DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment 4

Student Name: Ashaf Khan **UID: 23BCS11123**
Branch: BE CSE **Section/Group: 23BCSKRG_3A**
Semester: 6th **Date of Performance:03/02/26**
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: After completing this experiment, the student will be able to:

- 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:

- src/App.jsx - Added React.lazy and Suspense for code splitting
- src/components/Header.jsx - Added React.memo, useCallback, and Material UI components
- src/pages/Logs.jsx - Added useMemo for filtering and Material UI components
- src/pages/dashboard.jsx - Added useMemo, useCallback, and Material UI components

- src/pages/login.jsx - Added useCallback and Material UI components
- src/routes/ProtectedRoute.jsx - Added React.memo and cleaned up formatting

App.jsx:

```
import { lazy, Suspense } from "react";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import { CircularProgress, Box } from "@mui/material";

import Header from "./components/Header";
import ProtectedRoute from "./routes/ProtectedRoute";

// Lazy loading components - they load only when needed
const Logs = lazy(() => import("./pages/Logs"));
const Dashboard = lazy(() => import("./pages/dashboard"));
const Login = lazy(() => import("./pages/login"));
const DashboardLayout = lazy(() => import("./pages/dashboardlayout"));
const DashboardSummary = lazy(() => import("./pages/dashboardsummary"));
const DashboardAnalytics = lazy(() => import("./pages/dashboardanalytics"));

// Loading fallback component
const LoadingFallback = () => (
  <Box sx={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height:
    '100vh' }}>
   <CircularProgress size={60} />
  </Box>
);

function App() {
  return (
    <BrowserRouter>
      <Suspense fallback={<LoadingFallback />}>
        <Routes>
          <Route path="/login" element={<Login />} />

          <Route
            path="/"
       element={
        <ProtectedRoute>
```

```
            <Header />
            <DashboardLayout />
          </ProtectedRoute>
        }
      >
        <Route index element={<Dashboard />} />
        <Route path="summary" element={<DashboardSummary />} />
        <Route path="analytics" element={<DashboardAnalytics />} />
        <Route path="logs" element={<Logs />} />
      </Route>
    </Routes>
  </Suspense>
 </BrowserRouter>
 );
}

export default App;
```

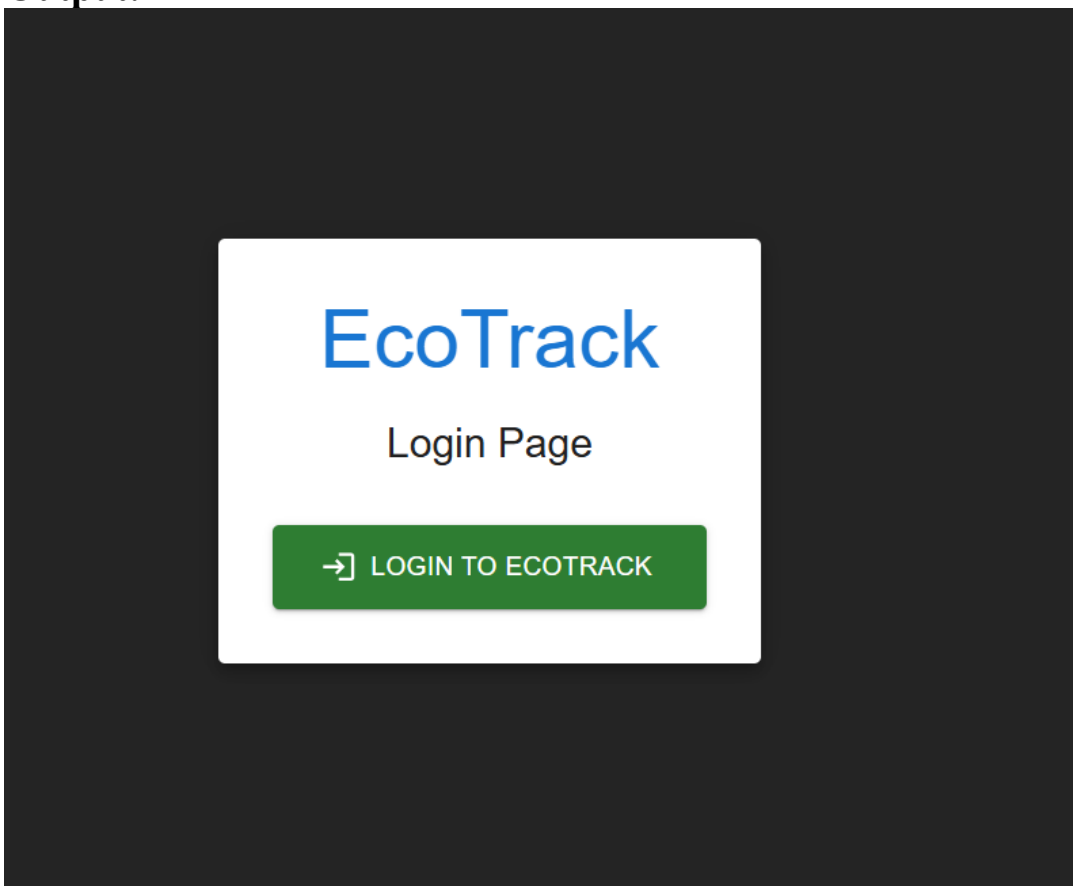
dashboard.jsx:

