Experiment 4

Student Name: Shivanshu Ranjan **UID: 23BCS10193**
Branch: CSE **Section/Group: KRG 3-B**
Semester: 6th **Date of Performance:03/02/2026**
Subject Name: Full Stack Development – II **Subject Code: 23CSH-309**

1. **Aim**: To optimize the performance of the EcoTrack React application using **memoization techniques** and **code splitting**, and to enhance the user interface using **enterprise-grade Material UI components**.

2. **Objective**:

- Understand the causes of unnecessary re-renders in React applications
- Optimize React components using React.memo to prevent avoidable re-renders
- Apply useMemo to efficiently compute derived data and avoid redundant calculations
- Use useCallback to memoize event handler functions and improve component performance
- Implement lazy loading of components and routes using React.lazy and Suspense
- Reduce initial bundle size and improve application load performance through code splitting
- Enhance the visual appearance and usability of the EcoTrack application using Material UI components
- Design a clean, consistent, and responsive user interface using Material UI layouts and typography

3. **Implementation / Code:**

Tools & Technologies Used:-

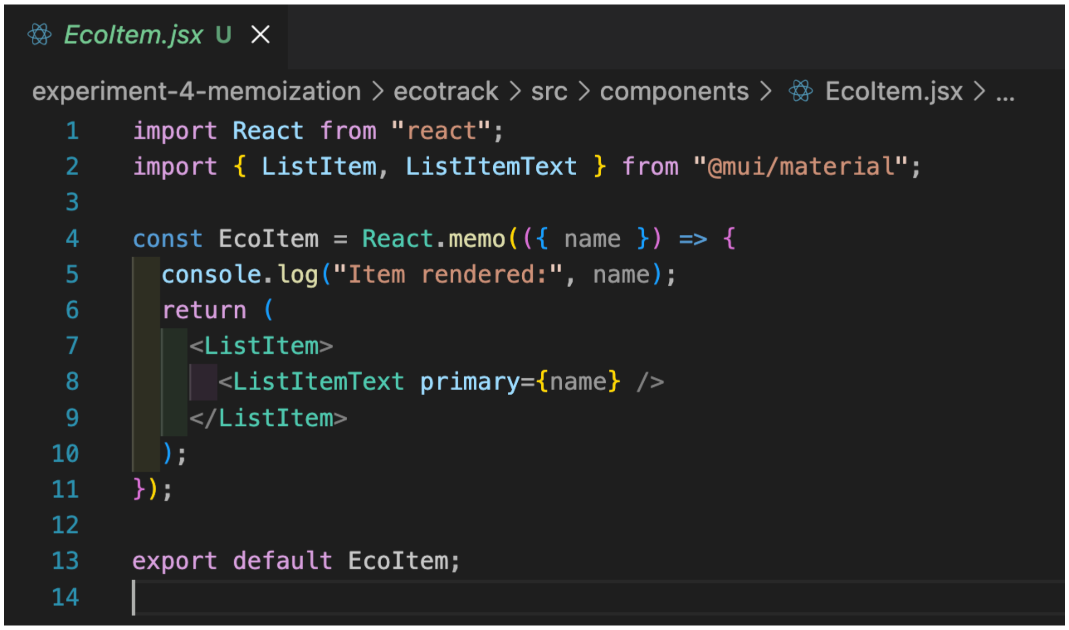
- React.js
- React Hooks (useMemo, useCallback)
- React Memo (React.memo)
- Material UI (MUI)
- JavaScript (ES6)
- VS Code
- Web Browser (Google Chrome / Firefox)

Implementation Description:-

- The EcoTrack application is optimized to improve **performance and user experience**.
- **Unnecessary component re-renders** are reduced using React.memo, which ensures components re-render only when their props change.

- The useMemo hook is used to memoize **expensive calculations**, preventing repeated execution on every render.
- The useCallback hook is applied to memoize **event handler functions**, ensuring stable function references across renders.
- Material UI components such as Container, Typography, Button, List, and Divider are used to create a **professional, responsive, and consistent UI**.
- The optimized structure improves application scalability, performance, and maintainability.

