Frontend Assignment-1 (ReactJS/NextJS)

🔧 Prerequisite

1. All assignments must be implemented using **Next.js 15**.

2. You may choose **App Router** or **Pages Router** based on the problem statement and your preference.

🚀 1. Blog Application with Comments and Likes

🧩 Objective:
Develop a feature-rich blog application using **Next.js 15** (App or Pages Router). This project focuses on real-world frontend skills including API integration, dynamic rendering, and user interaction handling. The application should simulate a blogging platform where users can browse posts, interact with them through likes, and engage in discussions via nested comment threads.

📋 Requirements:

- **Fetch Blog Data:**
  Retrieve blog posts from a public mock API (such as JSONPlaceholder or mockapi.io). Each post should include a title, a short content snippet (e.g., first 100 characters), and a "Read More" button that navigates to a detailed post page.

- **Post Detail Page:**
  On clicking "Read More", navigate to a dynamic route displaying the full blog content. Ensure the route is built dynamically using `getStaticPaths`/`getServerSideProps` (Pages Router) or `generateStaticParams`/`fetch` (App Router).

- **Like Functionality:**
  Add a like button to each post. Track the number of likes and allow users to toggle like/unlike. You may persist likes either in local state (like `useState`, `useReducer`, or Zustand) or by simulating API updates using a mock endpoint.

- **Nested Comments System:**
  Implement a comment section for each blog post. Users should be able to add new comments and reply to existing ones, forming a threaded discussion. You should design the UI to clearly show parent-child relationships among comments.

🧠 **Skills Evaluated:**

- REST API integration and data fetching.
- Dynamic routing and page generation in Next.js 15.
- State management for UI interactivity.
- Conditional rendering and dynamic list handling.
- Building reusable and nested UI components.
- UX considerations for comment threading and navigation.

TLDR:

- Fetch blog posts from a mock API (e.g., JSONPlaceholder or mockapi.io).
- Display posts with title, content snippet, and a "Read More" button.
- Allow users to like a post (store in local state or mock API).
- Add a comment section with nested replies (threaded).

✨ Bonus Optional Features to Suggest:

- Light/Dark Theme Toggle
- Responsiveness
- Custom loading skeletons.