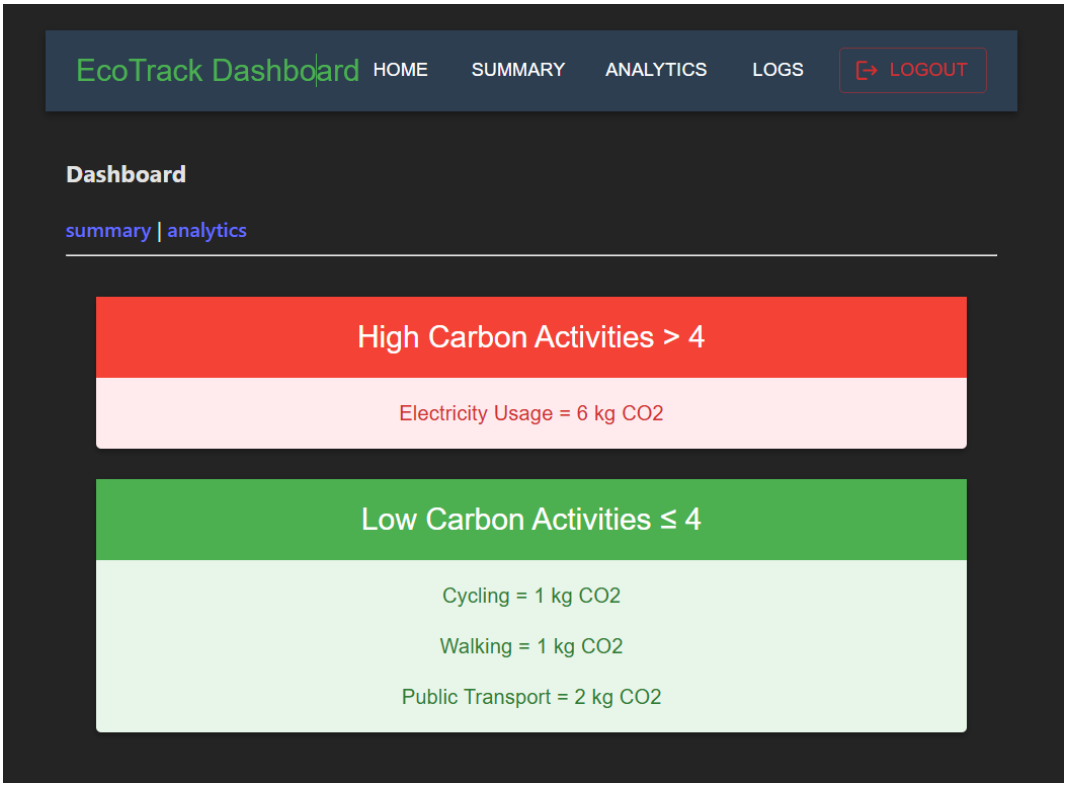
```
import { useDispatch, useSelector } from 'react-redux';
import { useEffect, useMemo, useCallback } from 'react';
import { fetchLogs } from '../store/logSlice';
import { Box, Typography, List, ListItem, Button, CircularProgress, Paper } from
    '@mui/material';
import RefreshIcon from '@mui/icons-material/Refresh';

const Dashboard = () => {
   const dispatch = useDispatch();
   const { data: logs, status } = useSelector((state) => state.logs);

   useEffect(() => {
      if (status === 'idle') {
         dispatch(fetchLogs());
      }
   }, [status, dispatch]);

   // useCallback memoizes the function to prevent recreation on every render
   const handleRefresh = useCallback(() => {
      dispatch(fetchLogs());
```

```
}, [dispatch]);

if (status === 'loading') {
  return (
    <Box sx={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height:
 '50vh' }}>
      <CircularProgress />
    </Box>
  );
}

// useMemo prevents recalculating totals on every render
const totalActivities = useMemo(() => {
  return logs.reduce((acc, log) => {
    acc[log.activity] = (acc[log.activity] || 0) + log.carbon;
    return acc;
  }, {});
}, [logs]);

// useMemo for filtering operations
const highCarbon = useMemo(() => logs.filter(log => log.carbon > 4), [logs]);
const lowCarbon = useMemo(() => logs.filter(log => log.carbon <= 4), [logs]);

return (
  <Box sx={{ p: 3 }}>
    <Paper elevation={3} sx={{ p: 3, mb: 3 }}>
      <Typography variant="h4" gutterBottom>Total Activities</Typography>
      <List>
        {Object.entries(totalActivities).map(([activity, carbon]) => (
          <ListItem key={activity}>
            <Typography>{activity}: {carbon} Kg CO2</Typography>
          </ListItem>
        ))}
      </List>
    </Paper>

    <Paper elevation={3} sx={{ p: 2, mb: 2, backgroundColor: '#ffebee' }}>
