📝 Zadanie: Aplikacja do tworzenia fiszek językowych

🎯 Cel zadania

Celem jest stworzenie pełnoprawnej aplikacji webowej umożliwiającej użytkownikom tworzenie, przeglądanie i powtarzanie fiszek służących do nauki słownictwa. Aplikacja ma wspierać naukę poprzez zestawy fiszek zawierające słowo oraz jego tłumaczenie w różnych językach.

💻 Technologie

- **Frontend**: React + TypeScript (opcjonalnie z biblioteką UI: MUI, Tailwind lub inną)
- **Backend**: Node.js (Express.js)
- **Baza danych**: lokalny serwer w Node.js oparty o plik `.json` (np. z użyciem `lowdb` lub `json-server`)
- **Dodatkowe (mile widziane)**: React Query / SWR, JWT (jeśli zaimplementowane logowanie), Unit testy (Jest, Vitest)

🔧 Wymagane funkcjonalności

1. **Zarządzanie zestawami fiszek**
    - Tworzenie zestawów (nazwa, opis, język źródłowy i docelowy)
    - Dodawanie fiszek (np. `cat → kot`)
    - Edytowanie i usuwanie własnych zestawów oraz fiszek 📌 Obowiązkowe wsparcie dla języka angielskiego i polskiego (w obu kierunkach).
2. **Tryb nauki**
    - Wyświetlanie fiszek z wybranego zestawu w losowej kolejności
    - Możliwość oznaczenia „znam” / „nie znam”
    - Prosty algorytm powtórek na podstawie odpowiedzi użytkownika
3. **Statystyki nauki**
    - Liczba powtórek
    - Liczba zapamiętanych fiszek
    - Historia nauki (opcjonalnie)

🚀 Opcjonalne funkcjonalności

- Logowanie / rejestracja użytkownika (JWT)
- Obsługa wielu języków tłumaczeń (np. hiszpański, niemiecki, francuski)
- Eksport / import zestawów fiszek (CSV lub JSON)
- Wersja PWA z działaniem offline
- Responsywność interfejsu (działanie na urządzeniach mobilnych)

📌 Organizacja pracy

- Backlog zadań należy przygotować w darmowym narzędziu typu:
  - Trello, ClickUp lub Jira Free
- Repozytorium GIT (np. GitHub) z historią commitów
- Wersja demo wdrożona przez Netlify, Vercel lub Render
- Plik `README.md` z instrukcją uruchomienia projektu lokalnie

🗄️ Baza danych – lokalny serwer Node.js

🔷 Opcja 1: `lowdb` (zalecana)

- Minimalistyczna baza danych oparta o plik `.json`
- Instalacja:

```
npm install lowdb
```

- Przykład użycia:

```
import { Low } from 'lowdb'
import { JSONFile } from 'lowdb/node'

const adapter = new JSONFile('db.json')
const db = new Low(adapter)

await db.read()
db.data ||= { flashcards: [] }
db.data.flashcards.push({ word: "cat", translation: "kot" })
await db.write()
```

🔷 Opcja 2: `json-server`

- Symuluje REST API na podstawie pliku `db.json`
- Instalacja:

```
npm install -g json-server
```

- Uruchomienie:

```
json-server --watch db.json --port 3001
```

💡 Porady

1. Zacznij od stworzenia struktury katalogów i podstawowego UI (np. ekran dodawania fiszek)
2. Pracuj iteracyjnie – dodawaj funkcje krok po kroku (Minimum Viable Product)
3. Najpierw zadbaj o logikę działania, potem o wygląd i styl
4. Często testuj aplikację i zapisuj zmiany w GIT
5. Pracuj w osobnych gałęziach — każda funkcja = nowa gałąź
6. Wysyłaj zapytania HTTP przez `fetch`, `axios`, lub `React Query`