Assignment

Learner Details

- **Name: Shrayanth S**
- **Enrollment Number: SU625MR011**
- **Batch / Class: June 2025 MERN**
- **Assignment: Toggle Theme using React**
- **Date of Submission: 11/08/2025**

Problem Solving Activity 1.1

1. Program Statement

The program allows a user to switch between **Light** and **Dark** themes in a React application.

- **Input:** A button click to toggle the theme.
- **Output:** The application's background and text colors change based on the current theme (Light or Dark).
  The program uses **React Context API** to share the theme state across multiple components (NavBar and ThemeToggle) without passing props manually.

---

2. Algorithm

- Create a ThemeContext using createContext() to store theme data and a toggle function.
- Create a ThemeProvider component to hold:
- theme state (light or dark)
- toggleTheme function to switch between themes.
- Wrap application components inside the ThemeProvider to share theme data.
- In NavBar component:
- Access the current theme from ThemeContext.
- Apply different background and text colors based on theme value.
- In ThemeToggle component:
- Access the toggleTheme function from ThemeContext.

- On button click, call toggleTheme to switch themes.
- Then we render both components and verify that the theme changes dynamically.

---

3. Pseudocode

START

CREATE ThemeContext

FUNCTION ThemeProvider(children):
  SET theme = "dark"
  FUNCTION toggleTheme():
    IF theme == "light" THEN
      theme = "dark"
    ELSE
      theme = "light"
  RETURN ThemeContext.Provider(value={theme, toggleTheme}) containing children

FUNCTION NavBar():
  GET theme from ThemeContext
  SET navbarStyle:
    background = light? "#f0f0f0" : "#333"
    color = light? "#000" : "#fff"
  DISPLAY nav bar with current theme name

FUNCTION ThemeToggle():
  GET toggleTheme from ThemeContext
  DISPLAY button "Toggle Theme"
  ON button click CALL toggleTheme

MAIN APP:

WRAP NavBar and ThemeToggle inside ThemeProvider

END

---

4. Program Code

ThemeProvider.jsx | eslint.config.js | App.jsx | ThemeToggle.jsx | NavBar.jsx

src > App.jsx > ...

```jsx
import React from 'react';
import './App.css'
import { ThemeProvider } from './Components/ThemeProvider'
import NavBar from './Components/NavBar';
import ThemeToggle from './Components/ThemeToggle';

function App() {

  return (
    <ThemeProvider>
      <div>
        <NavBar/>
        <ThemeToggle/>
      </div>
    </ThemeProvider>
  );
};

export default App;

```

ThemeProvider.jsx | eslint.config.js | App.jsx | ThemeToggle.jsx | NavBar.jsx

src > Components > NavBar.jsx > NavBar

```jsx
import { useContext } from "react"
import { ThemeContext } from "./ThemeProvider"


const NavBar = () =>{
    const{theme} = useContext(ThemeContext);

    const navbarStyle = {
        padding:"300px",
        backgroundColor:theme === "light" ? "#f0f0f0" : "#333",
        color:theme === "light"?"#000":"#fff",
    };

    return <nav style={navbarStyle}>Current Theme:{theme}</nav>;
};

export default NavBar;

```

ThemeProvider.jsx | eslint.config.js | App.jsx | ThemeToggle.jsx | NavBar.jsx

src > Components > ThemeProvider.jsx > ...

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


//Create Context
// eslint-disable-next-line react-refresh/only-export-components
export const ThemeContext = createContext();

//Create a provider component
export const ThemeProvider = ({children})=>{
    const [theme, setTheme] = useState("dark"); // Toggle Using Use State: ligh
//Toggle Function
const toggleTheme=()=>{
    setTheme((prevTheme)=>(prevTheme === "light"?"dark":"light"));
};

