Experiment 5

Student Name: Ashutosh Yadav **UID: 23BCS11023**
Branch: CSE **Section/Group: KRG-3-A**
Semester: 6th **Date of Performance:16/02/2026**
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:

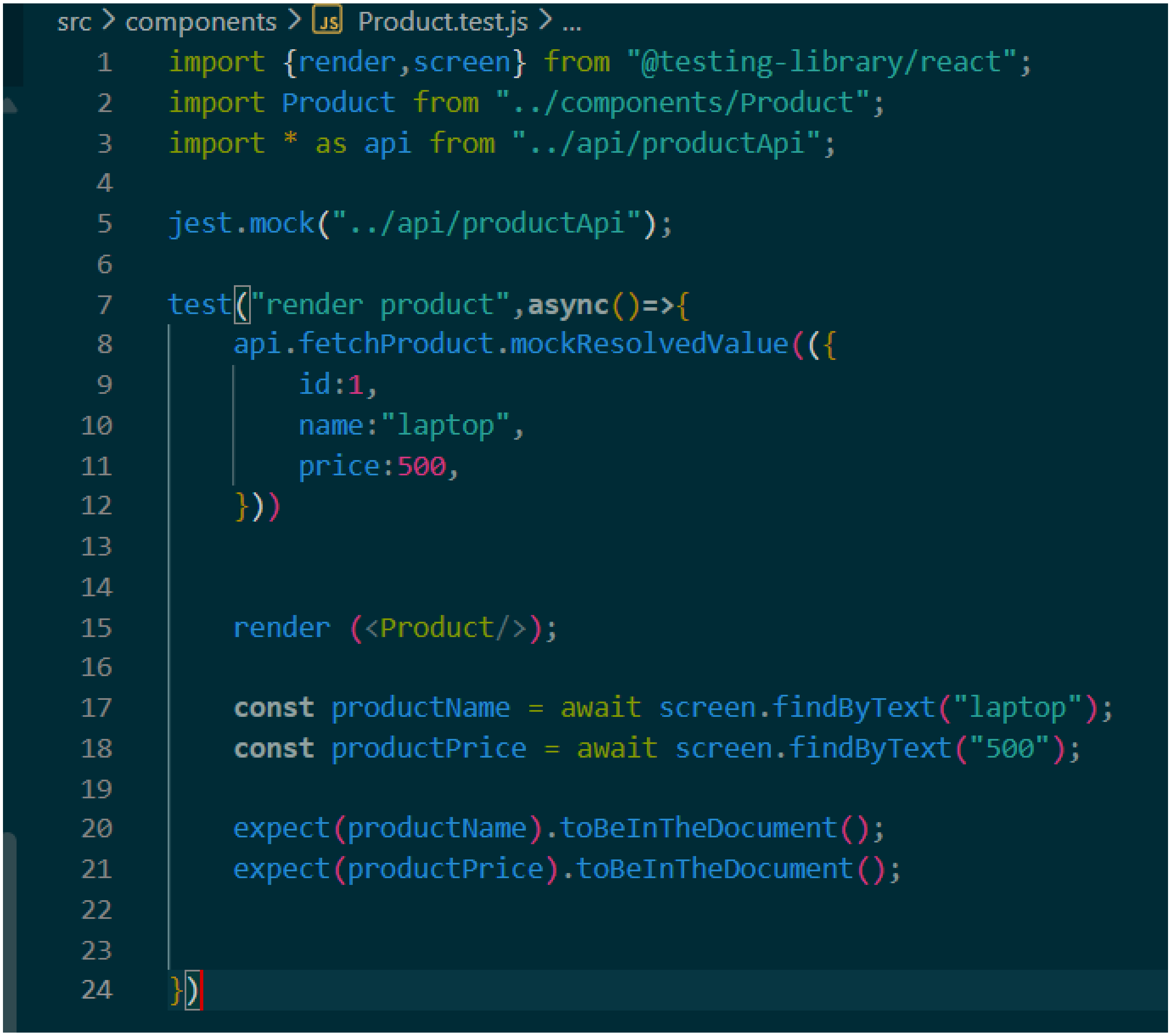
1. Understand the purpose of automated testing in frontend applications
2. Write unit tests for JavaScript utility functions using Jest
3. Use different Jest matchers to validate expected outputs and behaviors
4. Test React components using React Testing Library
5. Verify UI rendering by querying elements 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