      <Typography variant="h5" sx={{ color: '#d32f2f', mb: 1 }}>High Carbon
 (&gt; 4 Kg)</Typography>
```

```
            <List>
              {highCarbon.map((log) => (
                <ListItem key={log.id}>
                  <Typography>{log.activity}</Typography>
                </ListItem>
              ))}
            </List>
          </Paper>

          <Paper elevation={3} sx={{ p: 2, mb: 3, backgroundColor: '#e8f5e9' }}>
            <Typography variant="h5" sx={{ color: '#2e7d32', mb: 1 }}>Low Carbon (≤ 4
      Kg)</Typography>
            <List>
              {lowCarbon.map((log) => (
                <ListItem key={log.id}>
                  <Typography>{log.activity}</Typography>
                </ListItem>
              ))}
            </List>
          </Paper>

          <Button
            variant="contained"
            color="primary"
            startIcon={<RefreshIcon />}
            onClick={handleRefresh}
            disabled={status === 'loading'}
          >
            {status === 'loading' ? 'Loading...' : 'Refresh Logs'}
          </Button>
        </Box>
      );
};

export default Dashboard;
```

4. Output:


EcoTrack
Login Page
LOGIN TO ECOTRACK


EcoTrack Dashboard
HOME
SUMMARY
ANALYTICS
LOGS
LOGOUT
Dashboard
summary | analytics
High Carbon Activities > 4
Electricity Usage = 6 kg CO2
Low Carbon Activities ≤ 4
Cycling = 1 kg CO2
Walking = 1 kg CO2
Public Transport = 2 kg CO2

5. Learning Outcome:

- Identify and reduce unnecessary re-renders in React applications.
- Optimize React components using React.memo, useMemo, and useCallback.
- Implement lazy loading and code splitting to improve application performance.
- Enhance overall React app efficiency through performance optimization techniques.
- Design a clean, responsive user interface using Material UI components.