return(
    <div>
    <ThemeContext.Provider value = {{theme, toggleTheme}}>
        {children}
        </ThemeContext.Provider>
        </div>
)
}
```

ThemeProvider.jsx | eslint.config.js | App.jsx | ThemeToggle.jsx | NavBar.j

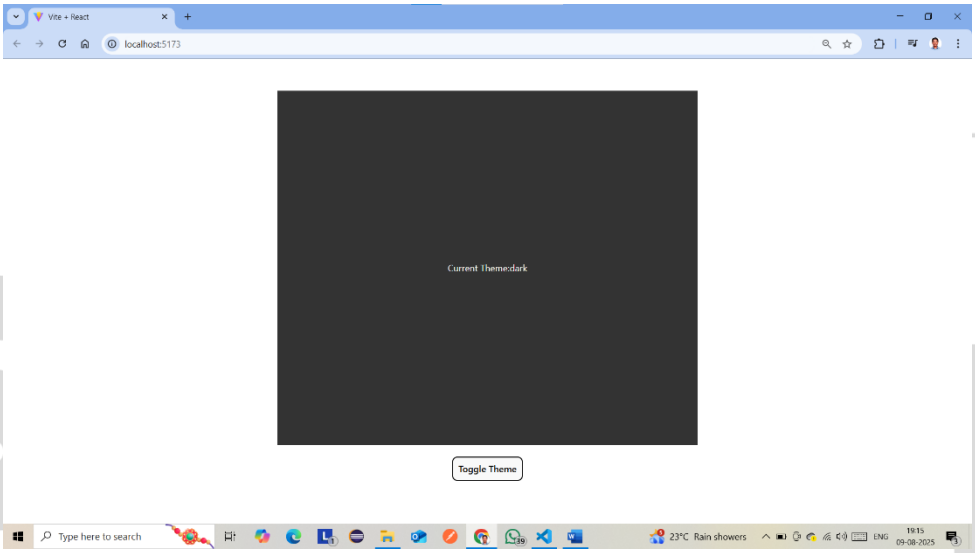
src > Components > ThemeToggle.jsx > ...

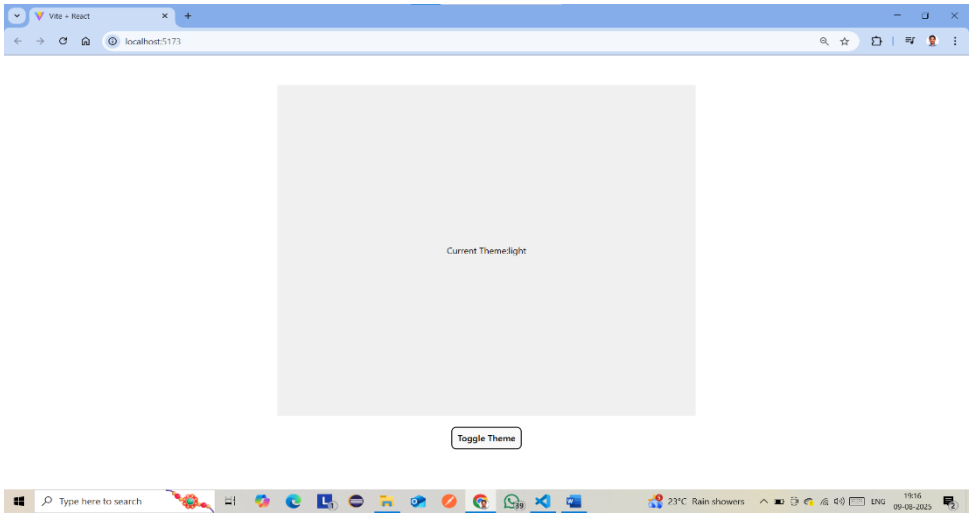
```jsx
import { useContext } from "react"
import { ThemeContext } from "./ThemeProvider"

const ThemeToggle = () => {
    const{toggleTheme} = useContext(ThemeContext);


return(
    <button onClick={toggleTheme} style={{margin:"20px", padding:"10px"}}>
        Toggle Theme
    </button>
);
};

export default ThemeToggle;
```

5. Test Cases

Test Case	Input	Expected Output	Result
1	Initial load	Dark theme with dark background and white text	Pass
2	Click Toggle once	Light theme with light background and black text	Pass

6. Screenshots of Output





7. Observation / Reflection

At first, I found it a bit tricky to set up the Context API and share the theme between different components. I also had to work carefully on the style so the background changed correctly. While doing this, I learned how to use the Context API to manage data for the whole app without passing it through props, and how to change styles using state. Next time, I plan to make the theme change look smoother with animations and save the theme in local storage so it stays the same even after reloading the page.