React App Testing with Vite + TypeScript

Agenda

- Why test React apps?
- Project setup using Vite
- Installing testing libraries
- Running tests
- package.json demo
- Small component & test example
- Summary

Why Testing Matters

- Prevent regressions
- Improve code confidence
- Safe refactoring
- Catch bugs early
- Required for production apps & CI/CD

Project Setup

Create React + TS App

- npm create vite@latest react-test-demo -- --template react-ts
- cd react-test-demo
- npm install
- npm run dev

Opens at http://localhost:5173

Default React + TS structure created

Testing Tools Used

Vitest

- Test runner by Vite team
- Jest-like syntax
- Super fast

React Testing Library

- Renders components
- Queries DOM like a user
- Encourages best practices

Install Testing Packages

npm install -D vitest @testing-library/react

npm install -D @testing-library/jest-dom jsdom

npm install -D @testing-library/user-event jsdom

- vitest → test runner
- jest-dom → extra matchers
- jsdom → browser-like environment

Configure Vite

vite.config.ts

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
export default defineConfig({
  plugins: [react()],
  test: {
    globals: true,
    environment: 'jsdom',
    setupFiles: './src/setupTests.ts',
  },
})
```

Test Setup File

src/setupTests.ts

import '@testing-library/jest-dom'

Adds matchers like:

- toBeInTheDocument()
- toHaveTextContent()

Understanding the Test Syntax

- **describe()** - The describe function groups related tests together. It takes a description string and a callback function containing the tests. You can nest describe blocks for better organization.
- **it() / test()** - The it function (or test, they're identical) defines a single test case. The description should clearly state what behavior is being tested. Write it as "it should..." for clarity.
- **expect()** - The expect function creates an assertion. It takes a value and returns an object with matcher methods like toBe(), toEqual(), toBeTruthy(), etc.

How to Run Tests

npm run test

npm run test:ui

npm run coverage

- Watch mode enabled
- UI dashboard available
- Coverage report generated

Sample Component

Greeting.tsx

```
type Props = { name: string };


export const Greeting = ({ name }: Props) => {
  return <h1>Hello, {name}</h1>;
};
```

Sample Test File

Greeting.test.tsx

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

import { describe, it, expect } from 'vitest';

import { Greeting } from '../Greeting';
```

Sample Test File

```
describe('Greeting Component', () => {
  it('renders name correctly', () => {
    render(<Greeting name="User" />);
    expect(
      screen.getByText('Hello, User')
    ).toBeInTheDocument();
  });
});
```

React Testing

- In **React Testing Library** (or DOM Testing Library), getByText can accept:
- **A string** — exact match.
- **A regular expression** — pattern match.
- **A function** — custom match logic.
- Using a regex with the i flag makes the test **more flexible** and avoids failures due to case differences in the rendered text.

expect(screen.getByText(/hello world/i)).toBeInTheDocument();

Summary

- Vite + TS = fast setup
- Vitest works out-of-box
- React Testing Library is user-centric
- Ideal for capstone demos
- Production-ready testing approach