Experiment No. 2

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

Prerequisites :-
Before building this project, the following knowledge/tools are required:

- **HTML, CSS, JavaScript** - for basic web development foundation.
- **React.js** - Functional components, JSX, props, and state.
- **React Hooks** - useState, useEffect, useContext, custom hooks.
- **Node.js & npm** - To set up the React environment.
- **Code Editor** - VS Code (or equivalent).

Theory :-
This experiment demonstrates **how React Hooks simplify state management, side effects, context sharing, and code reusability**.

Core Hooks Used :-

1. **useState**
    - Manages local state inside components like Counter (count value), Calculator (num1, num2), and useLocalStorage.
2. **useEffect**
    - Handles **side effects** such as updating document.title when the count changes in Counter.
    - Also syncs values to localStorage in the custom hook useLocalStorage.
3. **useContext**
    - Provides and consumes theme values (light or dark) across the app through ThemeContext.
    - Used in Profile to **toggle theme** globally.
4. **Custom Hook (useLocalStorage)**
    - Encapsulates logic to **persist data in localStorage**.
    - Improves reusability and abstraction compared to writing localStorage logic in each component.

Program Code :-

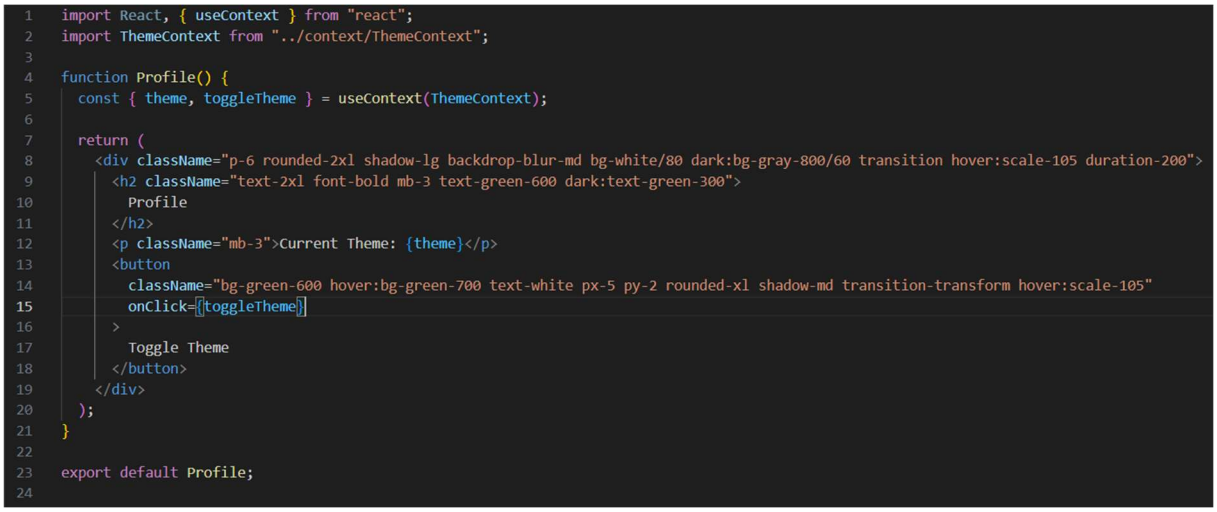
- Calculator.js

```jsx
import React, { useState, useCallback, useMemo, useRef } from "react";

function Calculator() {
  const [num1, setNum1] = useState("");
  const [num2, setNum2] = useState("");
  const inputRef = useRef(null);

  const addNumbers = useCallback(() => {
    return Number(num1) + Number(num2);
  }, [num1, num2]);

  const squaredSum = useMemo(() => {
    return addNumbers() * addNumbers();
  }, [addNumbers]);

  return (
    <div className="p-6 rounded-2xl shadow-lg backdrop-blur-md bg-white/80 dark:bg-gray-800/60 transition hover:scale-105 duration-200 md:col-span-2">
      <h2 className="text-2xl font-bold mb-3 text-pink-600 dark:text-pink-300">
        Calculator (Extra Hooks)
      </h2>
      <div className="flex flex-col md:flex-row gap-3 mb-4">
        <input
          ref={inputRef}
          type="number"
          placeholder="Enter num1"
          className="border border-gray-400 dark:border-gray-600 rounded-lg px-3 py-2 focus:outline-none focus:ring-2 focus:ring-pink-500"
          value={num1}
          onChange={(e) => setNum1(e.target.value)}
        />
        <input
          type="number"
          placeholder="Enter num2"
          className="border border-gray-400 dark:border-gray-600 rounded-lg px-3 py-2 focus:outline-none focus:ring-2 focus:ring-pink-500"
          value={num2}
          onChange={(e) => setNum2(e.target.value)}
        />
      </div>
      <div className="text-lg mb-3">
        <p>Sum: <span className="font-semibold">{addNumbers()}</span></p>
        <p>Square of Sum (useMemo): <span className="font-semibold">{squaredSum}</span></p>
      </div>
      <button
        onClick={() => inputRef.current.focus()}
        className="bg-pink-600 hover:bg-pink-700 text-white px-5 py-2 rounded-xl shadow-md transition-transform hover:scale-105"
      >
        Focus First Input (useRef)
      </button>
    </div>
  );
}

export default Calculator;
```

