```
import React, { useState, useEffect, useCallback, useRef } from "react";
import { ColorModeContext, useMode } from "./theme";
import { CssBaseline, ThemeProvider } from "@mui/material";
import {
  Routes,
  Route,
  Navigate,
  useLocation,
  useNavigate,
} from "react-router-dom";
import Topbar from "./scenes/global/topbar";
import Sidebar from "./scenes/global/sidebar";
import Dashboard from "./scenes/dashboard";
import Location from "./scenes/device-location";
import HistoricalData from "./scenes/historical";
import DeviceConfig from "./scenes/device-config";
import LoginPage from "./scenes/login";

function App() {
  const [theme, colorMode] = useMode();
  const [loggedIn, setLoggedIn] = useState(false);
  const location = useLocation();
  const navigate = useNavigate();
  const inactivityTimer = useRef(null);

  const handleLogout = useCallback(() => {
    localStorage.removeItem("isLoggedIn");
    setLoggedIn(false);
    navigate("/");
    clearTimeout(inactivityTimer.current);
  }, [navigate, inactivityTimer]);

  const startInactivityTimer = useCallback(() => {
    clearTimeout(inactivityTimer.current);
    inactivityTimer.current = setTimeout(
      () => {
        handleLogout();
      },
      30 * 60 * 1000,
    );
  }, [handleLogout]);

  useEffect(() => {
    const isAuthenticated = localStorage.getItem("isLoggedIn");
    if (isAuthenticated === "true") {
      setLoggedIn(true);
  }

  const handleWindowClose = (event) => {
    if (
      !event.currentTarget.performance ||
      event.currentTarget.performance.navigation.type !== 1
    ) {
      handleLogout();
    }
  };

  window.addEventListener("beforeunload", handleWindowClose);

  return () => {
    window.removeEventListener("beforeunload", handleWindowClose);
    clearTimeout(inactivityTimer.current);
  };
}, [loggedIn, handleLogout, startInactivityTimer]);

useEffect(() => {
  if (loggedIn && location.pathname !== "/") {
    localStorage.setItem("lastVisitedRoute", location.pathname);
  }
}, [loggedIn, location.pathname]);

useEffect(() => {
  if (loggedIn && location.pathname === "/") {
    const lastVisitedRoute = localStorage.getItem("lastVisitedRoute");
    if (lastVisitedRoute) {
      navigate(lastVisitedRoute);
    } else {
      navigate("/dashboard");
    }
  }
}, [loggedIn, location.pathname, navigate]);

const handleLogin = () => {
  localStorage.setItem("isLoggedIn", "true");
  setLoggedIn(true);
  navigate("/dashboard");
  startInactivityTimer();
};

const handleUserActivity = useCallback(() => {
  clearTimeout(inactivityTimer.current);
  startInactivityTimer();
```

```
}, [startInactivityTimer]);

useEffect(() => {
  document.addEventListener("mousedown", handleUserActivity);
  document.addEventListener("keydown", handleUserActivity);

  return () => {
    document.removeEventListener("mousedown", handleUserActivity);
    document.removeEventListener("keydown", handleUserActivity);
  };
}, [handleUserActivity]);

return (
  <ColorModeContext.Provider value={colorMode}>
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <div className="app">
        {loggedIn && <Sidebar onLogout={handleLogout} />}
        <main className="content">
          {loggedIn && <Topbar loggedIn={loggedIn} onLogout={handleLogout} />}
          <Routes>
            <Route
              path="/"
              element={
                loggedIn ? (
                  <Navigate to="/dashboard" replace />
                ) : (
                  <LoginPage onLogin={handleLogin} />
                )
              }
            />
            <Route
              path="/dashboard"
              element={loggedIn ? <Dashboard /> : <Navigate to="/" replace />}
            />
            <Route
              path="/device-location"
              element={loggedIn ? <Location /> : <Navigate to="/" replace />}
            />
            <Route
              path="/device-config"
              element={
                loggedIn ? <DeviceConfig /> : <Navigate to="/" replace />
              }
            />
            <Route
```

```
                path="/historical"
                element={
                  loggedIn ? <HistoricalData /> : <Navigate to="/" replace />
                }
              />
            </Routes>
          </main>
        </div>
      </ThemeProvider>
    </ColorModeContext.Provider>
  );
}

export default App;
```