

Practical Number 2

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

THEORY:-

1. React Hooks

- Hooks are special functions introduced in React 16.8 that allow developers to use state and other React features without writing class components.
- **useState:** A hook that lets us add and update state variables in functional components.
- **useEffect:** A hook used for handling side effects such as fetching data, manipulating the DOM, or storing data in localStorage.
- **useContext:** A hook that allows components to consume values from a React Context without passing props down manually.

2. Context API

- The Context API provides a way to share data across the component tree without having to pass props at every level.
- It consists of three main parts:
 - `createContext()` → Creates a new context.
 - `Provider` → Supplies the context value to child components.
 - `useContext()` → Consumes the context value inside a component.

3. Custom Hooks

- A custom hook is a reusable function built using other hooks.
- It follows the “use” prefix naming convention (e.g., `useLocalStorage`).

- Custom hooks help reduce code duplication and improve readability when the same logic is required across multiple components.

4. **LocalStorage**

- LocalStorage is a web API that allows storing key-value pairs in the browser with no expiration date.
- Data stored in localStorage persists even after refreshing or closing the browser.
- It is commonly used for saving user preferences, themes, or temporary app data.

5. **Dark and Light Themes**

- Theme switching is a feature in modern applications that allows toggling between light and dark user interfaces.
- React, combined with Context API and TailwindCSS, provides a simple way to manage theme state.
- TailwindCSS includes built-in support for dark mode using the `dark:` modifier.

CODE:-

```
src > hooks > JS useLocalStorage.js > ...
1  import { useState, useEffect } from "react";
2
3  export default function useLocalStorage(key, initialValue) {
4    // Read from localStorage once on init
5    const [storedValue, setStoredValue] = useState(() => {
6      try {
7        const item = localStorage.getItem(key);
8        return item ? JSON.parse(item) : initialValue;
9      } catch (error) {
10        console.error(error);
11        return initialValue;
12      }
13    });
14
15    // Save to localStorage whenever it changes
16    useEffect(() => {
17      localStorage.setItem(key, JSON.stringify(storedValue));
18    }, [key, storedValue]);
19
20    return [storedValue, setStoredValue];
21  }
22
```

```
src > # App.css > .logo:hover
1  #root {
2    max-width: 1280px;
3    margin: 0 auto;
4    padding: 2rem;
5    text-align: center;
6  }
7
8  .logo {
9    height: 6em;
10   padding: 1.5em;
11   will-change: filter;
12   transition: filter 300ms;
13 }
14 .logo:hover {
15   filter: drop-shadow(0 0 2em #646cffaa);
16 }
17 .logo.react:hover {
18   filter: drop-shadow(0 0 2em #61dafbaa);
19 }
20
21 @keyframes logo-spin {
22   from {
23     transform: rotate(0deg);
24   }
25   to {
26     transform: rotate(360deg);
27   }
28 }
29
30 @media (prefers-reduced-motion: no-preference) {
31   a:nth-of-type(2) .logo {
32     animation: logo-spin infinite 20s linear;
33   }
34 }
35
36 .card {
37   padding: 2em;
38 }
39
40 .read-the-docs {
41   color: #888;
42 }
43
```

```

src > App.jsx > ...
1  import React, { useContext } from "react";
2  import { ThemeContext } from "../ThemeContext"; // Context for Dark/Light mode
3  import useLocalStorage from "../hooks/useLocalStorage"; // Custom hook for data persistence
4
5  export default function App() {
6    // Access theme state (light/dark) and toggle function from ThemeContext
7    const { theme, toggleTheme } = useContext(ThemeContext);
8
9    // Store teams and tasks in localStorage so they persist after refresh
10   const [teams, setTeams] = useLocalStorage("teams", []);
11   const [tasks, setTasks] = useLocalStorage("tasks", []);
12
13   // Local states for form inputs
14   const [teamName, setTeamName] = React.useState("");
15   const [teamMembers, setTeamMembers] = React.useState("");
16   const [taskName, setTaskName] = React.useState("");
17   const [selectedTeam, setSelectedTeam] = React.useState("");
18
19   // Add a new team with members
20   const handleAddTeam = () => {
21     if (!teamName.trim() || !teamMembers.trim()) return;
22     const membersArray = teamMembers.split(",").map(m => m.trim()); // split members by comma
23     setTeams([...teams, { name: teamName, members: membersArray }]);
24     setTeamName(""); // reset input
25     setTeamMembers(""); // reset input
26   };
27
28   // Add a new task assigned to a selected team
29   const handleAddTask = () => {
30     if (!taskName.trim() || !selectedTeam) return;
31     const team = teams.find(t => t.name === selectedTeam); // find selected team
32     setTasks([...tasks, { task: taskName, team }]);
33     setTaskName(""); // reset input
34     setSelectedTeam(""); // reset dropdown
35   };

