Technický úkol: Restaurant Menu Summarizer

Tento úkol testuje tvé schopnosti pracovat s LLM API, strukturovanými výstupy a real-world daty z webu. Představ si, že vytváříš nástroj pro lidi, kteří chtějí rychle zjistit, co je dnes k obědu v jejich oblíbené restauraci.

Zadání

Vytvoř buď BE službu (REST API) nebo FE aplikaci, která:

1. Přijme URL adresu stránky s menu restaurace
2. Získá obsah stránky (pomocí scraperu nebo LLM built-in search/web fetch)
3. Pomocí LLM API vyextrahuje a sumarizuje menu na dnešní den
4. Uloží výsledek do cache
5. Vrátí strukturovaná data v použitelném formátu

Use case příklad

Vstup:

URL: https://www.restaurace-example.cz/menu
Dnešní datum: 22. října 2025 (středa)

Očekávaný výstup (strukturovaný):

JSON

```
{
	"restaurant_name": "Restaurace Example",
	"date": "2025-10-22",
	"day_of_week": "středa",
	"menu_items": [{
		"category": "polévka",
		"name": "Hovězí vývar s nudlemi",
		"price": 45,
		"allergens": ["1", "3", "9"]
	},
	{
		"category": "hlavní jídlo",
		"name": "Kuřecí řízek s bramborovou kaší",
		"price": 145,
		"allergens": ["1", "3", "7"],
		"weight": "150g"
	}],
	"daily_menu": true,
```

```
        "source_url": "https://www.restaurace-example.cz/menu"
}
```

Technické požadavky

1. Web Content Retrieval
    a. Implementuj získání obsahu stránky pomocí:
        i. Variantu A: Vlastní scraper (Cheerio, Puppeteer, Playwright, BeautifulSoup)
        ii. Variantu B: LLM built-in search/web fetch (např. Claude web search, Perplexity API)
    b. V README vysvětli, kterou variantu jsi zvolil/a a proč2.
2. LLM API Integration
    a. Použij libovolné LLM API (OpenAI, Anthropic Claude, Google Gemini)
    b. Implementuj structured output (validovaná JSON schema pro menu items)
    c. Použij tool calling/function calling pro minimálně jednu z těchto funkcí:
        i. Parsování a normalizace cen (např. "145,-" → 145)
        ii. Detekce dnešního dne v týdnu z různých formátů
        iii. Konverze měr/vah do standardního formátu
        iv. Vlastní funkce podle tvého uvážení
3. Architektura
    a. BE varianta: REST API s minimálně 1 endpointem:
        i. POST /summarize` - sumarizace menu z URL
    b. FE varianta: Single page aplikace s formulářem (URL input) a zobrazením výsledků
    c. Pro BE není potřeba UI, stačí dokumentace API (README + příklady cURL/Postman)
4. Caching menu dat (POVINNÉ)
    a. Implementuj ukládání vyextrahovaných menu dat z LLM API
    b. Při opakovaném volání se stejnou URL a datem vrať data z cache místo nového LLM volání
    c. Řešení by mělo být persistentní storage (SQLite, PostgreSQL, Redis)
    d. Cache by měla mít klíč: `URL + datum` (menu se mění denně) -
    e. Cache by měla mít TTL nebo logiku invalidace (např. po půlnoci smazat starší záznamy)
    f. V README vysvětli své rozhodnutí o typu cachingu
5. Testy
    a. Minimálně 2 unit testy (např. parsování odpovědi, validace struktury) -
    b. Minimálně 1 integrační test (např. celý flow analýzy)
    c. 1 test cachingu - ověř, že při druhém volání se neposílá request na LLM API
    d. Bonus: E2E test (pokud FE)
6. Kód kvalita
    a. Čistý, čitelný kód s komentáři u složitějších částí
    b. Error handling (zejména pro LLM API volání)
    c. Environment variables pro API klíče

d. README s instrukcemi pro spuštění

Zakázané

- Žádné AI frameworky (LangChain, LlamaIndex, CrewAI, atd.)
- Pokud framework použiješ, vysvětli v README proč a jakou hodnotu přináší

Plusové body (není povinné)

- Docker/Docker Compose - ready-to-run kontejner
- Autentizace - jednoduchý API key nebo JWT token
- FE design - pokud děláš FE
- Bonus nápady (implementuj, pokud máš čas a chuť):
  - Detekce vegetariánských/veganských jídel
  - Filtrování podle alergenů
  - Cenové srovnání menu z více restaurací
  - Slack/Discord bot integrace
  - Webhook notifikace při změně menu

Co nás zajímá

Hodnotíme

1. Prompt engineering - jak instruuješ LLM k extrakci správných dat z semi-strukturovaného textu
2. Data extraction - správnost a úplnost vyextrahovaných informací
3. Structured outputs - správné využití function calling a JSON schemas
4. Caching implementace - efektivní ukládání a získávání dat, správná invalidace cache
5. Error handling - jak řešíš nedostupné stránky, špatný formát, chybějící data
6. Kód čitelnost - struktura projektu, naming conventions
7. Praktičnost řešení - použitelnost v reálném světě (co když stránka nemá dnešní menu?)

Nepodstatné

- Pixel-perfect design (pokud FE)
- Podporu všech možných formátů menu na internetu
- Pokročilé DevOps (stačí základní Docker, pokud ho děláš)
- 100% přesnost - chápeme, že menu stránky mají různé formáty

Co když nestíháš?

Raději než vibe coding udělej toto:

- Implementuj core funkcionalitu čistě
- Napiš v README, co by sis přál dodělat a jak by ses k tomu dostal

- Ukaž, že víš co děláš, i když to není 100% hotové

Technický stack (doporučený, ne povinný)

- Backend:
  - Node.js + Express/Fastify nebo NestJS
  - TypeScript
  - Jest pro testy
  - Volitelně: (PostgreSQL/SQLite + Prisma) | Redis
- Frontend:
  - React/Next.js
  - TypeScript
  - Tailwind CSS (nebo cokoliv, co znáš)
  - Volitelně: React Query pro API calls
- Python alternativa:
  - FastAPI nebo Flask
  - pytes

Testovací URL

Pro testování můžeš použít různé restaurace s denním menu. URL adresy restaurací přidej do README.

Edge Cases k zvážení

Nemusíš všechny řešit, ale je dobré o nich vědět:

- Stránka není dostupná (404, timeout)
- Menu není v textové podobě (pouze obrázek)
- Menu neobsahuje dnešní den (jen víkendové menu, nebo menu na celý týden)
- Nekonzistentní formát cen ("145,-" vs "145 Kč" vs "145")
- Chybějící alergeny nebo jiné informace
- Co když je dnes svátek a restaurace má zavřeno?
- Co když se menu během dne změní? (cache invalidation strategie)

V README stačí popsat, jak by ses k těmto situacím postavil/a, pokud je nestihneš implementovat.

Jak odevzdat

1. Pushni kód do public GitHub/GitLab repozitáře
2. Zahrň v README:
   a. Jak projekt spustit (step-by-step)
   b. Jak spustit testy
   c. Tvé úvahy o řešení (2-3 odstavce)

d. Co by ses chtěl zeptat nebo prodiskutovat
e. Pošli link na repozitář

Hodně štěstí! Těšíme se na tvé řešení a následný rozhovor o tom, jak by ses k problému postavil/a v reálném projektu.