FocusTasks Solution

---

1. Project Setup

Install required packages:

```
npm create vite@latest focustasks --template react
npm install react-router-dom
```

Folder structure:

```
src/
  components/
  context/
  reducers/
  pages/
  App.jsx
  main.jsx
```

---

2. ThemeContext (useContext)

ThemeContext.jsx:

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

```
export const ThemeContext = createContext();


export function ThemeProvider({ children }) {

  const [theme, setTheme] = useState("light");

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


  return (

    <ThemeContext.Provider value={{ theme, toggleTheme }}>

      <div className={theme}>

        {children}

      </div>

    </ThemeContext.Provider>

  );

}


Usage: wrap <App /> in ThemeProvider.


------------------------------------------------------------

3. taskReducer (useReducer)

------------------------------------------------------------

taskReducer.js:


export function taskReducer(state, action) {
```

```
  switch (action.type) {
    case "ADD_TASK":
      return [...state, action.payload];

    case "TOGGLE_TASK":
      return state.map(task =>
        task.id === action.payload
          ? { ...task, isCompleted: !task.isCompleted }
          : task
      );

    case "DELETE_TASK":
      return state.filter(task => task.id !== action.payload);

    default:
      return state;
  }
}
```

---

4. Task State Provider

Inside App.jsx:

```
import { useReducer, useEffect } from "react";
import { taskReducer } from "./reducers/taskReducer";

function App() {
  const [tasks, dispatch] = useReducer(taskReducer, []);

  useEffect(() => {
    const stored = localStorage.getItem("tasks");
    if (stored) {
      dispatch({ type: "LOAD_TASKS", payload: JSON.parse(stored)
});
    }
  }, []);

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

  return (...routes...);
}
```

---

5. Tasks Page (Form + useRef)

Tasks.jsx:

```
import { useState, useRef, useContext } from "react";

function Tasks({ tasks, dispatch }) {
  const [title, setTitle] = useState("");
  const inputRef = useRef(null);

  useEffect(() => {
    inputRef.current.focus();
  }, []);

  const addTask = (e) => {
    e.preventDefault();
    if (!title.trim()) return;

    const newTask = {
      id: Date.now(),
      title,
      isCompleted: false,
    };

    dispatch({ type: "ADD_TASK", payload: newTask });
    setTitle("");
  };
```

```
  return (
    <div>
      <form onSubmit={addTask}>
        <input ref={inputRef} value={title} onChange={e =>
setTitle(e.target.value)} />
        <button>Add</button>
      </form>

      <ul>
        {tasks.map(task => (
          <li key={task.id}>
            <input type="checkbox"
              checked={task.isCompleted}
              onChange={() => dispatch({ type: "TOGGLE_TASK",
payload: task.id })}
            />
            {task.title}
            <button onClick={() => dispatch({ type:
"DELETE_TASK", payload: task.id })}>
              Delete
            </button>
          </li>
        ))}
      </ul>
```

```
    </div>
  );
}

------------------------------------------------------------
6. Dashboard (API fetch)
------------------------------------------------------------
Dashboard.jsx:

import { useEffect, useState } from "react";

function Dashboard({ tasks }) {
  const [quote, setQuote] = useState("");
  const [loading, setLoading] = useState(false);

  const fetchQuote = async () => {
    try {
      setLoading(true);
      const res = await fetch("https://api.quotable.io/random");
      const data = await res.json();
      setQuote(data.content);
    } catch (err) {
      setQuote("Error loading quote");
    } finally {
```

```
      setLoading(false);
    }
  };

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

  const completed = tasks.filter(t => t.isCompleted).length;
  const pending = tasks.length - completed;

  return (
    <div>
      <h2>Dashboard</h2>
      <p>Pending tasks: {pending}</p>
      <p>Completed tasks: {completed}</p>

      <h3>Quote of the Moment:</h3>
      {loading ? "Loading..." : quote}
      <button onClick={fetchQuote}>Refresh Quote</button>
    </div>
  );
}
```

---

7. Settings Page (Theme Toggle)

Settings.jsx:

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

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

  return (
    <div>
      <h2>Settings</h2>
      <p>Current Theme: {theme}</p>
      <button onClick={toggleTheme}>Toggle Theme</button>
    </div>
  );
}
```

---

8. Router Setup

App.jsx:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Dashboard from "./pages/Dashboard";
import Tasks from "./pages/Tasks";
import Settings from "./pages/Settings";


return (
  <BrowserRouter>
    <Navbar />
    <Routes>
      <Route path="/" element={<Dashboard tasks={tasks} />} />
      <Route path="/tasks" element={<Tasks tasks={tasks}
dispatch={dispatch} />} />
      <Route path="/settings" element={<Settings />} />
    </Routes>
  </BrowserRouter>
);
```