```

```

37  return (
38    // Overall App Container with gradient + dark mode support
39    <div className="min-h-screen bg-gradient-to-br from-blue-50 to-indigo-100 dark:from-gray-800 dark:to-gray-900 transition-colors duration-700">
40
41      {/* HEADER: Title + Theme Toggle Switch */}
42      <div className="flex justify-between items-center px-6 py-4">
43        {/* App Title */}
44        <h1 className="text-4xl font-extrabold text-indigo-700 dark:text-indigo-300 drop-shadow-md transition-colors duration-700">
45          Collaboration App
46        </h1>
47
48        {/* Dark/Light Mode Toggle Switch */}
49        <label className="relative inline-flex items-center cursor-pointer">
50          <input
51            type="checkbox"
52            checked={theme === "dark"} // checkbox tracks theme
53            onChange={toggleTheme} // toggle on change
54            className="sr-only peer"
55          />
56          {/* Toggle UI: sliding circle */}
57          <div className="w-14 h-8 bg-gray-300 peer-focus:outline-none peer-focus:ring-2 peer-focus:ring-indigo-400 dark:peer-focus:ring-yellow-400 rounded-full peer
58            peer-checked:after:translate-x-6 after:content-[''] after:absolute after:top-[4px] after:left-[4px] after:bg-white after:border-gray-300 after:border after:
59            <span className="ml-3 text-sm font-medium text-gray-900 dark:text-gray-100 transition-colors duration-700">
60              {theme === "light" ? "Light Mode" : "Dark Mode"}
61            </span>
62          </label>
63        </div>

```

```

65  /* MAIN SECTION: Forms to Add Teams and Tasks */
66  <div className="grid grid-cols-1 md:grid-cols-2 gap-10 w-full max-w-6xl mx-auto px-6 mt-6">
67
68    /* FORM 1: Add Team */
69  <div className="bg-white dark:bg-gray-800 p-6 rounded-2xl shadow-lg hover:shadow-2xl transition-colors duration-700">
70    <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200 mb-4">
71      Add Team
72  </h2>
73    /* Team Name Input */
74  <input
75      type="text"
76      placeholder="Team Name"
77      value={teamName}
78      onChange={(e) => setTeamName(e.target.value)}
79      className="border border-gray-300 dark:border-gray-600 p-3 rounded-lg w-full mb-3 focus:ring-2 focus:ring-indigo-400 outline-none bg-white dark:bg-gray-
80    />
81    /* Members Input */
82  <input
83      type="text"
84      placeholder="Members (comma separated)"
85      value={teamMembers}
86      onChange={(e) => setTeamMembers(e.target.value)}
87      className="border border-gray-300 dark:border-gray-600 p-3 rounded-lg w-full mb-3 focus:ring-2 focus:ring-indigo-400 outline-none bg-white dark:bg-gray-
88    />
89    /* Add Team Button */
90  <button
91    onClick={handleAddTeam}
92    className="w-full bg-indigo-600 text-white py-2 px-4 rounded-lg hover:bg-indigo-700 transition-colors duration-700"
93  >
94      Add Team
95  </button>
96  </div>

