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 > ...

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

export default function useLocalStorage(key, initialValue) {
  // Read from localStorage once on init
  const [storedValue, setStoredValue] = useState(() => {
    try {
      const item = localStorage.getItem(key);
      return item ? JSON.parse(item) : initialValue;
    } catch (error) {
      console.error(error);
      return initialValue;
    }
  });

  // Save to localStorage whenever it changes
  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(storedValue));
  }, [key, storedValue]);

  return [storedValue, setStoredValue];
}
```

src > # App.css > .logo:hover

```
#root {
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
  text-align: center;
}

.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: filter 300ms;
}
.logo:hover {
  filter: drop-shadow(0 0 2em #646cffaa);
}
.logo.react:hover {
  filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

@media (prefers-reduced-motion: no-preference) {
  a:nth-of-type(2) .logo {
    animation: logo-spin infinite 20s linear;
  }
}

.card {
  padding: 2em;
}

.read-the-docs {
  color: #888;
}
```

src > App.jsx > ...

```jsx
import React, { useContext } from "react";
import { ThemeContext } from "./ThemeContext";  // Context for Dark/Light mode
import useLocalStorage from "./hooks/useLocalStorage"; // Custom hook for data persistence

export default function App() {
  // Access theme state (light/dark) and toggle function from ThemeContext
  const { theme, toggleTheme } = useContext(ThemeContext);

  // Store teams and tasks in localStorage so they persist after refresh
  const [teams, setTeams] = useLocalStorage("teams", []);
  const [tasks, setTasks] = useLocalStorage("tasks", []);

  // Local states for form inputs
  const [teamName, setTeamName] = React.useState("");
  const [teamMembers, setTeamMembers] = React.useState("");
  const [taskName, setTaskName] = React.useState("");
  const [selectedTeam, setSelectedTeam] = React.useState("");

  // Add a new team with members
  const handleAddTeam = () => {
    if (!teamName.trim() || !teamMembers.trim()) return;
    const membersArray = teamMembers.split(",").map((m) => m.trim()); // split members by comma
    setTeams([...teams, { name: teamName, members: membersArray }]);
    setTeamName("");     // reset input
    setTeamMembers("");  // reset input
  };

  // Add a new task assigned to a selected team
  const handleAddTask = () => {
    if (!taskName.trim() || !selectedTeam) return;
    const team = teams.find((t) => t.name === selectedTeam); // find selected team
    setTasks([...tasks, { task: taskName, team }]);
    setTaskName("");      // reset input
    setSelectedTeam("");  // reset dropdown
  };
```

```jsx
  return (
    // Overall App Container with gradient + dark mode support
    <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">

      {/* HEADER: Title + Theme Toggle Switch */}
      <div className="flex justify-between items-center px-6 py-4">
        {/* App Title */}
        <h1 className="text-4xl font-extrabold text-indigo-700 dark:text-indigo-300 drop-shadow-md transition-colors duration-700">
          🌟 Collaboration App
        </h1>

        {/* Dark/Light Mode Toggle Switch */}
        <label className="relative inline-flex items-center cursor-pointer">
          <input
            type="checkbox"
            checked={theme === "dark"}   // checkbox tracks theme
            onChange={toggleTheme}       // toggle on change
            className="sr-only peer"
          />
          {/* Toggle UI: sliding circle */}
          <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
            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:
          <span className="ml-3 text-sm font-medium text-gray-900 dark:text-gray-100 transition-colors duration-700">
            {theme === "light" ? "Light Mode" : "Dark Mode"}
          </span>
        </label>
      </div>
```

```
{/* MAIN SECTION: Forms to Add Teams and Tasks */}
<div className="grid grid-cols-1 md:grid-cols-2 gap-10 w-full max-w-6xl mx-auto px-6 mt-6">

  {/* FORM 1: Add Team */}
  <div className="bg-white dark:bg-gray-800 p-6 rounded-2xl shadow-lg hover:shadow-2xl transition-colors duration-700">
    <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200 mb-4">
      ➕ Add Team
    </h2>
    {/* Team Name Input */}
    <input
      type="text"
      placeholder="Team Name"
      value={teamName}
      onChange={(e) => setTeamName(e.target.value)}
      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-
    />
    {/* Members Input */}
    <input
      type="text"
      placeholder="Members (comma separated)"
      value={teamMembers}
      onChange={(e) => setTeamMembers(e.target.value)}
      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-
    />
    {/* Add Team Button */}
    <button
      onClick={handleAddTeam}
      className="w-full bg-indigo-600 text-white py-2 px-4 rounded-lg hover:bg-indigo-700 transition-colors duration-700"
    >
      Add Team
    </button>
  </div>
```

