Take-Home Assignment: Offline-First Notes App with Sync

🚀 Project Overview

You are tasked with building a **Markdown-based Notes Application** that works smoothly **offline** and syncs data **when online**. The app should persist notes locally using IndexedDB, allow users to create/edit/delete notes, and sync changes to a mock backend API once internet connectivity is restored.

This project is designed to evaluate your React skills, ability to manage offline-first applications, handle asynchronous syncing, and make architectural decisions under real-world constraints.

🎯 Requirements

Core Features

1. **Note Creation & Editing**
   - Users can create new notes.
   - Notes have a **title** and **content** (Markdown supported).
   - Editing updates notes instantly and autosaves changes.
2. **Offline Persistence**
   - Notes are stored in **IndexedDB** to enable full offline usage.
   - Users can create/edit/delete notes while offline.
   - Data persists across browser refreshes and restarts.
3. **Syncing**
   - When the app detects that it's online, it should sync local changes with the mock backend.

- Sync both new notes and updates/deletions.
- Implement a basic conflict resolution strategy (e.g., client-wins or last-write-wins).
- Show syncing status per note (e.g., “Unsynced”, “Syncing…”, “Synced”, “Error”).

4. **Connectivity Awareness**
   - Detect online/offline status using the browser’s API.
   - UI should clearly indicate connection status and sync progress.
5. **Note Listing & Searching**
   - Display a list of notes sorted by last updated time.
   - Provide a search bar to filter notes by title or content.
6. **User Experience**
   - Autosave note content during editing with debounce (e.g., 500ms).
   - Responsive and accessible UI.

🧩 Data Model

Each note should have at least the following structure:

```
{
  id: string;              // Unique UUID
  title: string;           // Note title
  content: string;         // Markdown text content
  updatedAt: string;       // ISO timestamp of last update
  synced: boolean;         // Whether note is synced with backend
}
```

📡 Mock API

Use any mock backend API tool (e.g., MockAPI, json-server) with the following REST endpoints:

- `GET /notes` — fetch all notes
- `POST /notes` — create a new note
- `PUT /notes/:id` — update a note
- `DELETE /notes/:id` — delete a note

⚙️ Technical Suggestions

Concern	Suggested Tools/Libraries
React & State Management	React (hooks, context, or state libs)
IndexedDB Integration	`idb`, `dexie`
Markdown Editing	`react-markdown`, `react-mde`
Offline Detection	`navigator.onLine`, `online/offline` event listeners
HTTP Client	Fetch API or Axios
Styling	Your choice (CSS Modules, Tailwind, etc.)

🕵️ Evaluation Criteria

You will be evaluated on:

- **Correctness & Completeness:** All core features implemented correctly.
- **Architecture & Code Quality:** Modular, clean, and maintainable code.
- **Offline & Sync Logic:** Robustness and reliability of offline storage and syncing.
- **UI/UX:** User-friendly interface with appropriate feedback (loading states, error handling).

- **Readme & Documentation:** Clear instructions and explanations.
- **Bonus:** Handling edge cases, conflict resolution, search, tests.

💡 Bonus (Optional)

- Search notes by content/title (full-text search)
- Tagging or categorization of notes
- Conflict resolution UI to manually resolve sync conflicts
- Unit/integration tests
- Deploy as a PWA with a service worker for full offline support

---

📦 Deliverables

- **Source code repository** (GitHub, GitLab, etc.) with meaningful commit history.
- A **README.md** file that includes:
  - Setup instructions
  - Design decisions and tradeoffs you made
  - Any assumptions or limitations
  - Instructions on how to run and test the app
- Optional: deployed app link (e.g., Vercel, Netlify)

Getting Started

1. **Setup your React app** (e.g., with Create React App, Vite, Next.js).

2. Implement **IndexedDB integration** to store notes locally.
3. Build the **UI for listing and editing notes** with markdown support.
4. Add **connectivity detection** and syncing logic to push/pull from mock API.
5. Handle **sync status** and conflict resolution.
6. Polish UI/UX and add bonus features if time allows.
7. Document your work clearly.

FAQs

Q: Can I use external libraries?
Yes! Use what you need, but keep things understandable and maintainable.

Q: What if I can't implement everything?
Focus on correctness, architecture, and core features first. Document what's missing.

Q: How do I simulate offline mode?
Use browser devtools (e.g., Chrome DevTools → Network → Offline) or disable your internet connection.

If you have questions or want to discuss your approach, feel free to reach out.

Good luck, and have fun building! 🚀