```

```

98  /* FORM 2: Add Task */
99  <div className="bg-white dark:bg-gray-800 p-6 rounded-2xl shadow-lg hover:shadow-2xl transition-colors duration-700">
100 <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200 mb-4">
101   Add Task
102 </h2>
103 /* Task Name Input */
104 <input
105   type="text"
106   placeholder="Task Name"
107   value={taskName}
108   onChange={(e) => setTaskName(e.target.value)}
109   className="border border-gray-300 dark:border-gray-600 p-3 rounded-lg w-full mb-3 focus:ring-2 focus:ring-indigo-400 outline-none bg-white dark:bg-gray-
110 >
111 /* Select Team Dropdown */
112 <select
113   value={selectedTeam}
114   onChange={(e) => setSelectedTeam(e.target.value)}
115   className="border border-gray-300 dark:border-gray-600 p-3 rounded-lg w-full mb-3 bg-white dark:bg-gray-700 dark:text-white transition-colors duration-700"
116 >
117 <option value="">Select Team</option>
118 {teams.map((team, index) => (
119   <option key={index} value={team.name}>
120     {team.name}
121 </option>
122 ))}
123 </select>
124 /* Add Task Button */
125 <button
126   onClick={handleAddTask}
127   className="w-full bg-green-600 text-white py-2 px-4 rounded-lg hover:bg-green-700 transition-colors duration-700"
128 >
129     Add Task
130 </button>
131 </div>
132 </div>

```

```

134      /* TASK LIST SECTION */
135      <div className={bg-white dark:bg-gray-800 mt-10 p-6 rounded-2xl shadow-lg w-full max-w-6xl mx-auto transition-colors duration-700}>
136        <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200 mb-4">
137          Task List
138        </h2>
139
140        /* If no tasks yet */
141        {tasks.length === 0 ? (
142          <p className="text-gray-500 dark:text-gray-400">
143            No tasks added yet.
144          </p>
145        ) : (
146          /* Show all tasks with team + members */
147          <div className="space-y-4">
148            {tasks.map((t, index) => (
149              <div
150                key={index}
151                className="bg-gray-50 dark:bg-gray-700 p-4 rounded-lg shadow-sm transition-colors duration-700"
152              >
153                <p className="font-bold text-lg text-gray-700 dark:text-gray-100">
154                  {t.task}
155                </p>
156                <p className="text-sm text-gray-600 dark:text-gray-300 mb-2">
157                  Team: {t.team.name}
158                </p>
159                <ul className="list-disc list-inside text-gray-600 dark:text-gray-300 text-sm">
160                  {t.team.members.map((member, i) => (
161                    <li key={i}>{member}</li>
162                  ))}
163                </ul>
164              </div>
165            ))}
166          </div>
167        )}
168      </div>
169    </div>
170  );
171 }
172

