

## Experiment 5

**Student Name:** Manasvi Sharma

**Branch:** BE CSE

**Semester:** 6<sup>th</sup>

**Subject Name:** Full Stack Development

**UID:** 23BCS11815

**Section/Group:** KRG 3A

**Date of Performance:** 17/02/26

**Subject Code:** 23CSH-309

### 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.

### Objective:

After completing this experiment, the student will be able to:

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

### Implementation/Code:

#### EmissionResult.test.jsx:

```
import { render, screen } from "@testing-library/react";
import EmissionResult from "../../components/EmissionResult";

test("renders emission value", () => {
  render(<EmissionResult emission={50} />);
  expect(screen.getByText(/50 kg CO2/i)).toBeInTheDocument();
});
```

## calculateCarbon.test.js:

```
import { calculateCarbon } from "../utils/calculateCarbon";
```

```
test("calculates carbon emission correctly", () => {  
  expect(calculateCarbon(10, 2)).toBe(20);  
});
```

```
test("returns 0 for negative values", () => {  
  expect(calculateCarbon(-5, 2)).toBe(0);  
});
```

## Dashboard.test.js:

```
import { render, screen } from "@testing-library/react";  
import Dashboard from "../components/Dashboard";
```

```
test("loads eco data asynchronously", async () => {  
  render(<Dashboard />);  
  expect(screen.getByText("Loading...")).toBeInTheDocument();
```

```
  const loadedText = await screen.findByText("Eco Data Loaded");  
  expect(loadedText).toBeInTheDocument();  
});
```

## Output:

```
PS C:\Users\My\FS_6th sem\testing\testing> npm test  
  
> testing@0.1.0 test  
> react-scripts test  
  
PASS src/testing/components/EmissionResult.test.jsx  
PASS src/testing/unit/calculateCarbon.test.js  
PASS src/testing/integration/Dashboard.test.js  
PASS src/App.test.js  
  
Test Suites: 4 passed, 4 total  
Tests: 5 passed, 5 total  
Snapshots: 0 total  
Time: 6.395 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.  
█
```

### **Learning Outcome:**

- Learned how to set up and use Jest and React Testing Library in a React application.
- Gained the ability to write unit tests to verify JavaScript utility functions.
- Understood how to test React components and validate UI rendering through DOM queries.
- Acquired skills in testing asynchronous behavior and mocking API calls in frontend applications.
- Developed the ability to debug failing tests and analyze application behavior systematically using automated testing tools.