Assignment 2 — ShopEase Wishlist & Theme Enhancements

Objective:

Enhance your existing **ShopEase e-commerce application** to implement a **Wishlist feature** and a **Light/Dark Theme switcher**, demonstrating React concepts like **event handling**, **conditional rendering**, **lists & keys**, **Context API**, and **modular styling**.

Tasks (requirements)

1. **Product List Page**
    - Create a Product List page that displays multiple products using the map() function.
    - Each product should be rendered using a reusable ProductCard component.
    - Ensure every list item uses a unique key (e.g., product ID).

2. **Add to Wishlist Button**
    - Add a button labeled **"Add to Wishlist"** on each ProductCard.
    - On click, use **event handling** to toggle the wishlist state (add/remove).
    - Change the button text dynamically — "Add to Wishlist" → "Remove from Wishlist."

3. **Conditional Rendering**
    - When a product is added to the wishlist, show a visible label like **"In Wishlist"**.
    - Use conditional rendering to change button styles or text based on wishlist state.

4. **Wishlist Management using Context API**
    - Create a **WishlistContext** to store and manage wishlist data globally.
    - Provide methods to **add**, **remove**, and **check** products in the wishlist.
    - Avoid prop drilling — access context directly inside ProductCard components.

5. **Styling with CSS Modules**
    - Style the ProductCard using a **CSS Module** file (ProductCard.module.css).
    - Maintain consistent layout, spacing, and responsive design.

6. **Theme Switcher (Light/Dark Mode)**
    - Implement a **ThemeContext** using the Context API.
    - Add a toggle button to switch between **Light** and **Dark** themes.
    - Apply global theme styles across the app (e.g., background, text color, button color).

Implementation notes (recommended)

- Use **functional components** and **React hooks** (useState, useContext, etc.).
- Organize code using folders: components, contexts, data, styles.
- Keep components small, reusable, and well-structured.
- Use a **Vite** setup (preferred) or **Create React App**.
- Styling can use **CSS Modules**, **Tailwind**, or plain CSS — keep it clean and consistent.
- No backend is required — manage all data on the frontend.

Deliverables (what to submit)

1. **README.md** file containing:
   o Project setup and run instructions (e.g., npm install, npm run dev).
   o Notes on implementation decisions and theme handling.
   o Description of Context usage for Wishlist and Theme.
2. **ZIP File of the Project** uploaded before the deadline.

Grading Rubric (suggested)

Functionality (60%)

- Product list renders correctly using map() *(10%)*
- "Add to Wishlist" button toggles state properly *(20%)*
- Conditional label "In Wishlist" displays correctly *(10%)*
- Context API used effectively for global wishlist state *(20%)*

Code Quality & Structure (25%)

- Clean component separation and folder organization
- Proper use of hooks and context
- Readable, maintainable, and modular code

Styling & Presentation (15%)

- ProductCard styled with CSS Modules
- Light/Dark theme implemented using Context API
- Neat, consistent, and responsive UI

Submission Instructions

• Upload the **ZIP file** of your project before the deadline.
• Ensure your **README.md** contains setup instructions and key notes.