RAPORT – AI Planner (Inteligentny plan dnia z LLM)

1. Cel projektu

Celem projektu było stworzenie aplikacji **AI Planner - inteligentny plan dnia**, która generuje harmonogram na podstawie listy zadań oraz ograniczeń czasowych. W projekcie wykorzystano **LLM (Large Language Model)** jako agenta planującego, który interpretuje zadania zapisane w języku naturalnym oraz buduje plan w postaci ustrukturyzowanej (JSON).

---

2. Założenia funkcjonalne

Aplikacja umożliwia:

- wprowadzenie zadań w formie tekstu (język naturalny),
- ustawienie ograniczeń dnia: godzina rozpoczęcia i zakończenia,
- określenie preferencji użytkownika: poziom energii (niska/średnia/wysoka),
- generowanie harmonogramu dnia z konkretnymi godzinami,
- dodanie uzasadnienia dla każdego przydzielonego zadania (reason),
- zwrócenie listy zadań, których nie udało się zaplanować (unscheduled).

---

3. Wykorzystane technologie

Python – implementacja logiki aplikacji

- **Streamlit** – interfejs webowy (UI)
- **Groq API** – dostęp do modelu językowego (LLM)
- **Model LLM:** llama-3.3-70b-versatile
- **JSON** – format wyniku harmonogramu
- **Zmienne środowiskowe / secrets** – bezpieczne przechowywanie klucza API (GROQ_API_KEY)

---

4. Architektura rozwiązania

Rozwiązanie działa w modelu: **User Input → LLM Agent → JSON Schedule → UI**

Przepływ danych:

1. Użytkownik wpisuje zadania do aplikacji.
2. Użytkownik ustawia ograniczenia (start/koniec dnia, energia).
3. Aplikacja buduje prompt z zasadami planowania.
4. LLM (Groq) generuje harmonogram w formacie JSON.
5. Aplikacja parsuje JSON i prezentuje harmonogram w UI.

Autorzy: Joanna Jędryka
Kinga Saltarius
Zofia Zimnol

5. Rola LLM w projekcie (dlaczego to jest “AI”)

LLM pełni rolę **agenta planującego** (LLM-based Planning Agent).
Model:

- interpretuje semantykę zadań (np. które są cięższe / wymagają energii),
- planuje kolejność oraz godziny realizacji,
- generuje uzasadnienie decyzji,
- potrafi wskazać zadania, których nie da się wcisnąć w plan dnia.

Logika planowania została przekazana do modelu poprzez odpowiednio zaprojektowany **prompt**, który zawiera:

- dane użytkownika (zadania i ograniczenia),
- zasady planowania,
- wymagany format odpowiedzi (JSON).

---

6. Opis działania programu (krok po kroku)

6.1 Pobranie klucza API

Aplikacja pobiera klucz Groq (GROQ_API_KEY) z:

- st.secrets (np. .streamlit/secrets.toml) lub
- zmiennej środowiskowej systemu.

Jeśli klucz nie jest dostępny, aplikacja przerywa działanie i pokazuje komunikat.

6.2 Przygotowanie promptu

Zadania i ograniczenia są zamieniane na tekst promptu, który zawiera zasady:

- odpowiedź musi być **czystym JSON**,
- harmonogram musi mieścić się w ramach dnia,
- jeśli zadanie nie mieści się w czasie → trafia do listy unscheduled.

6.3 Wywołanie modelu przez Groq API

Model jest wywoływany metodą chat.completions.create() z parametrami:

- model="llama-3.3-70b-versatile"
- temperature=0.3 dla stabilnych wyników
- response_format={"type": "json_object"} aby wymusić zwrot JSON