- Counter.js

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

function Counter() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `Clicked ${count} times`;
  }, [count]);

  return (
    <div className="p-6 rounded-2xl shadow-lg backdrop-blur-md bg-white/80 dark:bg-gray-800/60 transition hover:scale-105 duration-200">
      <h2 className="text-2xl font-bold mb-3 text-purple-600 dark:text-purple-300">
        Counter
      </h2>
      <p className="mb-3 text-lg">You clicked {count} times</p>
      <button
        className="bg-purple-600 hover:bg-purple-700 text-white px-5 py-2 rounded-xl shadow-md transition-transform hover:scale-105"
        onClick={() => setCount(count + 1)}
      >
        Click Me
      </button>
    </div>
  );
}

export default Counter;
```

- Profile.js



```jsx
import React, { useContext } from "react";
import ThemeContext from "../context/ThemeContext";

function Profile() {
  const { theme, toggleTheme } = useContext(ThemeContext);

  return (
    <div className="p-6 rounded-2xl shadow-lg backdrop-blur-md bg-white/80 dark:bg-gray-800/60 transition hover:scale-105 duration-200">
      <h2 className="text-2xl font-bold mb-3 text-green-600 dark:text-green-300">
        Profile
      </h2>
      <p className="mb-3">Current Theme: {theme}</p>
      <button
        className="bg-green-600 hover:bg-green-700 text-white px-5 py-2 rounded-xl shadow-md transition-transform hover:scale-105"
        onClick={toggleTheme}
      >
        Toggle Theme
      </button>
    </div>
  );
}

export default Profile;
```

- ThemeContext.js

```jsx
import { createContext } from "react";

const ThemeContext = createContext();
export default ThemeContext;
```

- useLocalStorage.js

```jsx
import { useState, useEffect } from "react";

function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    const stored = localStorage.getItem(key);
    return stored ? JSON.parse(stored) : initialValue;
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [key, value]);

  return [value, setValue];
}

export default useLocalStorage;
```

- App.js

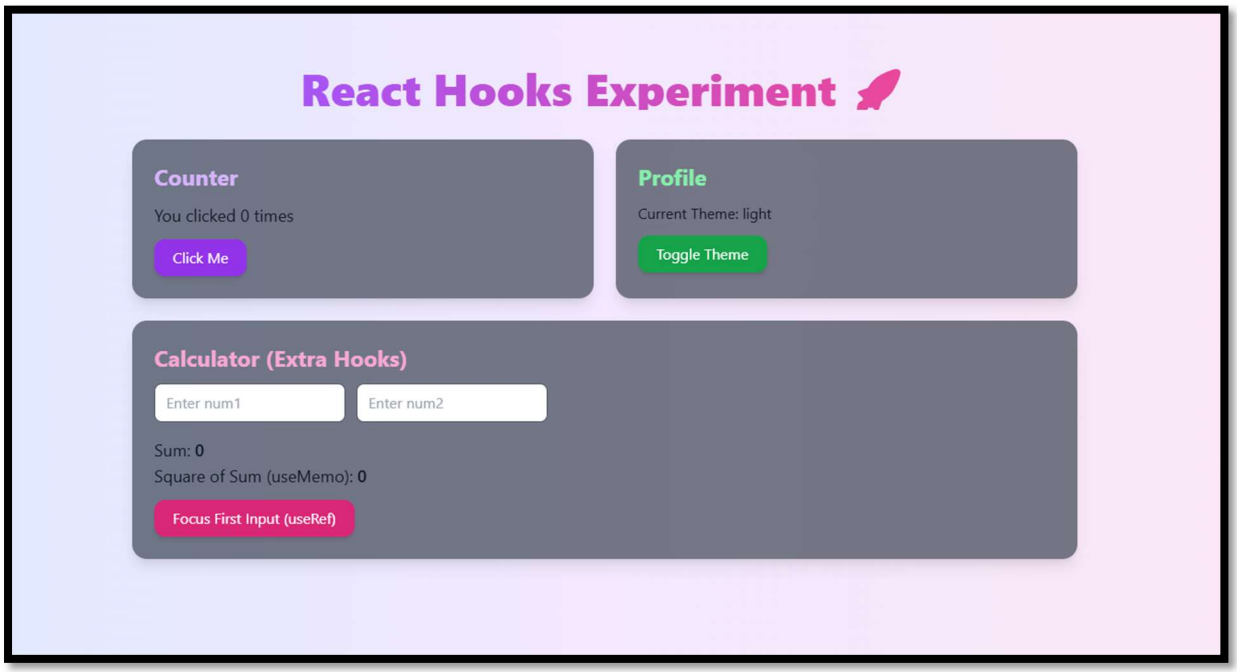
```jsx
import React from "react";
import Counter from "./components/Counter";
import Profile from "./components/Profile";
import Calculator from "./components/Calculator";
import useLocalStorage from "./hooks/useLocalStorage";
import ThemeContext from "./context/ThemeContext";

