Experiment 2

Aim : Experiment based on React Hooks (useEffect, useContext, custom hooks)

Theory :

Core Concept: React Hooks
This project demonstrates React Hooks - a modern pattern that allows functional components to manage state and side effects without class components.
Hooks Used:

- useState: Manages theme state (light/dark)
- useEffect: Handles API calls and DOM manipulation
- useContext: Provides global state management
- useCallback: Optimizes function performance

Key Patterns:

- Custom Hook: useRandomUser() encapsulates API logic
- Context API: Eliminates prop drilling for state sharing
- Functional Components: Pure functions returning JSX

30% Extra Features Added:

- Tailwind CSS Dark/Light Mode: Theme switching with CSS custom properties
- Loading Spinner: Animated spinner during API calls
- Enhanced UI: Responsive design with smooth transitions

Modern React development emphasizes functional components, hooks for state management, and reusable logic through custom hooks, demonstrating clean architecture principles.

Code :

Index.css

```
# index.css U ×
src > # index.css > ...
1   @import "tailwindcss";
2
3   /* CSS Custom Properties for theming */
4   :root {
5     --bg-primary: #f3f4f6;
6     --bg-secondary: #ffffff;
7     --text-primary: #111827;
8     --text-secondary: #6b7280;
9     --border-color: #d1d5db;
10  }
11
12  .dark {
13    --bg-primary: #111827;
14    --bg-secondary: #1f2937;
15    --text-primary: #ffffff;
16    --text-secondary: #d1d5db;
17    --border-color: #4b5563;
18  }
19
20  /* Apply custom properties */
21  body {
22    background-color: var(--bg-primary);
23    color: var(--text-primary);
24    transition: background-color 0.3s ease, color 0.3s ease;
25  }
```

App.jsx

App.jsx U

src > App.jsx > ...

```jsx
import React, { useState, useEffect, createContext, useContext, useCallback } from "react";

/* ---------------- CONTEXT ---------------- */
const AppContext = createContext();

/* ---------------- CUSTOM HOOK ---------------- */
Windsurf: Refactor | Explain |
const useRandomUser = () => {
  const [user, setUser] = useState(null);

  const fetchUser = useCallback(async () => {
    try {
      setUser(null); // show loading state
      const res = await fetch("https://randomuser.me/api/");
      const data = await res.json();
      setUser({ ...data.results[0], key: Date.now() });
    } catch (err) {
      console.error(err);
    }
  }, []);

  useEffect(() => {
    fetchUser();
  }, [fetchUser]);

  return { user, fetchUser };
};
```

```jsx
/* ---------------- PROVIDER (handles dark class on <html>) ---------------- */
Windsurf: Refactor | Explain | ×
const AppProvider = ({ children }) => {
  const [theme, setTheme] = useState("light");
  const { user, fetchUser } = useRandomUser();

  // keep a proper "dark" class on documentElement so Tailwind's dark: variants work
  useEffect(() => {
    console.log("Theme changed to:", theme);
    if (theme === "dark") {
      document.documentElement.classList.add("dark");
      console.log("Added dark class to html");
    } else {
      document.documentElement.classList.remove("dark");
      console.log("Removed dark class from html");
    }
  }, [theme]);

  Windsurf: Refactor | Explain | Generate JSDoc | ×
  const toggleTheme = () => {
    console.log("Toggle theme clicked, current theme:", theme);
    const newTheme = theme === "light" ? "dark" : "light";
    console.log("Setting new theme to:", newTheme);
    setTheme(newTheme);
  };

  return (
    <AppContext.Provider value={{ theme, toggleTheme, user, fetchUser }}>
      {children}
    </AppContext.Provider>
  );
```

```jsx
/* ---------------- COMPONENTS ---------------- */
Windsurf: Refactor | Explain | ×
const UserCard = () => {
  const { user } = useContext(AppContext);

  if (!user) {
    return (
      <div className="flex flex-col items-center">
        <div className="w-10 h-10 border-4 border-blue-500 border-t-transparent rounded-full animate-spin"></div>
        <p className="mt-3 text-lg">Fetching user...</p>
      </div>
    );
  }

  return (
    <div
      key={user.key}
      className="p-6 border-2 rounded-xl max-w-xs text-center shadow-lg animate-fadeIn"
      style={{
        backgroundColor: 'var(--bg-secondary)',
        color: 'var(--text-primary)',
        borderColor: 'var(--border-color)'
      }}
    >
      <img
        src={user.picture.large}
        alt="user"
        className="rounded-full mb-3 mx-auto border-4"
        style={{ borderColor: 'var(--border-color)' }}
      />
      <h3 className="text-xl font-semibold" style={{ color: 'var(--text-primary)' }}>
        {user.name.first} {user.name.last}
      </h3>
      <p className="text-sm" style={{ color: 'var(--text-secondary)' }}>{user.email}</p>
      <p className="text-sm" style={{ color: 'var(--text-secondary)' }}>{user.location.country}</p>
    </div>
  );
};
```

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
const Controls = () => {
  const { fetchUser, toggleTheme, theme } = useContext(AppContext);
  return (
    <div className="mt-5 flex flex-col gap-3 items-center">
      <div className="text-sm" style={{ color: 'var(--text-secondary)' }}>
        Current theme: {theme}
      </div>
      <div className="flex gap-3">
        <button
          onClick={fetchUser}
          className="px-4 py-2 bg-blue-500 hover:bg-blue-600 text-white rounded-lg transition"
        >
          Get New User
        </button>
        <button
          onClick={toggleTheme}
          className="px-4 py-2 bg-gray-700 hover:bg-gray-800 text-white rounded-lg transition"
        >
          Switch to {theme === "light" ? "Dark" : "Light"} Mode
        </button>
      </div>
    </div>
  );
};
```

```
/* ---------------- APP ---------------- */
Windsurf: Refactor | Explain | ×
const App = () => {
  // we don't need to manage dark styling here because .dark is on <html>
  return (
    <div
      className="min-h-screen flex flex-col items-center justify-center transition-colors duration-300"
      style={{
        backgroundColor: 'var(--bg-primary)',
        color: 'var(--text-primary)'
      }}
    >
      <h1 className="text-3xl font-bold mb-5" style={{ color: 'var(--text-primary)' }}>Random User Fetcher</h1>
      <UserCard />
      <Controls />
    </div>
  );
};

Windsurf: Refactor | Explain | Generate JSDoc | ×
export default function RootApp() {
  return (
    <AppProvider>
      <App />
    </AppProvider>
  );
}
```

Output :

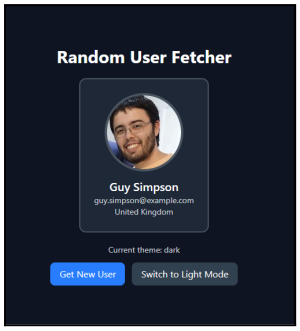

Figure 2.1:(DARK MODE)

Figure 2.2:(LIGHT MODE)

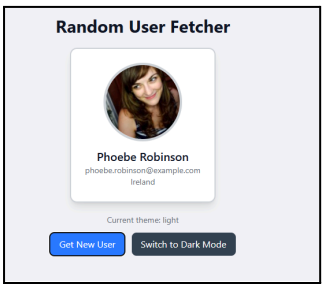

Figure 2.3:(NEW RANDOM USER FETCH USING UseEffect Hook)

Conclusion :

Thus we have used react hooks to create an application which fetches data from an API and shows it on the UI using tailwind css