```

```

src > # index.css
1  @tailwind base;
2  @tailwind components;
3  @tailwind utilities;
4

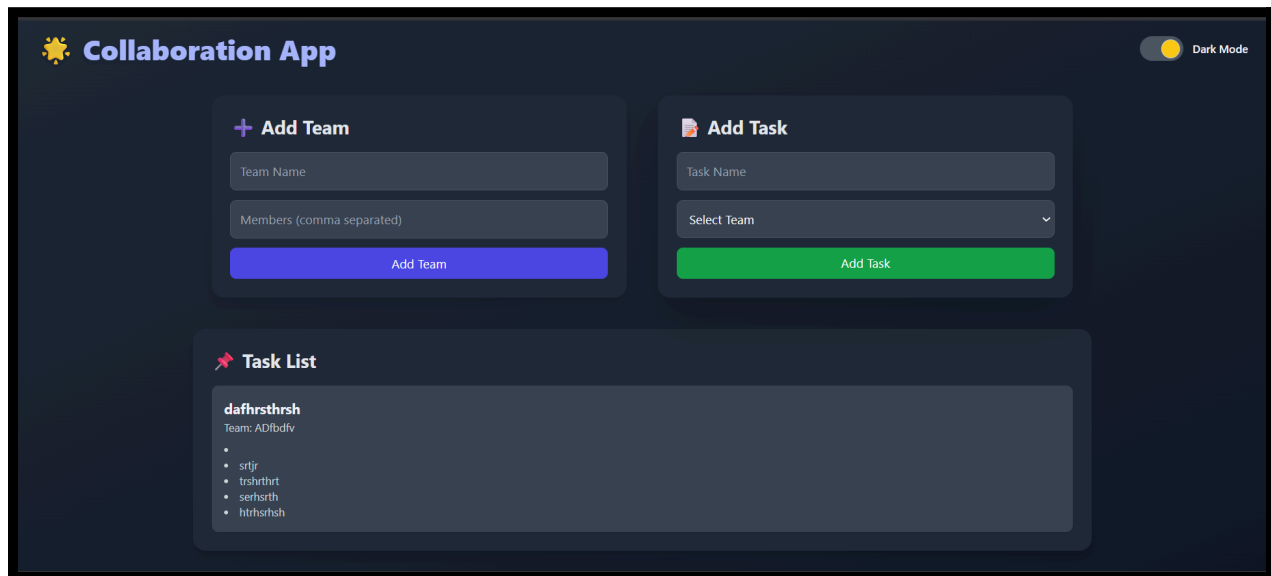
```

```

src > main.jsx
1  import React from "react";
2  import ReactDOM from "react-dom/client";
3  import App from "./App";
4  import ThemeProvider from "./ThemeContext"; // import ThemeProvider
5  import "./index.css";
6
7  ReactDOM.createRoot(document.getElementById("root")).render(
8    <React.StrictMode>
9      <ThemeProvider>
10        <App />
11      </ThemeProvider>
12    </React.StrictMode>
13  );
14

```

```
src > ThemeContext.jsx > ...
6  const ThemeProvider = ({ children }) => {
7    // ✅ Persist theme in localStorage
8    const [theme, setTheme] = useLocalStorage("theme", "light");
9
10   // Toggle function
11   const toggleTheme = () => {
12     setTheme(theme === "light" ? "dark" : "light");
13   };
14
15   // ✅ Apply Tailwind dark mode class to <html>
16   React.useEffect(() => {
17     const root = document.documentElement;
18     if (theme === "dark") {
19       root.classList.add("dark");
20     } else {
21       root.classList.remove("dark");
22     }
23   }, [theme]);
24
25   return (
26     <ThemeContext.Provider value={{ theme, toggleTheme }}>
27       {children}
28     </ThemeContext.Provider>
29   );
30 };
31
32 export default ThemeProvider;
33
```

OUTPUT:-

The screenshot displays the 'Collaboration App' interface in dark mode. At the top left is a sun icon and the app title 'Collaboration App'. At the top right is a 'Dark Mode' toggle switch. The interface is divided into three main sections:

- + Add Team:** Contains a 'Team Name' input field, a 'Members (comma separated)' input field, and a blue 'Add Team' button.
- + Add Task:** Contains a 'Task Name' input field, a 'Select Team' dropdown menu, and a green 'Add Task' button.
- Task List:** Features a red pin icon and the title 'Task List'. It shows a team entry for 'dafhrsthrsh' with the team ID 'ADfbdfv'. Below this, a list of members is displayed:
 - srljr
 - trshthrt
 - serhsrth
 - hthsrsh

CONCLUSION:-

The experiment demonstrated the use of React Hooks, Context API, and localStorage. The concepts of theme switching, persistent data storage, and custom hooks were successfully studied and implemented.