function App() {
  const [theme, setTheme] = useLocalStorage("theme", "light");

  const toggleTheme = () => {
    setTheme((prev) => (prev === "light" ? "dark" : "light"));
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      <div
        className={`min-h-screen flex flex-col items-center justify-center px-4 py-8
          ${
            theme === "light"
              ? "bg-gradient-to-r from-blue-100 via-purple-100 to-pink-100 text-gray-900"
              : "bg-gradient-to-br from-gray-900 via-gray-800 to-black text-white"
          }`}
      >
        <h1 className="text-4xl md:text-5xl font-extrabold mb-8 bg-gradient-to-r from-purple-500 to-pink-500 bg-clip-text text-transparent">
          React Hooks Experiment 🚀
        </h1>

        <div className="grid md:grid-cols-2 gap-6 w-full max-w-5xl">
          <Counter />
          <Profile />
          <Calculator />
        </div>

      </div>
    </ThemeContext.Provider>
  );
}export default App;
```

- index.js

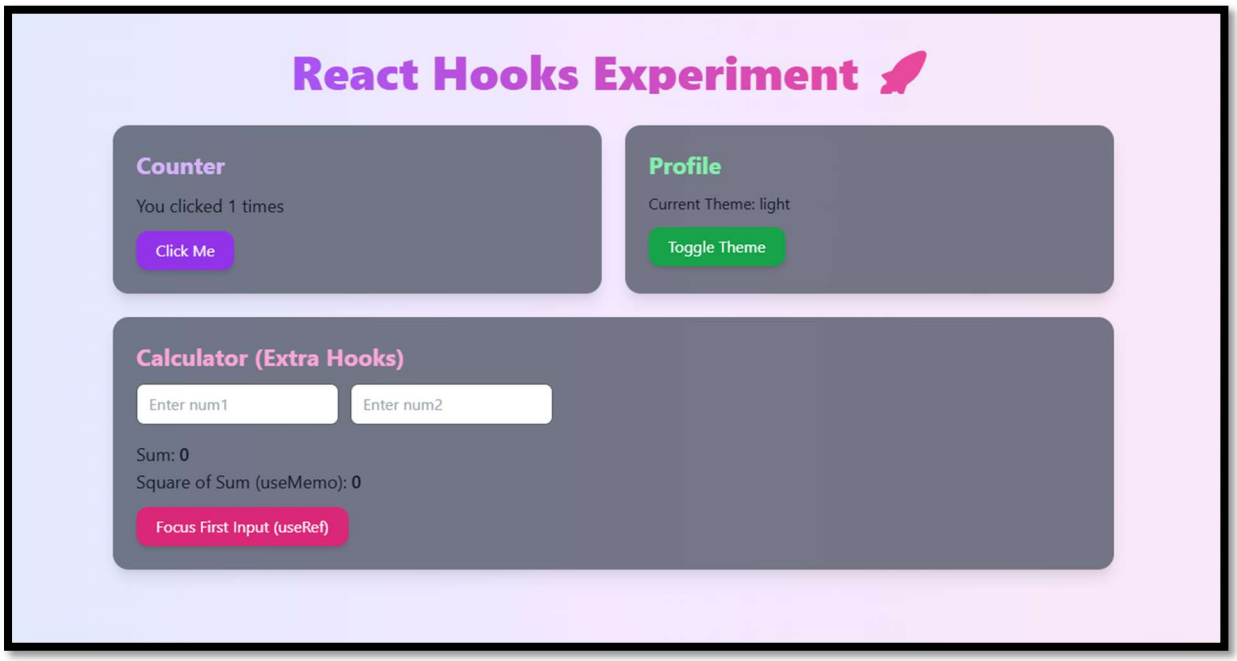
```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import "./index.css";