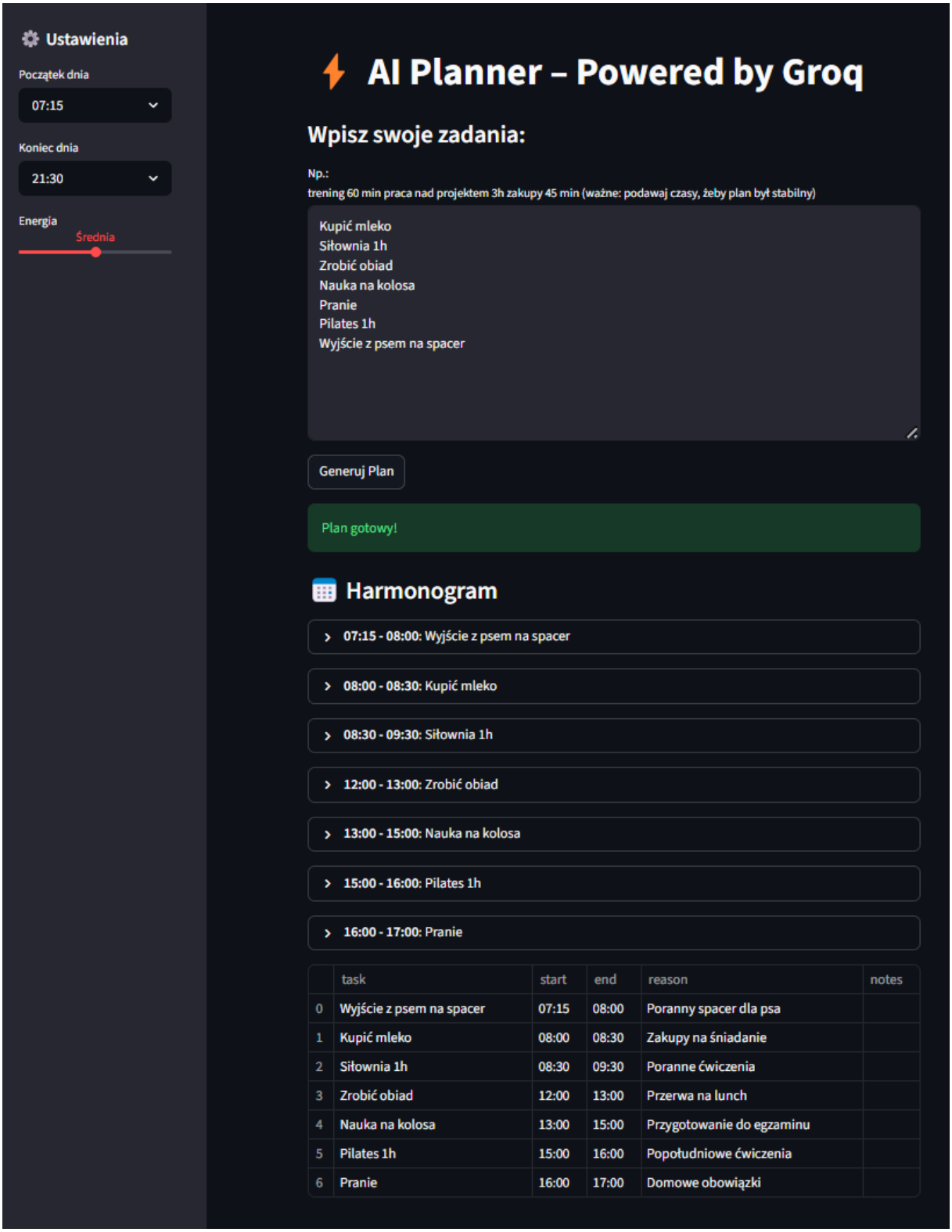
6.4 Prezentacja wyników

Aplikacja:

- wyświetla harmonogram jako listę bloków czasowych,
- pokazuje uzasadnienie (reason) i notatki (notes) w rozwijanych sekcjach,
- prezentuje wynik również w formie tabeli.

Autorzy: Joanna Jędryka
Kinga Saltarius
Zofia Zimnol

7. Przykładowe działanie



	task	start	end	reason	notes
0	Wyjście z psem na spacer	07:15	08:00	Poranny spacer dla psa	
1	Kupić mleko	08:00	08:30	Zakupy na śniadanie	
2	Siłownia 1h	08:30	09:30	Poranne ćwiczenia	
3	Zrobić obiad	12:00	13:00	Przerwa na lunch	
4	Nauka na kolosa	13:00	15:00	Przygotowanie do egzaminu	
5	Pilates 1h	15:00	16:00	Popołudniowe ćwiczenia	
6	Pranie	16:00	17:00	Domowe obowiązki	

Autorzy: Joanna Jędryka
Kinga Saltarius
Zofia Zimnol

9. Wnioski

Projekt pokazuje praktyczne zastosowanie LLM w roli agenta planującego. Dzięki temu aplikacja:

- obsługuje język naturalny jako input,
- generuje harmonogram bez ręcznie zakodowanego algorytmu optymalizacji,
- dostarcza uzasadnienia decyzji (ważne z punktu widzenia użytkownika),
- jest łatwo rozszerzalna (np. walidacja overlapów, replanning).

---

10. Możliwe rozszerzenia

walidacja poprawności harmonogramu (overlap, wyjście poza okno),

- automatyczne poprawianie planu przez LLM w przypadku konfliktów,
- integracja z kalendarzem,
- deadliny,
- przeplanowanie dnia po opóźnieniach.

---

11. Instrukcja uruchomienia projektu

1. Instalacja bibliotek:

```
pip install streamlit groq
```

2. Ustawienie klucza API:

- przez zmienną środowiskową:

```
$env:GROQ_API_KEY="TWÓJ_KLUCZ"
```

lub przez .streamlit/secrets.toml.

3. Uruchomienie:

```
streamlit run main.py
```

Autorzy: Joanna Jędryka
Kinga Saltarius
Zofia Zimnol