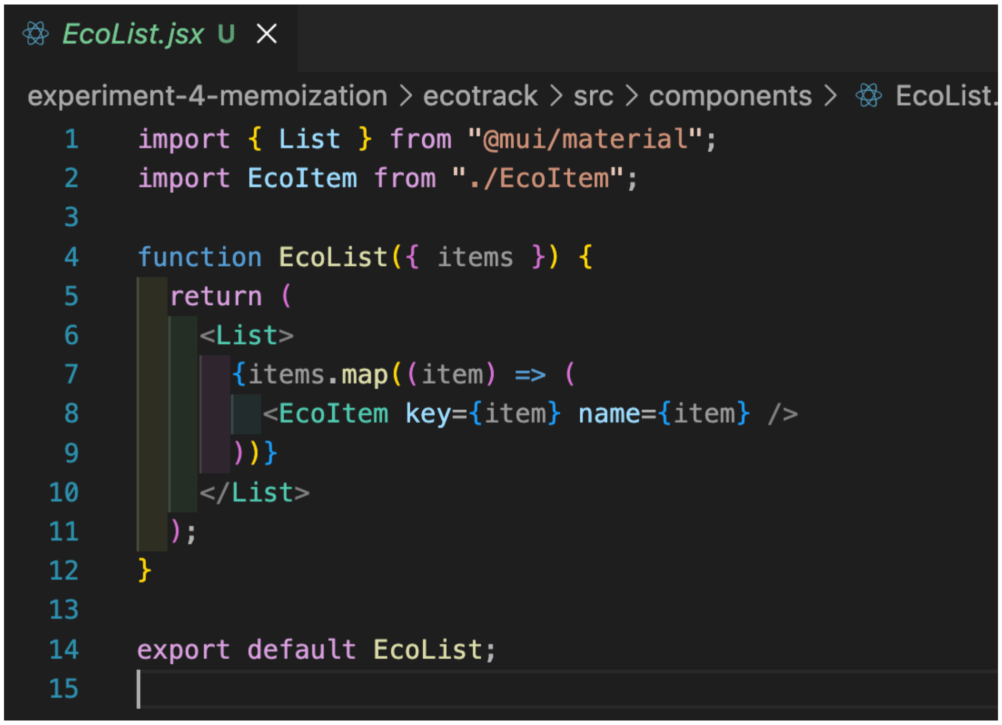
Sample Code Snippet:-



EcoItem.jsx U

experiment-4-memoization > ecotrack > src > components > EcoItem.jsx > ...