const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Output :-

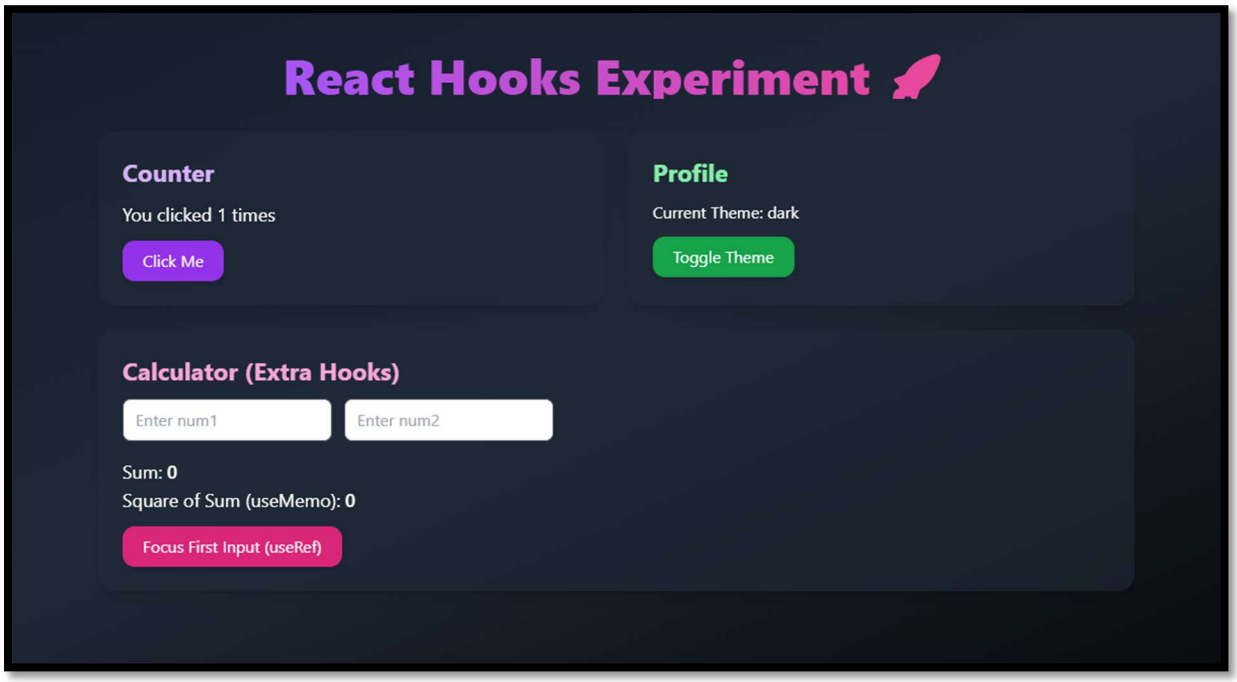
- HomePage :-



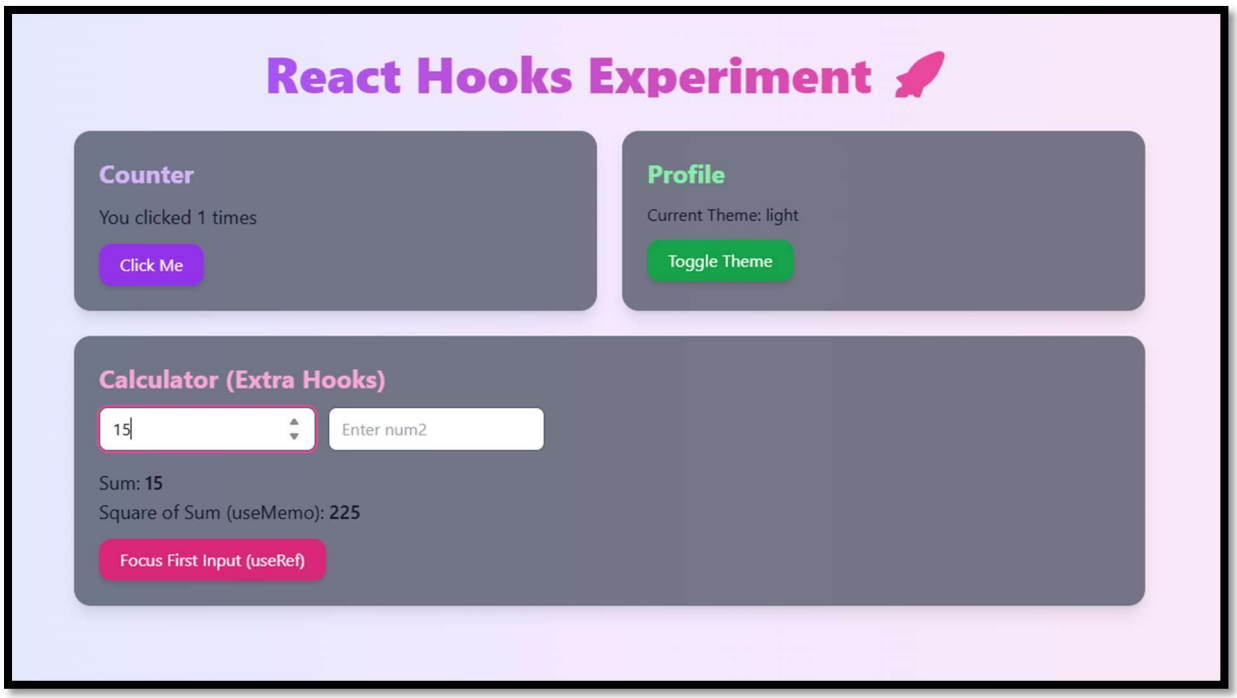
