



Experiment 5

Student Name: Tejinderpal Singh

Branch: CSE

Semester: 6th

Subject Name: Full Stack Development – II

UID: 23BCS80343

Section/Group: KRG 3-B

Date of Performance: 17/02/2026

Subject Code: 23CSH-309

1. Aim: To verify the correctness and reliability of the EcoTrack React application by writing automated tests using Jest and React Testing Library, and by analyzing application behavior using debugging tools.

2. Objective:

- Understand the purpose of automated testing in frontend applications
- Write unit tests for JavaScript utility functions using Jest
- Use different Jest matchers to validate expected outputs and behaviors
- Test React components using React Testing Library
- Verify UI rendering by querying elements from the DOM
- Implement asynchronous testing using `findBy` and `waitFor` methods
- Apply mocking to simulate API or external data responses in tests
- Perform snapshot testing to detect unintended UI changes
- Debug failing tests and application logic using browser Developer Tools and breakpoints
- Analyze application behavior and errors systematically rather than manual checking

3. Implementation / Code:

▪ Tools & Technologies Used:-

- React.js
- JavaScript (ES6)
- Jest Testing Framework
- React Testing Library
- VS Code
- Node.js & npm
- Web Browser (Chrome DevTools)

▪ Implementation Description:-

- The EcoTrack application is tested to ensure correctness of both logic and UI behavior.
- Unit testing is performed on utility functions (e.g., calculator function) using Jest.
- React Testing Library is used to render components and verify UI structure.
- Snapshot testing is applied to detect unintended UI changes over time.

- Automated tests improve application reliability and maintainability.
- Debugging tools such as browser DevTools and breakpoints help identify errors in logic or rendering.

▪ Sample Code Snippet:-

```

1  import { render, screen, fireEvent } from "@testing-library/react";
2  import App from "./App";
3
4  test("renders counter heading", () => {
5    render(<App />);
6    const heading = screen.getByText("Counter App");
7    expect(heading).toBeInTheDocument();
8  });
9
10 test("increments (alias) const screen: Screen<typeof import('d:/Full Stack 2/ceact-esting-
11   render(<App />) + app/node_modules/@testing-library/dom/types/queries">
12   - import screen
13   const button = screen.getByText("Increment");
14   const count = screen.getByTestId("count-value");
15
16   fireEvent.click(button);
17
18   expect(count).toHaveTextContent("1");
19 });

```

```

1  import { render, screen } from "@testing-library/react";
2  import Button from "./Button";
3
4  test("renders button text", () => {
5    render(<Button label="Submit" />);
6    const buttonElement = screen.getByText("Submit");
7    expect(buttonElement).toBeInTheDocument();
8  });

```

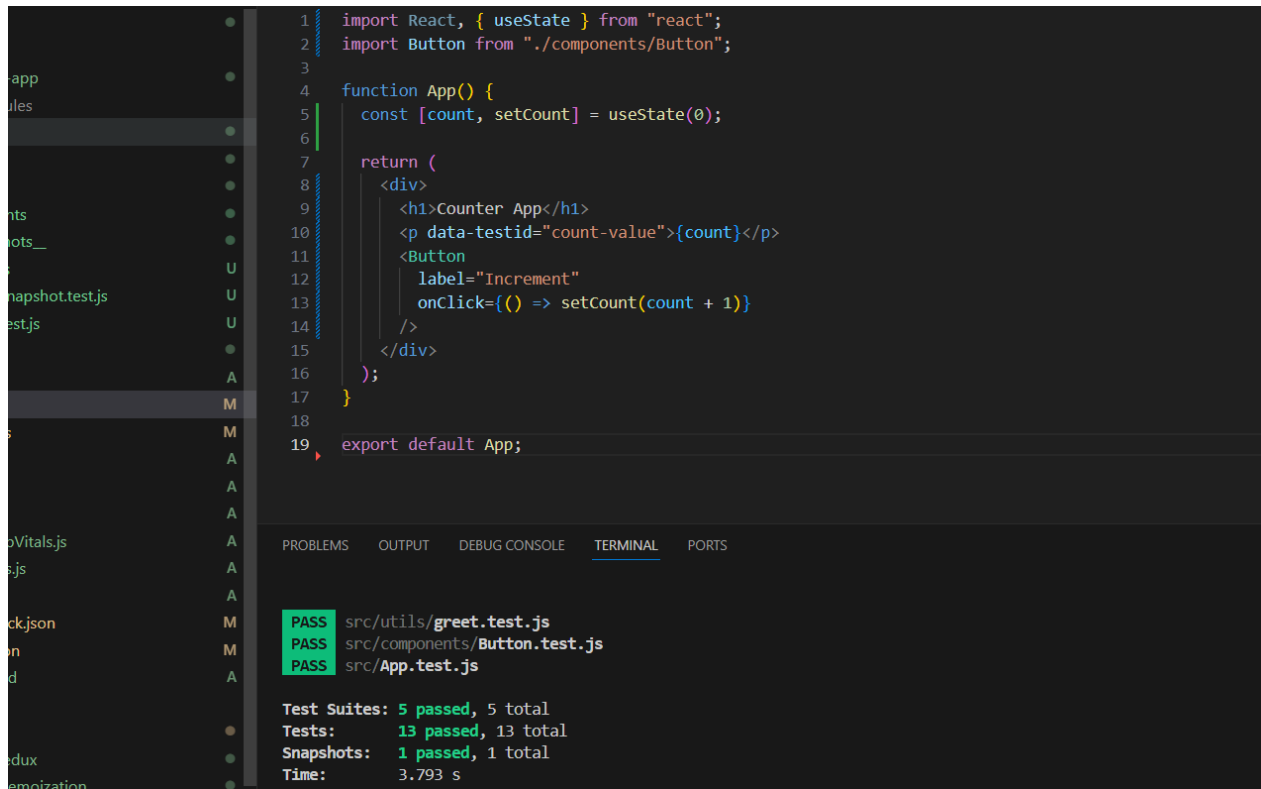
```

ceact-esting-app > src > components > JS Button.snapshot.test.js > ...
1  import renderer from "react-test-renderer";
2  import Button from "./Button";
3
4  test("snapshot test", () => {
5    const tree = renderer.create(<Button label="Submit" />).toJSON();
6    expect(tree).toMatchSnapshot();
7  });

```

4. Output:

- All Jest test cases executed successfully
- Utility function test passed
- React component snapshot test passed
- No unintended UI changes detected
- EcoTrack component rendered correctly during testing
- Debugging tools confirmed correct state updates and DOM rendering



```
1 import React, { useState } from "react";
2 import Button from "../components/Button";
3
4 function App() {
5   const [count, setCount] = useState(0);
6
7   return (
8     <div>
9       <h1>Counter App</h1>
10      <p data-testid="count-value">{count}</p>
11      <Button
12        label="Increment"
13        onClick={() => setCount(count + 1)}
14      />
15    </div>
16  );
17 }
18
19 export default App;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PASS src/utils/greet.test.js
PASS src/components/Button.test.js
PASS src/App.test.js

Test Suites: 5 passed, 5 total
Tests: 13 passed, 13 total
Snapshots: 1 passed, 1 total
Time: 3.793 s
```

5. Learning Outcomes (What I Have Learnt):

- Importance of automated testing in frontend applications
- Writing unit tests using Jest framework
- Using matchers like `toBe()` and `toMatchSnapshot()`
- Testing React components with React Testing Library
- Validating UI rendering through DOM queries
- Understanding snapshot testing for UI stability
- Debugging React applications using DevTools and breakpoints
- Improving software reliability and maintainability through testing