Technical Challenge – Local-First Next.js App

Time Limit: You have up to 3 hours to complete this challenge from the moment you receive the e-mail.

This is a timed exercise. Please respect the time limit and submit whatever you have at the end of the 3-hour window, even if incomplete. Partial implementations are acceptable.

You are asked to build a small local-first application using **Next.js**, with **Zustand** for state management, **Dexie.js** (IndexedDB) for local caching, and **Tailwind CSS** for responsive UI.

Focus on simplicity, clarity, and offline resilience.

Requirements

1. Data Fetching

- On initial load, fetch user data from: https://randomuser.me/api/?page=1&results=10
- Client-side pagination.

2. Local Caching (IndexedDB)

- After fetching, **store the user data in IndexedDB** using Dexie.js.
- On subsequent visits or if the user is **offline** or the API **fails**, load and display data from the **local IndexedDB cache**.

3. Global State (Zustand)

- Use Zustand to manage:
    - Loading states
    - Pagination states
    - Error or offline states
    - Mark users as favorite, and persist this also in IndexedDB
    - Ensure components reactively update with store changes.

4. Responsive UI (Tailwind CSS)

- Display users in a simple, responsive layout (e.g., cards or list).
- Use **Tailwind utility classes** to organize styles cleanly.

5. Graceful Degradation

- If the API is unreachable, show a **fallback UI** (e.g., “You are offline”).
- Continue showing the **most recent cached data** from IndexedDB.

Deliverables

Your project should:

- Be a working **Next.js app** using:
  - **Zustand** for global state
  - **Dexie.js** for IndexedDB caching
  - **Tailwind CSS** for styling
- Handle **offline fallback** gracefully

README.md

Include a short README that explains:

- How to install dependencies (`npm install`)
- How to run the project (`npm run dev`)
- How to simulate offline/failure scenarios
- Known issues or limitations
- What you would improve with more time

Nice to Have

These are not required but will help demonstrate extra initiative or experience:

1. **Manual “Go Offline” toggle** to simulate fallback behavior
2. **Search or filter users** by name/email
3. **Order users by fields** (name, email, etc)
4. **Dark mode support** using Tailwind’s `dark:` utilities
5. **Basic test coverage** with Jest or React Testing Library

Pick any (or none). If time is limited, mention in the README what you would've done next.

Submission Guidelines

- Push your code to a **public GitHub repository**, or provide a zipped folder.
- Ensure your commit history reflects your work.
- Share the link or zip via email with any final notes.

Final Notes

- No need to configure a full PWA.
- You may use LLMs or AI tools as long as you understand the output.
- Simplicity, clarity, and developer experience matter most.

Good luck, and have fun! 🚀