



## Experiment 5

**Student Name:** Raj Kumar Singh

**UID:** 23BCS11393

**Branch:** CSE

**Section/Group:** KRG 3-B

**Semester:** 6th

**Date of Performance:** 17/02/2026

**Subject Name:** Full Stack Development – II

**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.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH  
UNIVERSITY

Discover. Learn. Empower.

- 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:-

```
JS Tracker.test.js ×

src > components > JS Tracker.test.js > ...
1  // import { render, screen } from "@testing-library/react";
2  // import Tracker from "./Tracker";
3
4  // test("loads async data", async () => {
5  //   render(<Tracker />);
6
7  //   const text = await screen.findByText(/Eco data loaded/i, {}, { timeout: 3000 });
8
9  //   expect(text).toBeInTheDocument();
10 // });
11
12 import { render } from "@testing-library/react";
13 import Tracker from "./Tracker";
14
15 test("matches snapshot", () => {
16   const { asFragment } = render(<Tracker />);
17   expect(asFragment()).toMatchSnapshot();
18 })
```

JS calc.test.js ×

src > utils > JS calc.test.js > ...

```
1  import { add } from "./calc";
2
3  test("adds two numbers", () => {
4    |  expect(add(2, 3)).toBe(5);
5  })|
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH  
UNIVERSITY

Discover. Learn. Empower.

## 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

The screenshot shows the Visual Studio Code interface. The left sidebar (EXPLORER) displays the project structure for 'ECOTRACK'. It includes a folder for 'node\_modules', a 'public' folder, a 'src' folder containing 'components' (with a '\_\_snapshots\_\_' folder), 'tracker.css', 'Tracker.js', and 'Tracker.test.js', and a 'utils' folder containing 'calc.js', 'calc.test.js', 'App.css', 'App.js', 'index.css', 'index.js', 'logo.svg', 'reportWebVitals.js', 'setupTests.js', '.gitignore', 'package-lock.json', and 'package.json'. The 'utils' folder is currently selected. The right pane (TERMINAL) shows the output of a Jest test run. It displays two green 'PASS' messages: one for 'src/utils/calc.test.js' and another for 'src/components/Tracker.test.js'. Below these, it provides summary statistics: 'Test Suites: 2 passed, 2 total', 'Tests: 2 passed, 2 total', 'Snapshots: 1 passed, 1 total', and 'Time: 1.105 s'. It also indicates that all test suites related to changed files were ran. At the bottom of the terminal, there is a 'Watch Usage' section with instructions for navigating through the test results.

```
PASS  src/utils/calc.test.js
PASS  src/components/Tracker.test.js

Test Suites: 2 passed, 2 total
Tests:    2 passed, 2 total
Snapshots: 1 passed, 1 total
Time:    1.105 s

Ran all test suites related to changed files.

Watch Usage
> Press a to run all tests.
> Press f to run only failed tests.
> Press q to quit watch mode.
> Press p to filter by a filename regex pattern.
> Press t to filter by a test name regex pattern.
> Press Enter to trigger a test run.
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH  
UNIVERSITY

Discover. Learn. Empower.

## 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