Experiment-5

Student Name: Rashid Khan	**UID: 23BCS12551**
Branch: BE CSE	**Section/Group: KRG_3B**
Semester: 6th	**Date of Performance: 17/02/26**
Subject Name: Full Stack - II	**Subject Code: 23CSH-309**

1. Aim

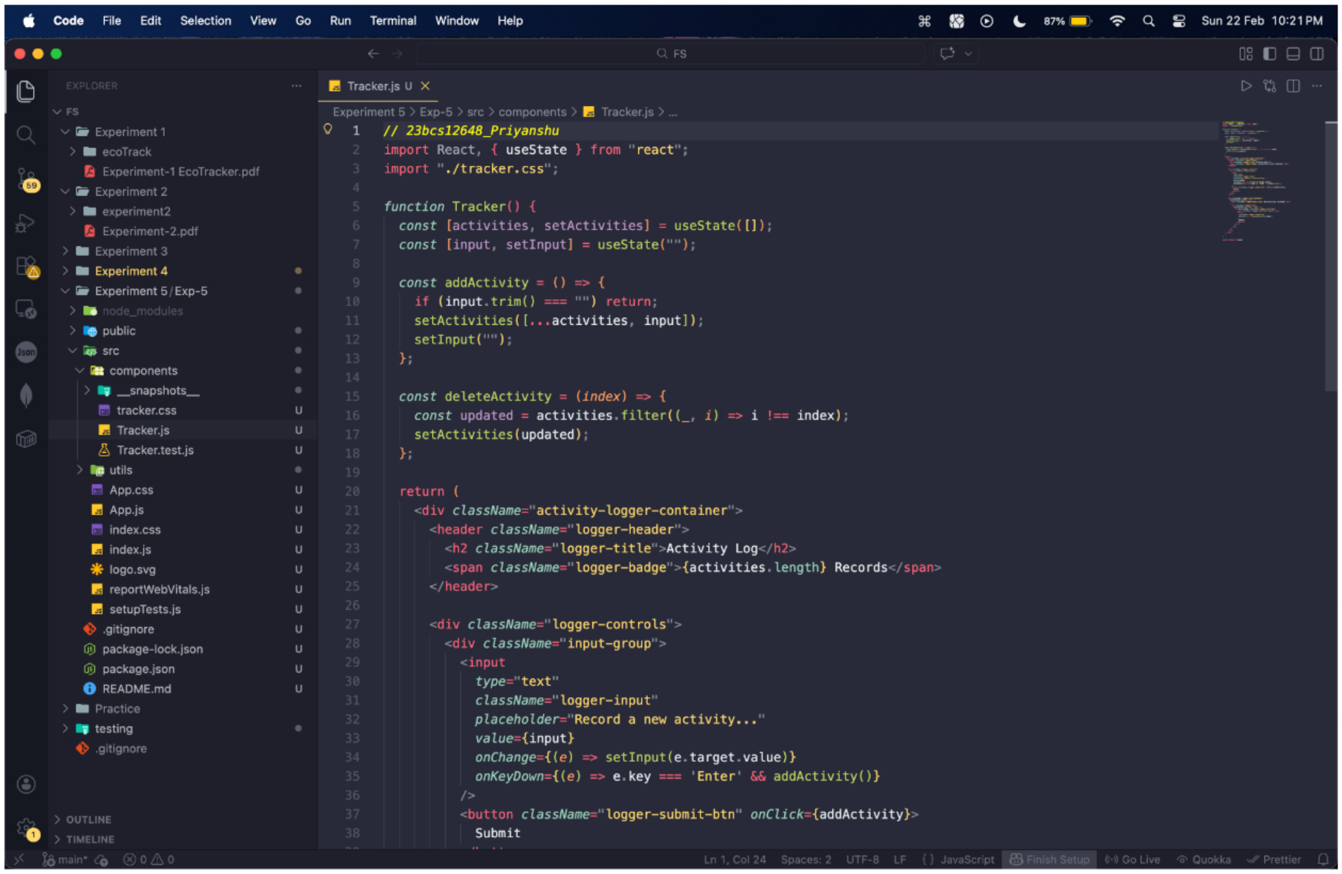
The primary goal of this experiment is to validate the stability and accuracy of the EcoTrack React application. This is achieved by implementing automated testing protocols using React Testing Library and Jest, alongside utilizing debugging utilities to scrutinize application performance.

2. Objectives

- To comprehend the role of automated testing within frontend development.
- To construct unit tests targeting JavaScript utility functions via the Jest framework.
- To apply various Jest matchers for the validation of anticipated application behaviors and outputs.
- To execute component testing utilizing the React Testing Library.
- To confirm accurate UI rendering through DOM element querying.
- To handle asynchronous test scenarios employing `waitFor` and `findBy` methodologies.
- To integrate mock functions for simulating external data and API responses during testing.
- To utilize snapshot testing as a mechanism for identifying inadvertent interface modifications.
- To troubleshoot application logic and failing tests using breakpoints and browser Developer Tools.

