's how each test works:

Test 1: Header Component

File: `src/tests/Header.test.tsx`

Purpose: Ensure the `<Header />` displays the `title` prop correctly.

How to write it:

1. Import `render`, `describe`, `it`, and `expect`.
2. Render the component with a test prop.
3. Use `getByText()` to find it.
4. Assert that it exists.

```
import { render } from '@testing-library/react';
import { describe, it, expect } from 'vitest';
import Header from '../components/Header';

describe('Header', () => {
  it('renders the title', () => {
    const { getByText } = render(<Header title="Test Title"
/>);
    expect(getByText('Test Title')).toBeDefined();
  });
});
```

Test 2: Counter Component

File: `src/tests/Counter.test.tsx`

Purpose: Check that the button increments the count when clicked.

How to write it:

1. Import `fireEvent` to simulate a click.
2. Render `<Counter />`.
3. Click the "Increment" button.
4. Expect count to change.

```
import { render, fireEvent } from '@testing-library/react';
import { describe, it, expect } from 'vitest';
import Counter from '../components/Counter';

describe('Counter', () => {
  it('increments the count', () => {
```

```
    const { getByText } = render(<Counter />);
    const button = getByText('Increment');
    fireEvent.click(button);
    expect(getByText('Count: 1')).toBeDefined();
  });
});
```

Test 3: TodoList Component

File: `src/tests/TodoList.test.tsx`

Purpose: Ensure that typing into the input and clicking "Add" shows the new item.

How to write it:

1. Get the input using `getByPlaceholderText()`.
2. Type text into the input field.
3. Click the “Add” button.
4. Check that the new todo item appears.

```
import { render, fireEvent } from '@testing-library/react';
import { describe, it, expect } from 'vitest';
import TodoList from '../components/TodoList';

describe('TodoList', () => {
  it('adds a todo item', () => {
    const { getByPlaceholderText, getByText } =
render(<TodoList />);
    const input = getByPlaceholderText('Add a todo') as
HTMLInputElement;
    const button = getByText('Add');

    fireEvent.change(input, { target: { value: 'New Todo' }
});
    fireEvent.click(button);

    expect(getByText('New Todo')).toBeDefined();
  });
```

```
});
```

Run the Tests

Once all tests are written:

```
npx vitest run       # Runs all tests once
npx vitest           # Starts Vitest in watch mode
```

We see output such as:

```
✓ Header renders the title
✓ Counter increments the count
✓ TodoList adds a todo item
```