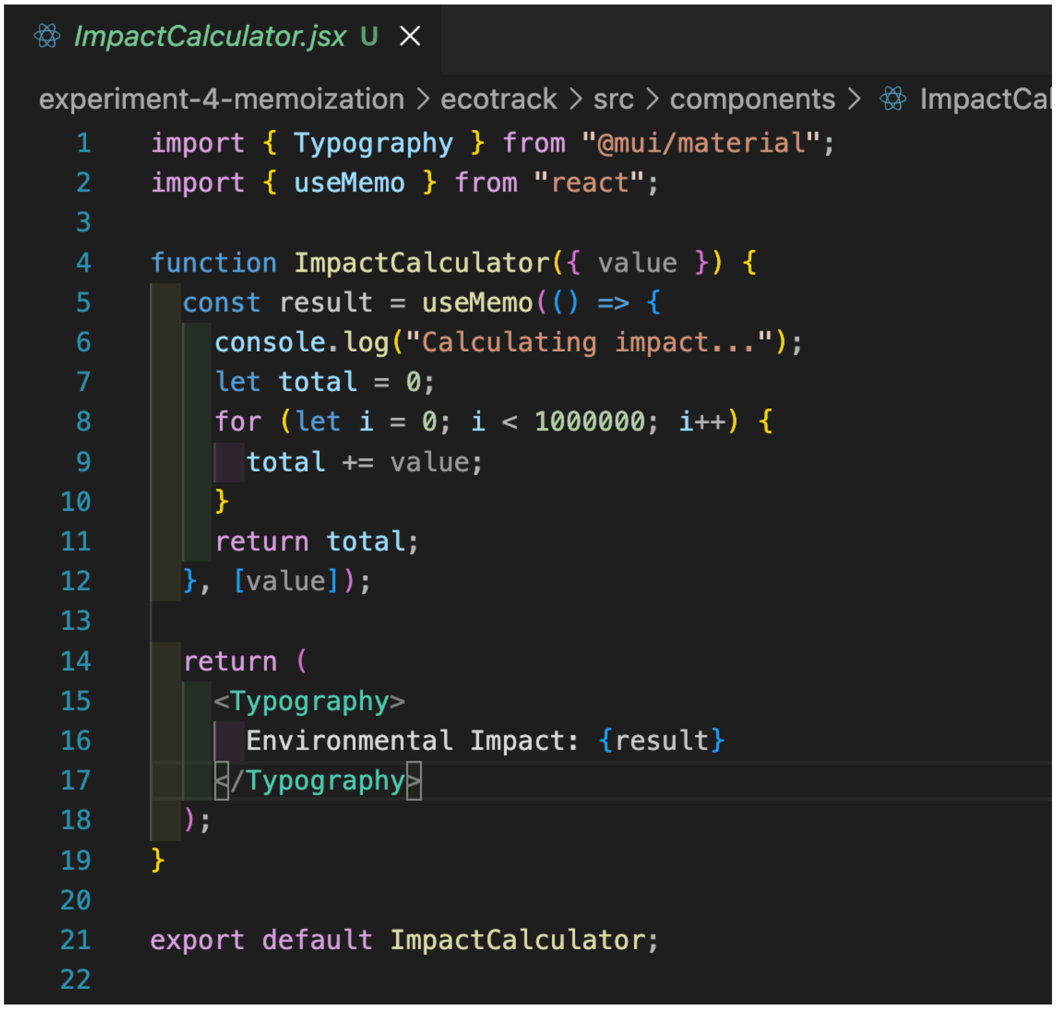
```jsx
import React from "react";
import { ListItem, ListItemText } from "@mui/material";

const EcoItem = React.memo(({ name }) => {
  console.log("Item rendered:", name);
  return (
    <ListItem>
      <ListItemText primary={name} />
    </ListItem>
  );
});

export default EcoItem;

```



EcoList.jsx U

experiment-4-memoization > ecotrack > src > components > EcoList.

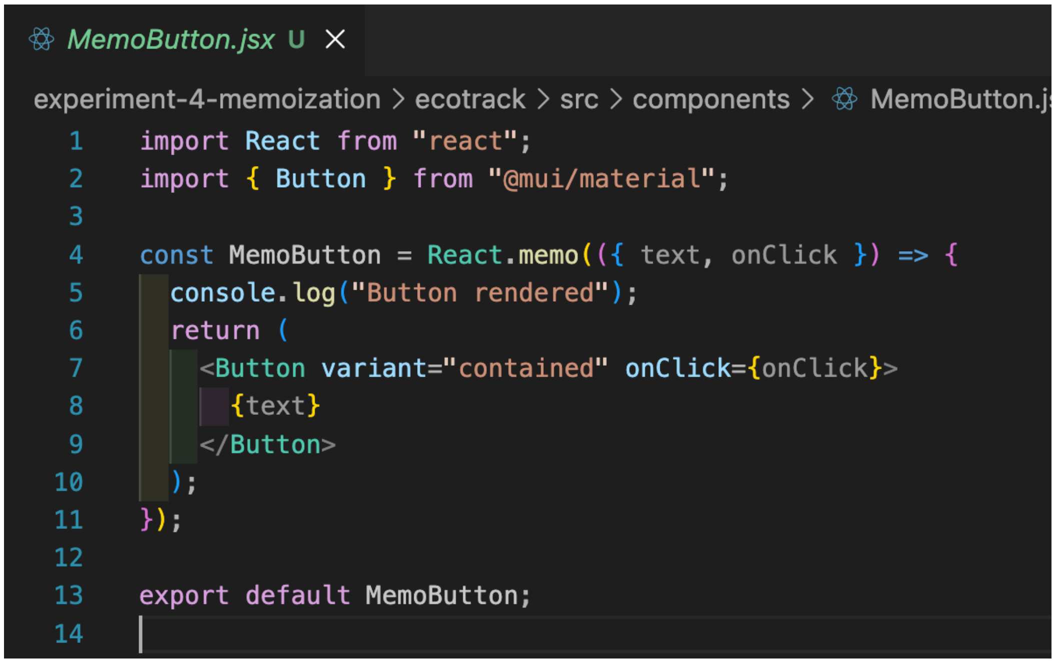
```jsx
import { List } from "@mui/material";
import EcoItem from "./EcoItem";

function EcoList({ items }) {
  return (
    <List>
      {items.map((item) => (
        <EcoItem key={item} name={item} />
      ))}
    </List>
  );
}

export default EcoList;