```
  {/* FORM 2: Add Task */}
  <div className="bg-white dark:bg-gray-800 p-6 rounded-2xl shadow-lg hover:shadow-2xl transition-colors duration-700">
    <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200 mb-4">
      📝 Add Task
    </h2>
    {/* Task Name Input */}
    <input
      type="text"
      placeholder="Task Name"
      value={taskName}
      onChange={(e) => setTaskName(e.target.value)}
      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-
    />
    {/* Select Team Dropdown */}
    <select
      value={selectedTeam}
      onChange={(e) => setSelectedTeam(e.target.value)}
      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-7
    >
      <option value="">Select Team</option>
      {teams.map((team, index) => (
        <option key={index} value={team.name}>
          {team.name}
        </option>
      ))}
    </select>
    {/* Add Task Button */}
    <button
      onClick={handleAddTask}
      className="w-full bg-green-600 text-white py-2 px-4 rounded-lg hover:bg-green-700 transition-colors duration-700"
    >
      Add Task
    </button>
  </div>
</div>
```

```jsx
      {/* TASK LIST SECTION */}
      <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">
        <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200 mb-4">
          📌 Task List
        </h2>

        {/* If no tasks yet */}
        {tasks.length === 0 ? (
          <p className="text-gray-500 dark:text-gray-400">
            No tasks added yet.
          </p>
        ) : (
          /* Show all tasks with team + members */
          <div className="space-y-4">
            {tasks.map((t, index) => (
              <div
                key={index}
                className="bg-gray-50 dark:bg-gray-700 p-4 rounded-lg shadow-sm transition-colors duration-700"
              >
                <p className="font-bold text-lg text-gray-700 dark:text-gray-100">
                  {t.task}
                </p>
                <p className="text-sm text-gray-600 dark:text-gray-300 mb-2">
                  Team: {t.team.name}
                </p>
                <ul className="list-disc list-inside text-gray-600 dark:text-gray-300 text-sm">
                  {t.team.members.map((member, i) => (
                    <li key={i}>{member}</li>
                  ))}
                </ul>
              </div>
            ))}
          </div>
        )}
      </div>
    </div>
  );
}
```

src > # index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

src > main.jsx

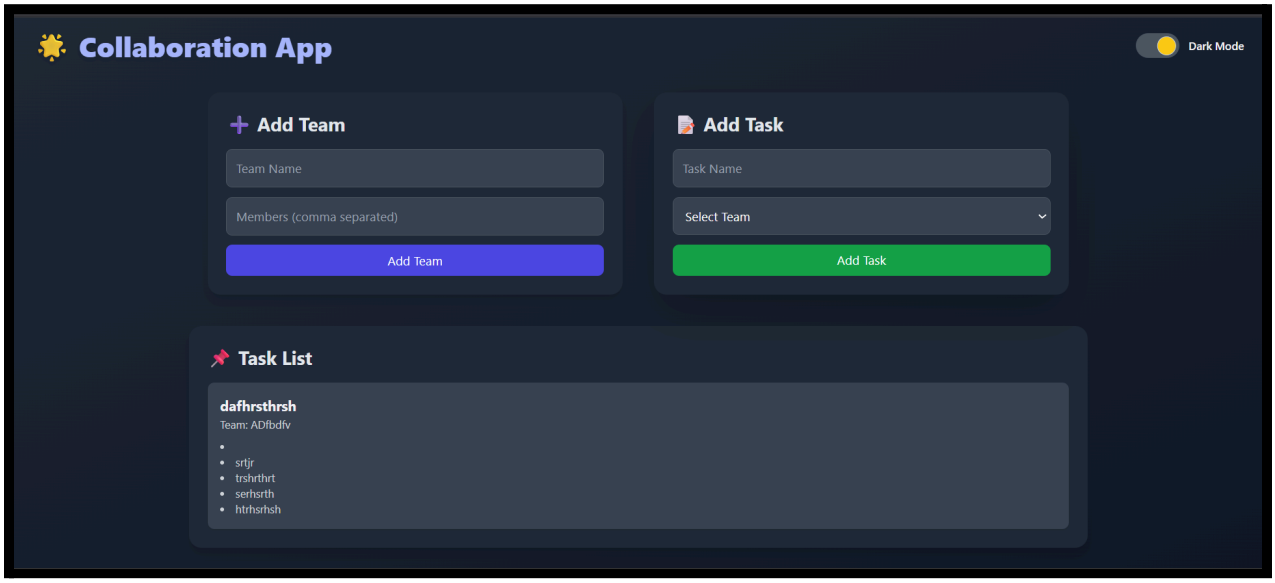
```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import ThemeProvider from "./ThemeContext"; // ✅ import ThemeProvider
import "./index.css";

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

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:-



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.