Zadanie Rekrutacyjne: Helpdesk Lite z Asystentem LLM, Design Systemem i Integracją API

Cel

Zbuduj małą, działającą aplikację **Helpdesk Lite**, która umożliwia użytkownikom zgłaszanie, przeglądanie i triagowanie ticketów.
Backend napisz w **Laravel**, frontend w **Angular**, interfejs oprzyj na **design systemie** z **Storybookiem**.
W procesie możesz (i **powinieneś**) korzystać z **LLM-a** (np. ChatGPT, Claude, Copilot itp.).

Chcemy zobaczyć **cały Twój proces**: jak analizujesz wymagania, projektujesz architekturę, korzystasz z LLM-a i doprowadzasz do działającego rezultatu.

Zakres funkcjonalny (MVP)

1. Użytkownicy i role

- Role: `admin`, `agent`, `reporter`.
- Logowanie (np. Laravel Sanctum / JWT / fake login).
- `reporter` widzi tylko swoje tickety, `agent` i `admin` widzą wszystkie.
- Można wykorzystać **publiczne API** do mockowania użytkowników (np. https://jsonplaceholder.typicode.com/users lub https://reqres.in/).

2. Tickety

- Pola:
  `title`, `description`, `priority`, `status`, `assignee`, `tags[]`, `created_at`, `updated_at`.
- CRUD: tworzenie, edycja, filtrowanie po statusie, priorytecie, przypisaniu, tagach.
- Historia zmian statusu (`ticket_status_changes` lub podobny model).

3. Triage Asystent (LLM)

- Na froncie przycisk **„Zasugeruj triage”**.
- Wywołanie endpointu w backendzie → backend komunikuje się z LLM (może być mock).
- Użytkownik może przyjąć lub odrzucić propozycję.

4. Integracja z zewnętrznym API

Dodaj **jedną integrację** z publicznym API (dowolne), np.:

- WeatherAPI – pokazanie pogody w mieście zgłaszającego,
- JSONPlaceholder – powiązanie ticketu z danymi użytkownika,
- ExchangeRate.host – przykład prostego fetchu danych (kursy walut).

Wymagania dla integracji:

- Obsłuż co najmniej dwa scenariusze: sukces + błąd (np. timeout, brak danych).
- Pokaż wynik integracji **na froncie** (np. w szczegółach ticketu lub osobnym komponencie).
- Zaimplementuj **cache lub prosty fallback** po stronie backendu (np. redis, session, plik, array).

Wymagania techniczne

Backend (Laravel)

- Laravel 10/11, PHP 8.2+.
- REST API (JSON), autoryzacja (Sanctum/JWT).
- Migracje + seedy (np. 3 role, 3 użytkowników, kilka ticketów).
- Testy: min. 3 jednostkowe + 2 integracyjne.
- Endpoint `POST /tickets/{id}/triage-suggest` – integracja z LLM/mocked.
- Endpoint `GET /external-data` – proxy do wybranego publicznego API z cache i obsługą błędów.

Frontend (Angular + Design System)

- Angular 16+.
- Routing, formularze reaktywne, stan (signals, NGRX lub inny).
- Użycie **design systemu** (np. Angular Material / inny).
- Responsywność (desktop + min. jedna adaptacja mobile).
- Komunikaty o błędach i loading states.
- Light/Dark mode mile widziany.
- Integracja z endpointem `GET /external-data` (np. pokazanie danych pogodowych lub użytkownika w karcie ticketu).

Design System + Storybook

Storybook

- Skonfiguruj **Storybook** dla Angulara:
  `npm run storybook` uruchamia lokalnie, `npm run build-storybook` buduje wersję statyczną.
- Włącz globalny theming (tokeny kolorów, spacing, radius, typografia).
- Dodaj Controls i Docs tab dla propsów komponentów.

Własne komponenty (2–3 sztuki)

Stwórz minimum **2–3 własne komponenty UI**, pokazane w Storybooku.
Propozycje:

1. `PriorityBadge` – wizualny wskaźnik priorytetu (Low/Medium/High).
2. `TicketCard` – karta skrótu ticketu (title, summary, tags, status).
3. `TriageSuggestionPanel` – panel z sugestią LLM (summary, przyciski Accept/Reject).

Każdy komponent:

- Widoczny w Storybooku z min. 3 stanami (np. default, loading, error).
- Ma opis, propsy, kontrolki, i obsługę light/dark mode.

Bonus:

- Storybook hostowany (np. na Chromatic / Vercel / GitHub Pages).

Praca z LLM – obowiązkowe artefakty

1. **Screencast (5–8 minut)**
   - Pokaż realną pracę z LLM (promptowanie, doprecyzowanie, refaktoryzacja).
   - Pokaż jak LLM pomaga Ci w: architekturze, generowaniu kodu, testach.
   - Pokaż iterację (coś poprawiasz, testujesz, uruchamiasz).
2. **README – sekcja „LLM Flow”**
   - Jak korzystałeś z LLM (strategie promptów, przykłady promptów, walidacja wyników).
   - Czy LLM pomógł w kodzie, testach, czy dokumentacji.
   - Jak radziłeś sobie z błędami / halucynacjami.

Deliverables

- Link do repozytorium (publiczne lub prywatne z dostępem).
- Działający **Docker Compose** (`backend`, `frontend`, `db`).
- Plik `.env.example` + **README** (uruchomienie, LLM Flow, decyzje).
- Link do screencastu pracy z LLM.
- (Opcjonalnie) Link do zdeployowanego Storybooka.
- 2–3 zrzuty ekranu gotowej aplikacji.

Timebox

6–8 godzin.
Nie oczekujemy perfekcji — zależy nam na **skończonym, logicznym MVP**, pokazującym Twoje podejście i styl pracy.

Wskazówki

- Nie liczy się ilość kodu, tylko jakość i sposób myślenia.
- Pokaż, że potrafisz **współpracować z LLM-em**, ale też **samodzielnie podejmować decyzje**.
- Zadanie ma sprawdzić Twój **warsztat, architekturę, dbałość o UX i DX**, a nie tempo pisania kodu.