```



ImpactCalculator.jsx U ×

experiment-4-memoization > ecotrack > src > components > ImpactCal

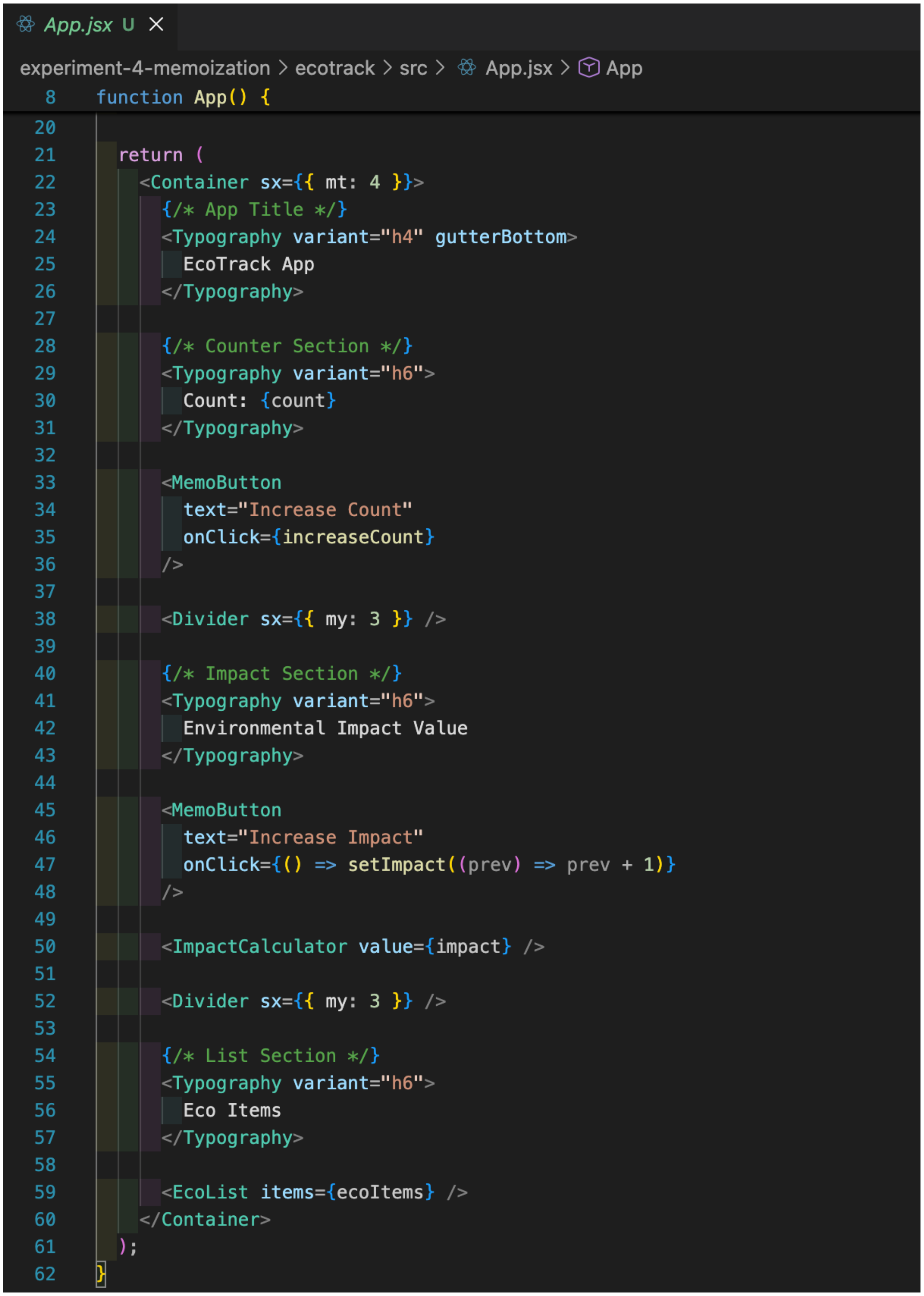
```jsx
import { Typography } from "@mui/material";
import { useMemo } from "react";

function ImpactCalculator({ value }) {
  const result = useMemo(() => {
    console.log("Calculating impact...");
    let total = 0;
    for (let i = 0; i < 1000000; i++) {
      total += value;
    }
    return total;
  }, [value]);

  return (
    <Typography>
      Environmental Impact: {result}
    </Typography>
  );
}

export default ImpactCalculator;

