



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.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH
UNIVERSITY

Discover. Learn. Empower.

- 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 X
experiment-4-memoization > ecotrack > src > components > ⌘ EcoItem.jsx > ...
1  import React from "react";
2  import { ListItem, ListItemText } from "@mui/material";
3
4  const EcoItem = React.memo(({ name }) => {
5    console.log("Item rendered:", name);
6    return (
7      <ListItem>
8        <ListItemText primary={name} />
9      </ListItem>
10     );
11   );
12
13  export default EcoItem;
14  |
```

```
⌘ EcoList.jsx U X
experiment-4-memoization > ecotrack > src > components > ⌘ EcoList.jsx > ...
1  import { List } from "@mui/material";
2  import EcoItem from "./EcoItem";
3
4  function EcoList({ items }) {
5    return (
6      <List>
7        {items.map((item) => (
8          <EcoItem key={item} name={item} />
9        )));
10     </List>
11   );
12 }
13
14  export default EcoList;
15  |
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH
UNIVERSITY

Discover. Learn. Empower.

⚙ ImpactCalculator.jsx U X

experiment-4-memoization > ecotrack > src > components > ⚙ ImpactCalculator.jsx

```
1  import { Typography } from "@mui/material";
2  import { useMemo } from "react";
3
4  function ImpactCalculator({ value }) {
5    const result = useMemo(() => {
6      console.log("Calculating impact...");
7      let total = 0;
8      for (let i = 0; i < 1000000; i++) {
9        total += value;
10     }
11     return total;
12   }, [value]);
13
14   return (
15     <Typography>
16       Environmental Impact: {result}
17     </Typography>
18   );
19 }
20
21 export default ImpactCalculator;
22
```

⚙ MemoButton.jsx U X

experiment-4-memoization > ecotrack > src > components > ⚙ MemoButton.jsx

```
1  import React from "react";
2  import { Button } from "@mui/material";
3
4  const MemoButton = React.memo(({ text, onClick }) => {
5    console.log("Button rendered");
6    return (
7      <Button variant="contained" onClick={onClick}>
8        {text}
9      </Button>
10    );
11  });
12
13  export default MemoButton;
14
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
App.jsx U X
experiment-4-memoization > ecotrack > src > App.jsx > App
8  function App() {
20
21    return (
22      <Container sx={{ mt: 4 }}>
23        {/* App Title */}
24        <Typography variant="h4" gutterBottom>
25          EcoTrack App
26        </Typography>
27
28        {/* Counter Section */}
29        <Typography variant="h6">
30          Count: {count}
31        </Typography>
32
33        <MemoButton
34          text="Increase Count"
35          onClick={increaseCount}
36        />
37
38        <Divider sx={{ my: 3 }} />
39
40        {/* Impact Section */}
41        <Typography variant="h6">
42          Environmental Impact Value
43        </Typography>
44
45        <MemoButton
46          text="Increase Impact"
47          onClick={() => setImpact((prev) => prev + 1)}
48        />
49
50        <ImpactCalculator value={impact} />
51
52        <Divider sx={{ my: 3 }} />
53
54        {/* List Section */}
55        <Typography variant="h6">
56          Eco Items
57        </Typography>
58
59        <EcoList items={ecoItems} />
60      </Container>
61    );
62 }
```

4. Output:

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



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- 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

The screenshot shows the EcoTrack application running in a browser. On the left, there's a counter set at 1 with a blue 'INCREASE COUNT' button. Below it is an environmental impact section with a 'INCREASE IMPACT' button and a value of 3,000,000. On the right, there's a list titled 'Eco Items' with three items: Tree, Water, and Energy. On the far right, the browser's developer tools are open, specifically the 'Console' tab. It shows a log of React component renderings and calculations. The log includes entries like 'Button rendered' from 'MemoButton.jsx:5', 'Calculating impact...' from 'ImpactCalculator.jsx:6', and 'Item rendered' for various eco items like Tree, Water, and Energy, all from 'EcoItem.jsx:5'. The console also displays a message about downloading React DevTools for better development experience and a link to the download page. At the bottom of the dev tools, there's a 'What's new' section for Chrome 144.

```
react-dom_client.js?v=8502f07a:20101
Download the React DevTools for a better development experience:
https://react.dev/link/react-devtools
Button rendered
Button rendered
Button rendered
Button rendered
Calculating impact...
Calculating impact...
Item rendered: Tree
Item rendered: Tree
Item rendered: Water
Item rendered: Water
Item rendered: Energy
Item rendered: Energy
Button rendered
Button rendered
Button rendered
Calculating impact...
Calculating impact...
>
Relevant data is sent to Google
Console AI assistance Issues What's new ×
What's new in DevTools 144
See all new features
See past highlights from Chrome 144
```

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