- After clicking on “Click Me” button of “Counter” section (useState & useEffect is used) :-

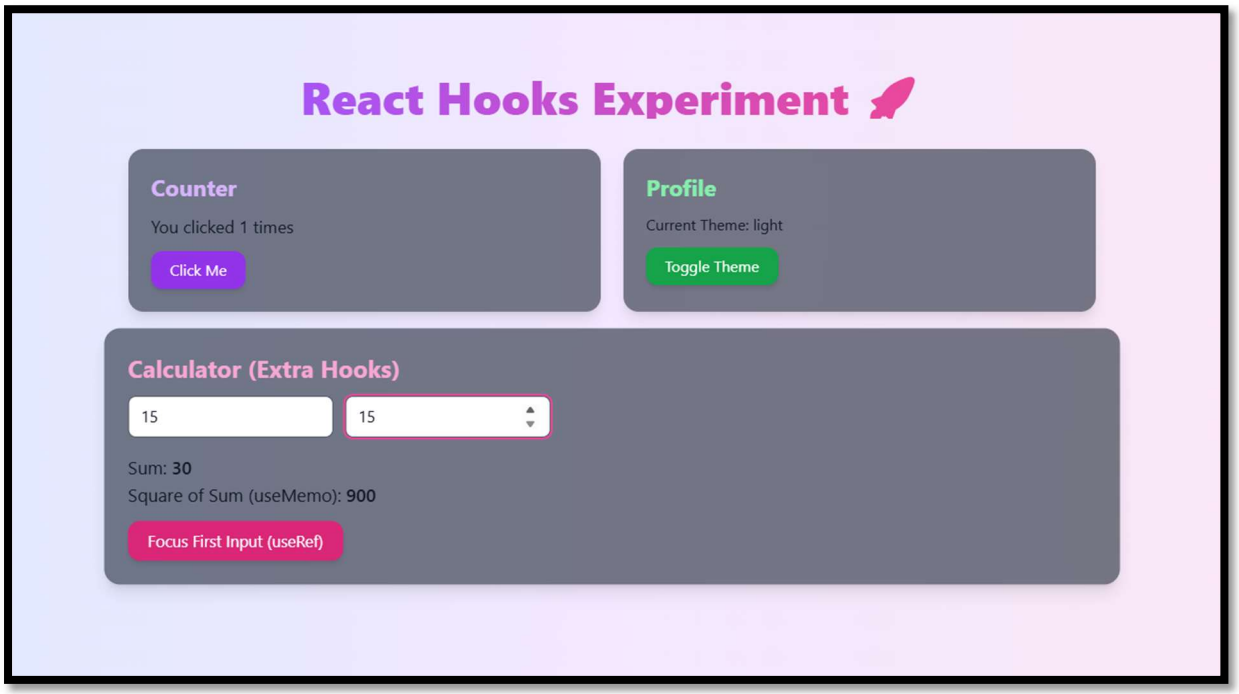


- After clicking on “Toggle Theme” button of “Profile” section (useContext is used) :-