```



MemoButton.jsx U ×

experiment-4-memoization > ecotrack > src > components > MemoButton.j

```jsx
import React from "react";
import { Button } from "@mui/material";

const MemoButton = React.memo(({ text, onClick }) => {
  console.log("Button rendered");
  return (
    <Button variant="contained" onClick={onClick}>
      {text}
    </Button>
  );
});

export default MemoButton;

```

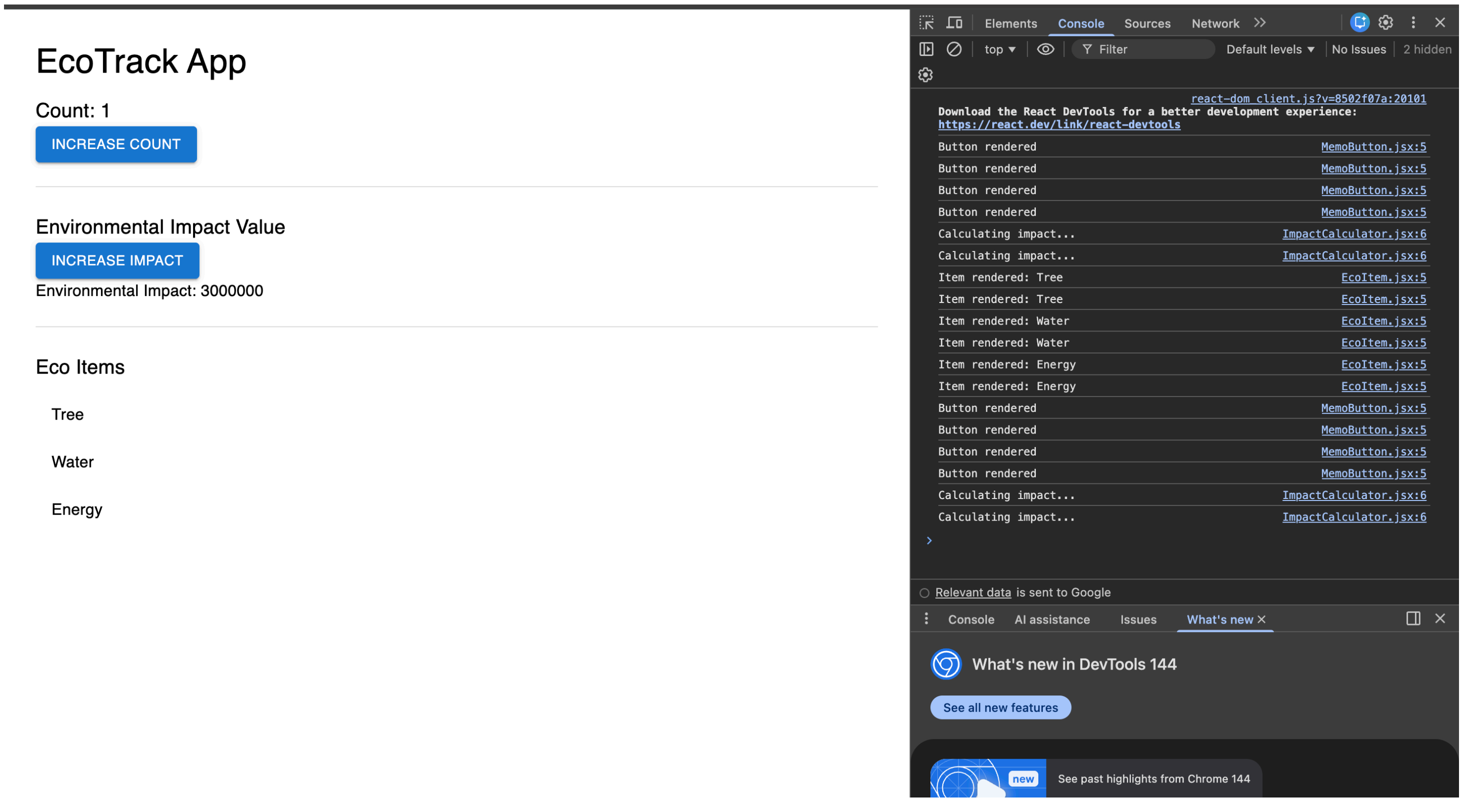


```
function App() {

  return (
    <Container sx={{ mt: 4 }}>
      {/* App Title */}
      <Typography variant="h4" gutterBottom>
        EcoTrack App
      </Typography>

      {/* Counter Section */}
      <Typography variant="h6">
        Count: {count}
      </Typography>

      <MemoButton
        text="Increase Count"
        onClick={increaseCount}
      />

      <Divider sx={{ my: 3 }} />

      {/* Impact Section */}
      <Typography variant="h6">
        Environmental Impact Value
      </Typography>

      <MemoButton
        text="Increase Impact"
        onClick={() => setImpact((prev) => prev + 1)}
      />

      <ImpactCalculator value={impact} />

      <Divider sx={{ my: 3 }} />

      {/* List Section */}
      <Typography variant="h6">
        Eco Items
      </Typography>

      <EcoList items={ecoItems} />
    </Container>
  );
}
```