- To transition from manual verification to the systematic analysis of errors and application behavior.

3. Implementation and Methodology



```
// 23bcs12648_Priyanshu
import React, { useState } from "react";
import "./tracker.css";

function Tracker() {
  const [activities, setActivities] = useState([]);
  const [input, setInput] = useState("");

  const addActivity = () => {
    if (input.trim() === "") return;
    setActivities([...activities, input]);
    setInput("");
  };

  const deleteActivity = (index) => {
    const updated = activities.filter((_, i) => i !== index);
    setActivities(updated);
  };

  return (
    <div className="activity-logger-container">
      <header className="logger-header">
        <h2 className="logger-title">Activity Log</h2>
        <span className="logger-badge">{activities.length} Records</span>
      </header>

      <div className="logger-controls">
        <div className="input-group">
          <input
            type="text"
            className="logger-input"
            placeholder="Record a new activity..."
            value={input}
            onChange={(e) => setInput(e.target.value)}
            onKeyDown={(e) => e.key === 'Enter' && addActivity()}
          />
          <button className="logger-submit-btn" onClick={addActivity}>
            Submit
```

FIG: Code for the Tracker.js File

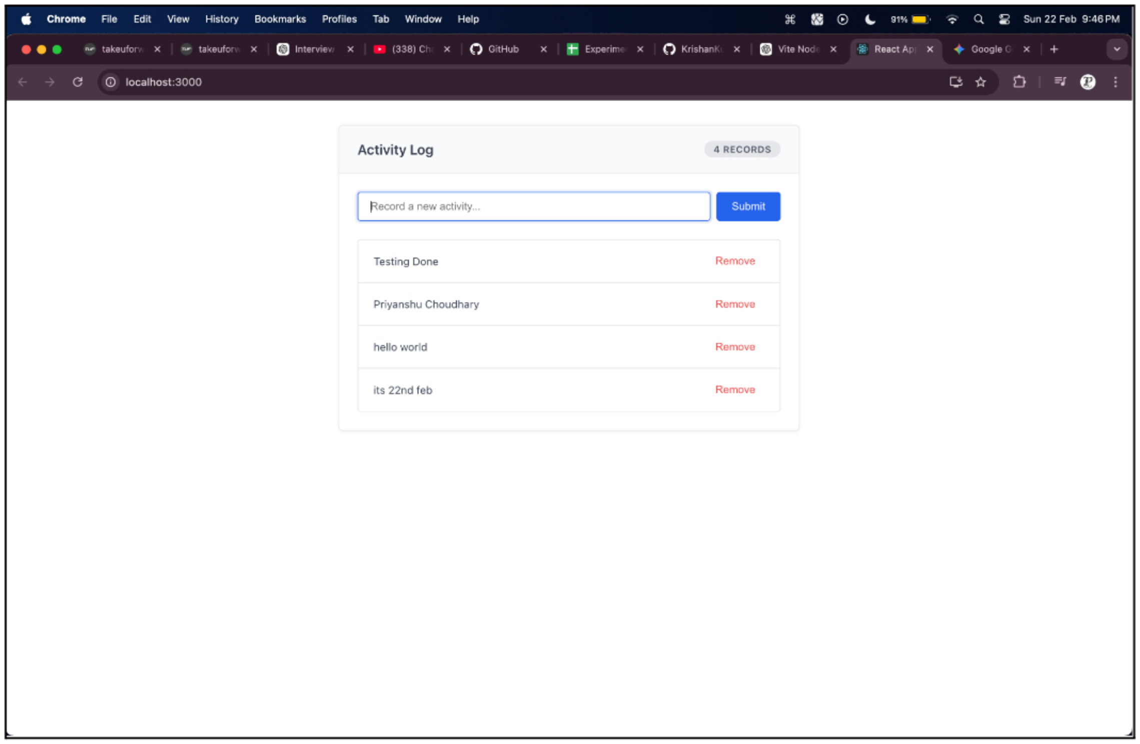
Tools & Technologies Utilized:

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

Implementation Description: The testing phase of the EcoTrack application guarantees the accuracy of both the user interface and underlying logic. Jest is utilized to perform unit tests on distinct utility functions, such as calculations. Furthermore, the React Testing Library ensures that UI components render with the correct structural hierarchy. To monitor UI consistency over time and catch unintended alterations, snapshot testing is implemented. Through these automated test suites, the overall maintainability and reliability of the software are significantly enhanced. Additionally, browser DevTools and strategic breakpoints are used to track down rendering or logical defects.