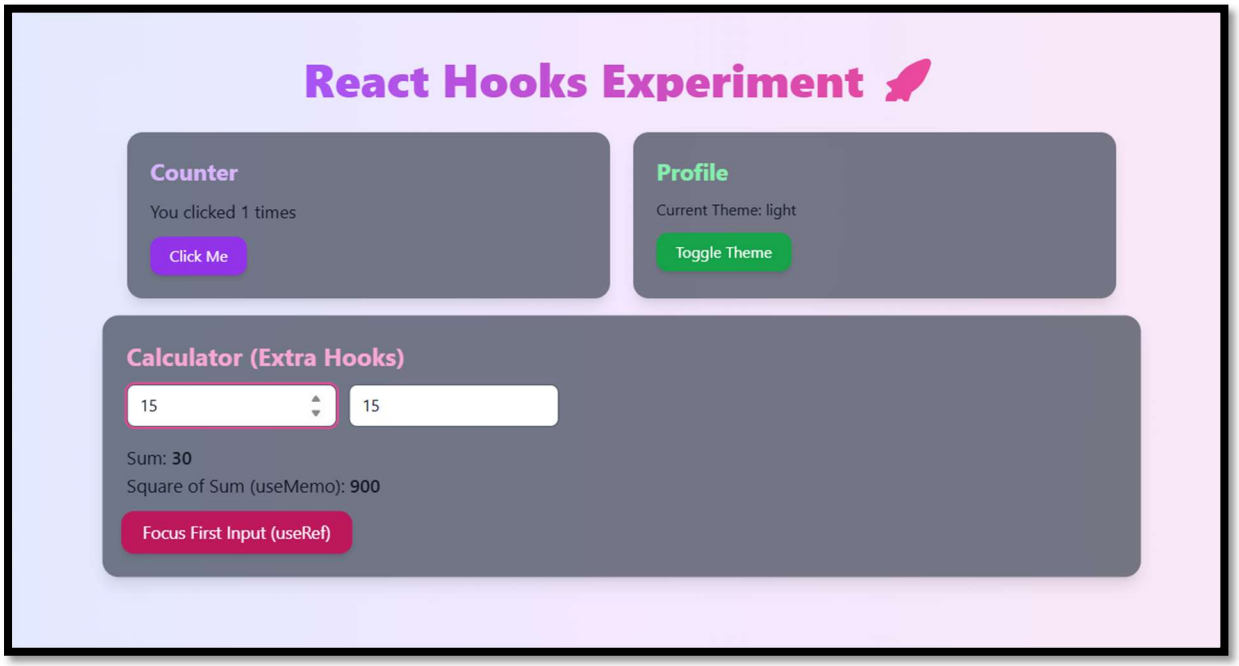


- Calculator Section (useMemo, useCallback, useRef, useState is used) :-





After clicking on “Focus First Input” button, it will shift the focus on first input field :-



30% Extra Content :-

Extra Hooks :-

1. **useMemo** (in Calculator)
    - Used to **memoize expensive calculations** like the squared sum of numbers.
    - Prevents unnecessary recomputation unless dependencies (num1, num2) change.

2. **useCallback** (in Calculator)
    - Wraps the addNumbers function to **prevent function re-creation** on every re-render.
    - Optimizes performance when passing functions as props or dependencies.
3. **useRef** (in Calculator)
    - Provides a reference to the **first input element**.
    - Demonstrates how refs can directly manipulate DOM elements (focus handling) without causing re-renders.

Conclusion :-

This experiment successfully demonstrates:

- The **core React Hooks (useState, useEffect, useContext)** and their practical applications.
- How to build a **custom hook (useLocalStorage)** to encapsulate logic and improve reusability.
- Managing **global state** (theme toggling) with useContext.
- It also explores **advanced hooks (useMemo, useCallback, useRef)** to optimize performance and handle direct DOM interactions.