DAY 24 – useState(), useEffect()

---

1. Weather Dashboard App

Core Concepts: Data Fetching, Conditional Rendering, State Updates, Cleanup

Real-Life Use Case:

A weather dashboard that fetches real-time weather for a searched city.

Requirements:

1. Use useState to manage:
    - city (user input)
    - weatherData (API response)
    - loading (boolean)
    - error (string)
2. Use useEffect to:
    - Fetch weather when city changes
    - Show loading spinner before data loads
3. Include:
    - Input field for city name (controlled component)
    - Display weather info (temperature, humidity, description)
    - Handle errors (invalid city names)
    - Cleanup effect if API call is canceled midway (use AbortController)
4. Bonus:
    - Automatically fetch default city weather on mount (e.g., “Chennai”)

Concepts Reinforced:

useState, useEffect (mount, dependency, cleanup), controlled input, async effect handling.

---

2. Pomodoro Timer App

Core Concepts: Timer Management, Cleanup, Conditional State

Real-Life Use Case:

A productivity timer app with work/break cycles.

□ Requirements:

1. Use useState for:
    - seconds, isRunning, mode ("work" or "break")
2. Use useEffect to:
    - Start a timer using setInterval when isRunning is true
    - Automatically switch to break mode when work timer ends
    - Cleanup interval on unmount or when paused
3. UI Features:
    - Start, Pause, Reset buttons
    - Show remaining time and mode
    - Dynamic background color (red for work, green for break)

Concepts Reinforced:

Timers, cleanup functions, dependency arrays, conditional rendering.

---

3. Live Cryptocurrency Tracker

Core Concepts: Polling API, State Comparison, Conditional Re-render

Real-Life Use Case:

Displays live cryptocurrency prices and updates automatically every 10 seconds.

□ Requirements:

1. Use useState to store:
    - cryptoList (fetched array of coins)
    - lastUpdated timestamp
    - error message
2. Use useEffect to:
    - Fetch crypto prices from an API like CoinGecko every 10 seconds
    - Cleanup the interval on unmount
3. UI Features:
    - Display coin name, symbol, price, 24h change

- Highlight price changes with colors (green ↑, red ↓)
- “Refresh Now” button to manually re-fetch

Concepts Reinforced:

useEffect intervals, cleanup, re-render control, dependency handling, API polling.

---

4. Online Status Tracker

Core Concepts: Browser Events, Cleanup, Conditional Rendering

Real-Life Use Case:

Tracks the user’s online/offline status and shows it in the UI.

□ Requirements:

1. Use useState for:
   - isOnline (boolean)
   - lastChanged (time)
2. Use useEffect to:
   - Listen to window.addEventListener('online') and 'offline'
   - Update isOnline state accordingly
   - Cleanup event listeners on unmount
3. UI Features:
   - Green indicator when online, red when offline
   - Show last status change time using new Date().toLocaleTimeString()
   - Optional: Play sound or toast notification when status changes

Concepts Reinforced:

Event listener setup and cleanup, real-time reactivity, conditional DOM updates.

---

5. To-Do List with Persistent Storage

Core Concepts: State Management, LocalStorage Sync, Controlled Inputs

Real-Life Use Case:

A task manager that saves data in localStorage.

□ Requirements:

1. Use useState to manage:
    - tasks (array of {id, title, completed})
    - inputValue
2. Use useEffect to:
    - Load saved tasks from localStorage on mount
    - Save updated tasks back to localStorage whenever it changes
3. UI Features:
    - Add, delete, mark-complete buttons
    - Filter tasks (All / Completed / Pending)
    - Display task count
4. Bonus:
    - Toast notification on add/delete
    - Persist theme (dark/light) in localStorage using another state

Concepts Reinforced:

State persistence, side-effects based on dependencies, cleanup (optional event listeners), controlled input.
