Day 15 – React Hooks & State Management

Challenge 5: React Context API

User Story:
As a user, I want my preferred theme (light/dark) to apply across all pages without passing props manually.

Problem Statement:
Implement a **ThemeContext** using the Context API.
Provide theme values at the top-level and consume them using `useContext()` in multiple child components.

Expected Outcome:

- Create ThemeContext.Provider in App.js.
- Switch between dark/light modes with a toggle.

Bonus:
Store theme preference in `localStorage` and persist it.

Challenge 6: Progressive Web App (PWA)

User Story:
As a frequent visitor, I want the app to work offline and be installable on my device.

Problem Statement:
Convert your existing React project into a **Progressive Web App**.
Enable offline caching using a **service worker** and add a **manifest.json**.

Expected Outcome:

- Lighthouse PWA score above 90.
- App installable from the browser.

Bonus:
Show an “Offline” banner when network connectivity is lost.

Challenge 7: React Hooks

User Story:
As a fitness enthusiast, I want my “Workout Tracker” to automatically track sets and rest time using React Hooks.

Problem Statement:
Create a functional component that uses `useState`, `useEffect`, and `useRef` to manage sets, timers, and auto-reset logic.

Expected Outcome:

- Timer updates every second using `useEffect`.
- Stop timer with cleanup using `clearInterval`.

Bonus:
Add a custom hook `useTimer()` to abstract timer logic.

Challenge 8: Redux / ngrx / ngrx-effects

User Story:
As an admin, I want to view and update product data globally across pages without prop drilling.

Problem Statement:
Implement global **state management** for product data using **Redux Toolkit**.
Use reducers and actions to fetch data from a mock API and update it.

Expected Outcome:

- Use `configureStore()`, `createSlice()`, and `useSelector()`.
- Dispatch `fetchProducts()` and `updateProduct()` actions.

Bonus:
Integrate `redux-thunk` or `ngrx-effects` for handling asynchronous calls.