4. Output Summary

- The execution of all Jest test suites was successful.
- Tests targeting utility functions passed without errors.
- The React component snapshot evaluations were completed successfully.
- No unintended modifications were detected in the user interface.
- The EcoTrack interface rendered properly throughout the testing phase.
- Verification via debugging tools demonstrated accurate DOM rendering and state management.



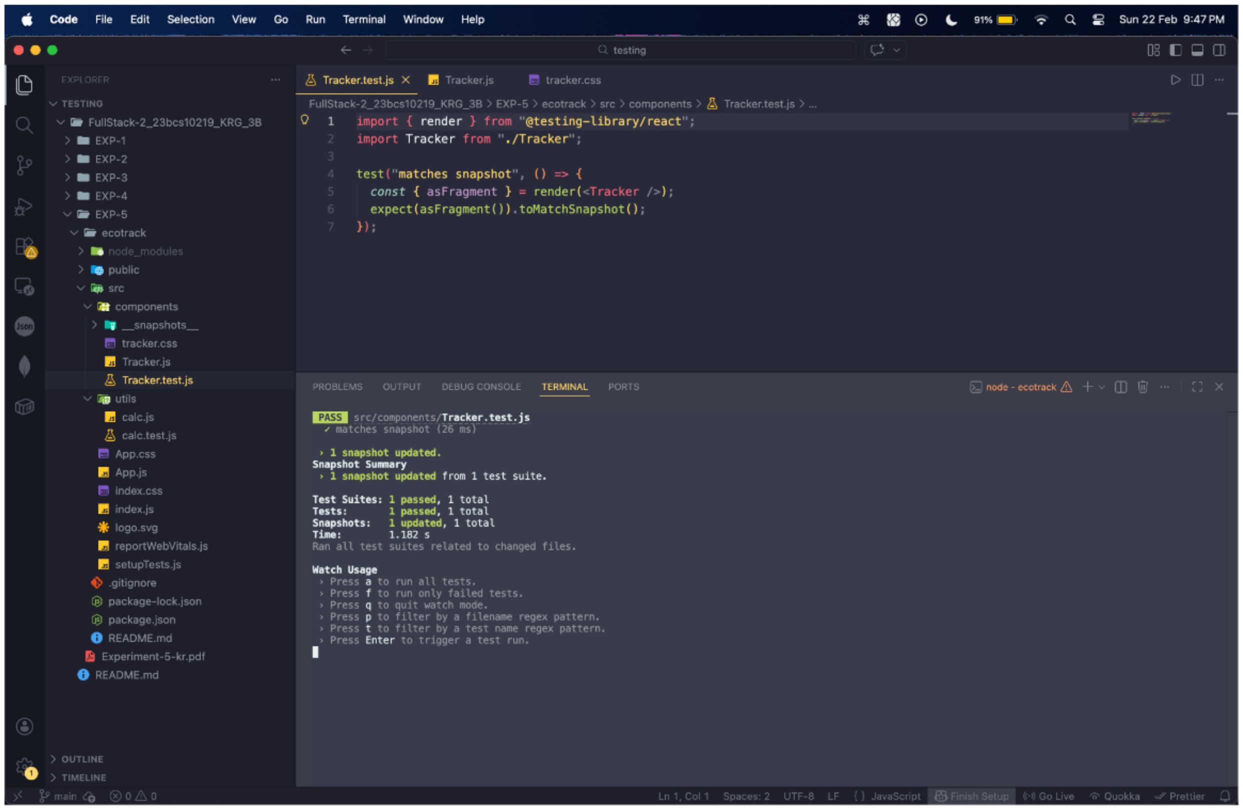
Testing Done

Priyanshu Choudhary

hello world

its 22nd feb

FIG: DashBoard UI



```
import { render } from "@testing-library/react";
import Tracker from "./Tracker";

test("matches snapshot", () => {
  const { asFragment } = render(<Tracker />);
  expect(asFragment()).toMatchSnapshot();
});
```

```
PASS  src/components/Tracker.test.js
  ✓ matches snapshot (26 ms)

 › 1 snapshot updated.
Snapshot Summary
 › 1 snapshot updated from 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 updated, 1 total
Time:        1.182 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.
```

FIG: Testing Status(All test cases passes)

5. Learning Outcomes

- Gained insight into the critical role of automated testing for frontend systems.
- Acquired practical experience drafting unit tests with the Jest framework.
- Developed proficiency in using specific matchers, including `toBe()` and `toMatchSnapshot()`.
- Learned to effectively test React elements utilizing the React Testing Library.
- Understood how to validate UI output by leveraging DOM queries.
- Recognized the value of snapshot testing in maintaining stable user interfaces.
- Enhanced skills in diagnosing React applications utilizing breakpoints and DevTools.
- Acknowledged how comprehensive testing strategies directly improve software maintainability and reliability.