3. Implementation / Code:

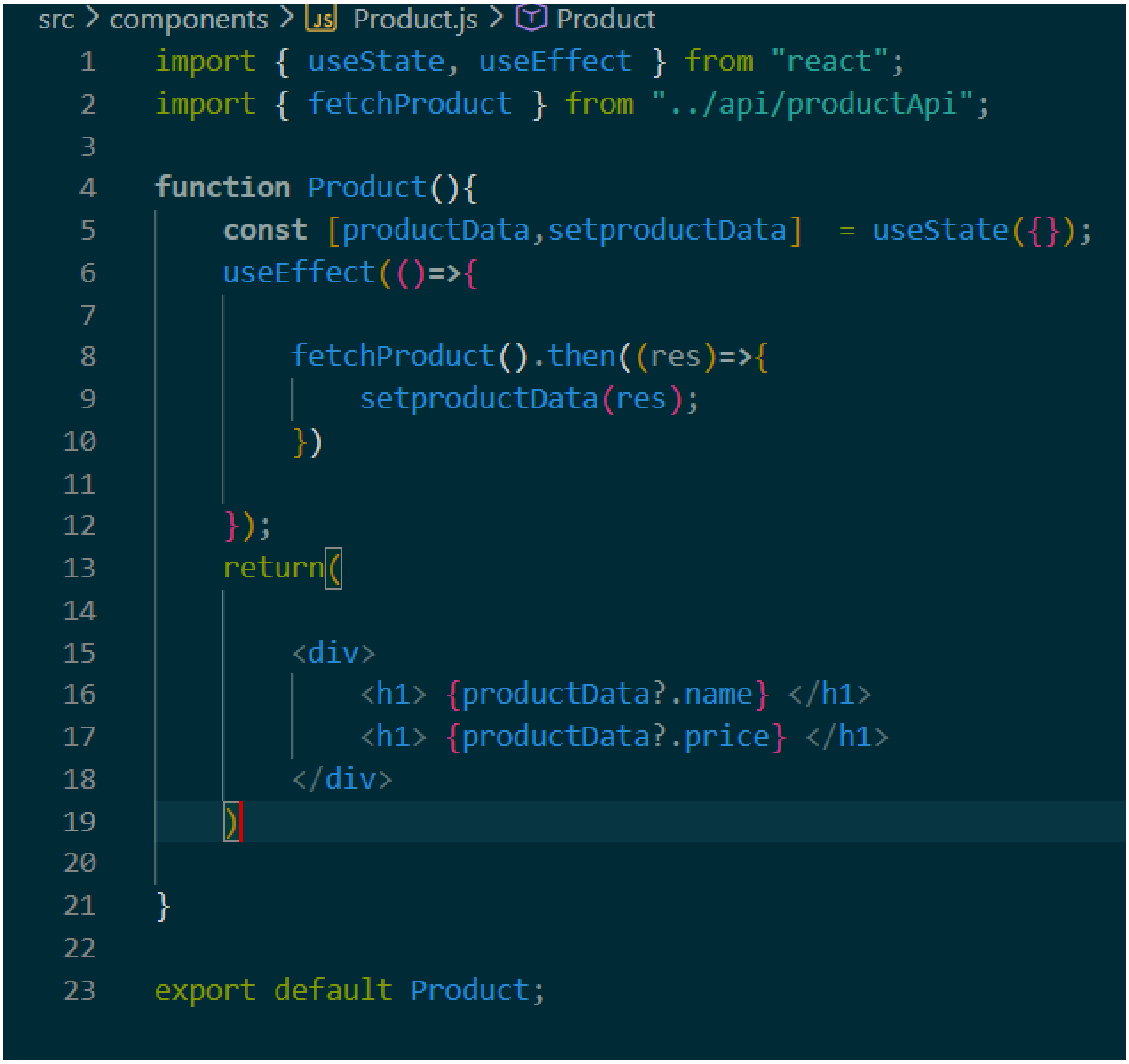
Implementation Description:

To ensure the correctness and reliability of the EcoTrack application, automated tests were implemented using Jest and React Testing Library to validate component rendering and asynchronous data fetching. Specifically, the Product component was tested to verify that product details are correctly retrieved from the API and displayed to the user. Additionally, application behavior was analyzed to identify and resolve stability issues, such as preventing infinite render loops by correctly managing useEffect dependencies, thereby optimizing performance and resource usage.

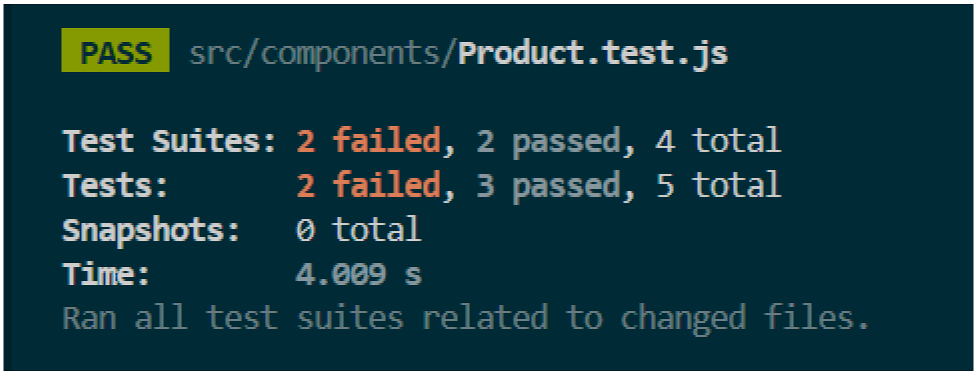
Sample Code Snippet:



```
import {render,screen} from "@testing-library/react";
import Product from "../components/Product";
import * as api from "../api/productApi";

jest.mock("../api/productApi");

test("render product",async()=>{
    api.fetchProduct.mockResolvedValue(({
        id:1,
        name:"laptop",
        price:500,
    }))


    render (<Product/>);

    const productName = await screen.findByText("laptop");
    const productPrice = await screen.findByText("500");

    expect(productName).toBeInTheDocument();
    expect(productPrice).toBeInTheDocument();


})
```



```
import { useState, useEffect } from "react";
import { fetchProduct } from "../api/productApi";

function Product(){
    const [productData,setproductData]  = useState({});
    useEffect(()=>{

        fetchProduct().then((res)=>{
            setproductData(res);
        })

    });
    return(

        <div>
            <h1> {productData?.name} </h1>
            <h1> {productData?.price} </h1>
        </div>
    )

}

export default Product;
```



```
 PASS  src/components/Product.test.js

Test Suites: 2 failed, 2 passed, 4 total
Tests:       2 failed, 3 passed, 5 total
Snapshots:   0 total
Time:        4.009 s
Ran all test suites related to changed files.
```

4. Output:

- Improved the runtime performance of the EcoTrack React application by minimizing unnecessary re-renders.
- Achieved faster initial load time and better scalability through effective code splitting and lazy loading.
- Developed a maintainable and performance-optimized React codebase using memoization best practices.
- Enhanced user experience with a clean, responsive, and enterprise-grade UI built using Material UI components.

5. Learning Outcomes

- **Ability to write and execute unit and integration tests** using Jest and React Testing Library to validate component behavior.
- **Skill in identifying and fixing bugs** by leveraging browser debugging tools and React DevTools.
- **Understanding of test-driven development (TDD) practices** to ensure reliable and maintainable React code.
- **Improved confidence in application stability** through automated test coverage and systematic debugging.

.