Project Testing phase

Tools used for Testing:

➢ **React testing libraries**

🧪 Common React Testing Libraries & Their Uses

Library	Purpose	Why It's Used
Jest	JavaScript testing framework	Built-in with Create React App, supports mocking, assertions, and snapshot tests
React Testing Library	Tests components from the *user's perspective*	Encourages best practices by focusing on UI behavior, not internal logic
Enzyme	Utility for testing React components (by Airbnb)	Allows shallow rendering and simulating events (less used now in favor of RTL)
Cypress	End-to-end testing framework	Tests full user flows in a real browser
Storybook	UI component explorer	Lets you build and test components in isolation visually

Why use these tools?

- Are well-supported and actively maintained

- Integrate smoothly with React and modern dev workflows

- Help catch bugs early and improve code quality

- Simulate real user behavior, not just internal logic

➢ **Npm testing libraries**

🧪 **Popular npm Testing Libraries**

Library	Purpose	Why It's Used
Jest	A full-featured testing framework	Supports unit, integration, and snapshot testing; great for React and Node.js
Mocha	Flexible test runner	Works well with other libraries like Chai for assertions
Chai	Assertion library	Lets you write readable test expressions like `expect(value).to.equal⑤`
React Testing Library	Tests React components from the user's perspective	Encourages testing behavior over implementation details
@testing-library/jest-dom	Adds custom matchers for DOM assertions in Jest	Makes tests more expressive and readable (e.g., `toBeInTheDocument()`)

Why use *these* tools?

- Are **well-documented** and actively maintained

- Integrate easily with modern JavaScript frameworks

- Help catch bugs early and improve confidence in your code

- Support **different levels of testing**: unit, integration, and end-to-end