Zápor Ádám: Regex-kisegítő AI – Projekt-dokumentáció

---

1 Feladat célja

- **Chatbot-mód** – a felhasználó természetes nyelven megfogalmazott igényei alapján az AI:
  1. regexet **generál**;
  2. azt **magyarázza** (szövegesen);
  3. képes **visszautalni** korábbi kérdésekre/válaszokra a **chatHistory**-nek köszönhetően.

- **Benchmark-mód** – 30 előre definiált regex-feladatot futtat automatikusan. Minden futás eredménye egy **`bench_stats.json`** fájlba kerül.

---

2 Magas szintű architektúra

Réteg	Fájl	Szerep
CLI alkalmazás	`regexp_main.py`	futtatható script; betölti az összes komponenst, konzolos párbeszédet kezel
Eszköz-függvények	`tools.py`	`validate_regex` és egyéb belső eszközök
OpenAI-schema	`sources.py`	a `tools=` mezőhöz szükséges JSON-sémák
Segéd-osztályok	`helpers.py`	`ChatHistory`, `RegexInfo`

Benchmark	`benchmarks.py`	30 tesztpromptot hív, statisztikát ír ki

---

3 Kulcs-fogalmak és függvények

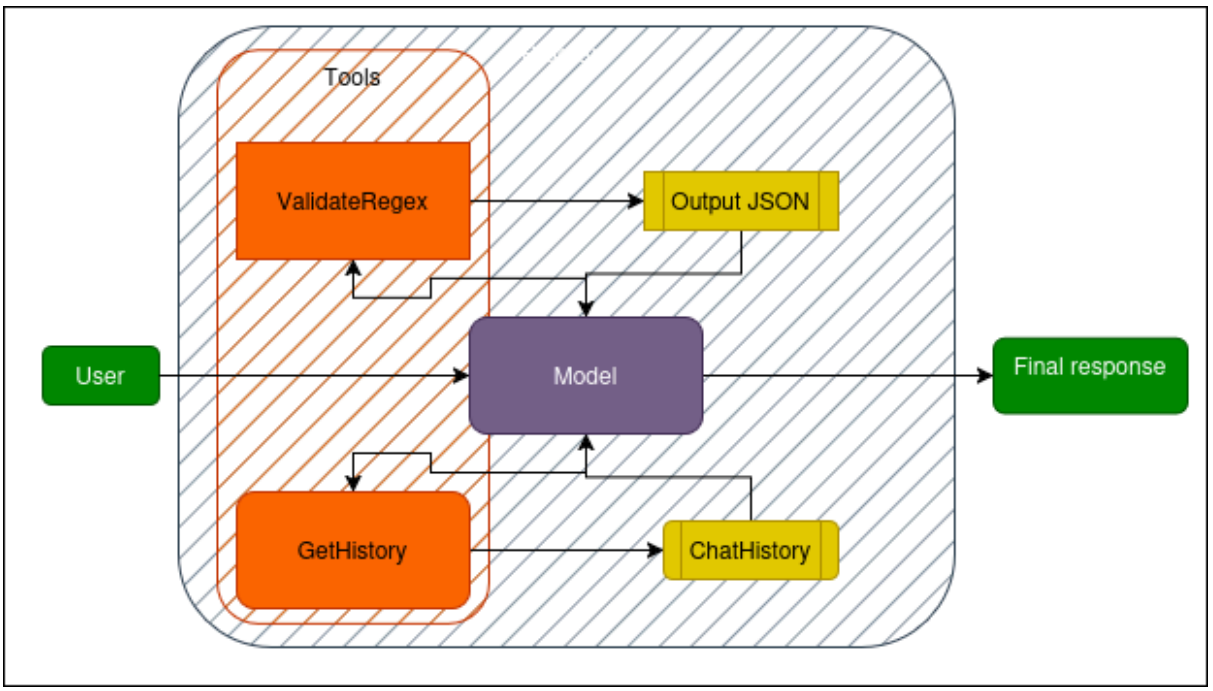
3.1 `validate_regex(pattern, test_str=None)` (`tools.py`)

Lépés	Művelet
1	Megpróbálja **compile-olni** a `pattern`-t. Hibánál → `ValueError`.
2	Ha `test_str` meg van adva, fut egy `search()`; siker esetén `is_match=True`, különben **`RegexNoMatchError`** (vagy `TypeError`).
3	Visszaad egy **JSON-stringet** a `RegexInfo` objektumból (flags, group-szám, is_match).

3.2 `ChatHistory` (`helpers.py`)

- Egy **deque**-ben tárolja a chat üzeneteket (max 41).
- Minden üzenet **serializálva** kerül bele (SDK-objektum → dict), így később gond nélkül `json.dumps()`-elhető.
  * `get_history(start, end)` → JSON slice, amit az AI **tool-híváson** keresztül kérhet le.

3.3 Tool flow



Először a User felteszi a kérdést. Az eredeti választ a modell a validateRegex függvény segítségével evaluál, majd egy JSON-t vagy stringet ad vissza. Tetszőlegesen a modell mghívhatja a korábbi promptokat és válaszokat, amit a GetHistory függvény visszaad. Végül ezekből újra készül egy prompt az AI-nak, és ennek a válasza lesz a végleges válasz.

---

4 Benchmark rendszer (30 feladat)

- **Feladatlista**: IPv4, IPv6, e-mail, ISO dátum, stb. (lásd `benchmark_prompts`).
- A modell **kötelezően** meghívja a `validate_regex`-et teszt-stringgel.
- A wrapper dekóder 3 esetet naplóz:
  * **ok** (compile + match)
  * **no_match** (RegexNoMatchError)
  * **invalid_regex** (re.error)
- Eredmény-fájl minta:

{

"summary": {"calls": 30, "ok": 24, "mismatches": 4, "errors": 2},

"per_prompt": […]

}

Tesztfutás alapján a modell a 30 feladatból ~4-et nem tud pontosan lefedni; főleg extrém hosszú vagy non-UTF-8 regexeknél hibázik.

---

5 Futtatási útmutató (rövid)

Ez a futtatáshoz kellő parancsok. Egy virtuális környzetben fut az AI, egy API-on keresztül, amihez sajnos kulcs kell, így ez inkább csak szemléletképpen van itt.

python3 -m venv .venv && source .venv/bin/activate

pip install -r requirements.txt

export OPENAI_API_KEY=…

python regexp_main.py # interaktív mód

python benchmarks.py # 30 teszt + JSON statisztika