4. Output:

- The EcoTrack application renders efficiently with reduced unnecessary re-renders

- Memoized components render only when required
- Expensive computations execute only when dependent values change
- Event handlers remain stable across renders
- Application UI is clean, responsive, and visually consistent
- Performance improvement is observed during state updates
- Material UI enhances professional appearance and usability



EcoTrack App

Count: 1

INCREASE COUNT

Environmental Impact Value

INCREASE IMPACT

Environmental Impact: 3000000

Eco Items

Tree

Water

Energy

```
                                   react-dom_client.js?v=8502f07a:20101
Download the React DevTools for a better development experience:
https://react.dev/link/react-devtools
Button rendered                                    MemoButton.jsx:5
Button rendered                                    MemoButton.jsx:5
Button rendered                                    MemoButton.jsx:5
Button rendered                                    MemoButton.jsx:5
Calculating impact...                          ImpactCalculator.jsx:6
Calculating impact...                          ImpactCalculator.jsx:6
Item rendered: Tree                                    EcoItem.jsx:5
Item rendered: Tree                                    EcoItem.jsx:5
Item rendered: Water                                   EcoItem.jsx:5
Item rendered: Water                                   EcoItem.jsx:5
Item rendered: Energy                                  EcoItem.jsx:5
Item rendered: Energy                                  EcoItem.jsx:5
Button rendered                                    MemoButton.jsx:5
Button rendered                                    MemoButton.jsx:5
Button rendered                                    MemoButton.jsx:5
Button rendered                                    MemoButton.jsx:5
Calculating impact...                          ImpactCalculator.jsx:6
Calculating impact...                          ImpactCalculator.jsx:6
```

5. **Learning Outcomes (What I Have Learnt):**

- Identify causes of performance issues in React applications
- Use React.memo to optimize component rendering
- Apply useMemo for expensive calculations
- Implement useCallback to optimize event handlers
- Improve application performance and scalability
- Use Material UI for enterprise-grade UI design
- Build efficient, optimized, and professional React applications