ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Koltai Barnabás**

2024 Hallgató törzskönyvi száma: **T/008415/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Biomatika és Alkalmazott Mesterséges Intelligencia Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Koltai Barnabás**
Törzskönyvi száma: T/008415/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Szöveg generáló, kiegészítő és javító programok az AI-meghajtott tanulást segítő platform részére és chat bot felhasználói felület fejlesztése
Text generating, supplementary and correction programs for the AI-powered learning platform and chat bot user interface development

Intézményi konzulens: Balázs Milán
Külső konzulens:

Beadási határidő: 2024. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Mesterséges intelligencia specializáció

A feladat

Az oktatási szektorban jelenleg megfigyelhető egy dinamikusan fejlődő trend, melynek középpontjában az intelligens szoftverek és alkalmazások állnak, különösen kiemelkedik közülük a mesterséges intelligencia területe, mely exponenciális növekedést mutat. Általánosságban ezek egy speciális területre fókuszálnak, mint például nyelvtanulás vagy programozás, de a piacon kevés az olyan termék, ami több különböző területet lefed. Az AI-meghajtott tanulást segítő platform potenciálisan képes megoldást nyújtani erre a problémára. A hallgató feladata a szakdolgozatban ezen platform AI alapú részfeladatainak megoldása.
A szakdolgozat keretében a hallgató feladata lesz az AI alapú fogalmazást segítő, E-mail író funkciók tervezése és implementálása, valamint a Chat Bot felhasználói felületének megvalósítása.

A dolgozatnak tartalmaznia kell:

- szakirodalom, fejlesztési technológiák áttekintése,
- az AI-meghajtott tanulást segítő platform áttekintése,
- AI alapú fogalmazást segítő funkció tervezése és implementálása,
- AI alapú E-mail író funkció tervezése és implementálása,
- Chat Bot felhasználói felületének megvalósítása,
- eredmények értékelése.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Ph.

.....
Dr. Eigner György
intézetigazgató

A szakdolgozat elévülésének határideje: **2026. december 15.**
(ÓE HKR 54.§ (10) bekezdés szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat/diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban/diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2024. 12. 11.

Voltai Bence

hallgató aláírása

Absztrakt rövid tartalmi összefoglaló:

A szakdolgozatunk témáját saját tapasztalatok és felmerült problémák alapján választottuk ki. Törekedtünk egy innovatív megoldásra a 21. századi tanulók segítésére, hogy élvezetesebbé tegyük számukra a tanulást korlátok nélkül, a mai trendeknek megfelelően.

Az AI-meghajtott tanulást segítő platform egy szokványostól lényegesen eltérő weboldal, ahol a megszokottól eltérően nem csak egy adott kérdésre találhat a felhasználó megoldást, hanem teljes értékű tananyagokat képes generálni és később kártyás rendszerrel ismételni és kikérdezni a tanultakat. Az említettek mellett kibővítésre kerül egyéb hasznos funkciókkal is, mint például a fogalmazás javító és az email író.

A projekten hárman dolgozunk csapatban, szakdolgozatom során a weboldal rám eső feladatait fogom részletesen elemezni, megtervezni és fejleszteni. Az egyik legfontosabb feladatom a beszélgető felület felhasználói felületének tervezése és implementálása. Emellett két mesterséges intelligencia alapú funkción is fogok dolgozni, melyek a Fogalmazást segítő és az Email író.

A beszámolóm első felében egy gyors áttekintés látható a feladat fontosságáról, felosztásáról, tervezéséről és az általam használt technológiákról. A második felében található az implementációs rész, tesztelés és összegzés.

Abstract short content summary:

We chose the topic of our thesis based on our own experiences and problems we have encountered. We sought to find an innovative solution to help 21st century students to make learning more enjoyable, without constraints, in line with today’s trends.

The AI-powered learning platform is different kind of website, where the user can not only find the answer to a specific question, but also generate full-fledged learning materials and later repeat and quiz the learning using a card system. In addition tot he above, it will be extended with other useful features, such as a composition assistant and an email writer.

Three of us are working ont he project as a team, during my thesis I will analyse, design and dewvelop the website’s tasks in detail. One of my main tasks is to design and implement the user interface of the chatbot. Besides, I will laso work on two AI-based functions, which are the Composition Assistant and the Email Writer.

The first part of my report hives a quick overview of the importance of the task, its division, its design and the technologies I used. The second half contains the implementation, testing and the summary part.

Tartalomjegyzék

1. BEVEZETÉS

1.1. Probléma fontossága

Az oktatási szektorban tapasztalható jelenlegi fejlemények arra utalnak, hogy az intelligens szoftverek és alkalmazások terén egyre növekvő figyelem és fejlesztési tevékenység zajlik. Ezen belül kiemelkedő fontosságú a mesterséges intelligencia területe, amelynek dinamikus növekedése figyelemre méltó.

Általánosságban megfigyelhető, hogy ezek a technológiák gyakran specializálódnak egy-egy szűkebb területre, például nyelvtanulásra vagy programozásra. Azonban az iparágban még mindig hiánycikknek számítanak az olyan megoldások, amelyek több különböző területen is hatékonyan működnek egyszerre. Ez a helyzet nem teljesen egyértelmű, különösen figyelembe véve az oktatás sokszínűségét és a tanulási igények változatosságát.

Ebben a szakaszban az AI-meghajtott tanulást segítő platform kiemelkedő jelentőséggel bír. Ez a platform potenciálisan olyan megoldásokat kínálhat, amelyek integrálják a mesterséges intelligenciát és személyre szabott tanulási módszereket, ezáltal hatékonyan támogatva a tanulást és oktatást. Az ilyen típusú technológiai fejlesztések stratégiai jelentőséggel bírnak az oktatási iparág jövőjének formálásában és optimalizálásában.

1.2. Felvezetés

Szakdolgozatom témáját két hallgatótársammal együtt fejlesztett projektre építem, ennek során egy felhasználóbarát, böngészőben futtatható programot készítünk diáktársaink számára, hogy segítsük a tanulmányaikat, élvezetesebbé tegyük azt. A platform egyfajta tanulótársként funkcionál és segít az egyedüli számonkérésre való felkészülésre.

Mindhárman jelentős, de lényegesen eltérő funkcióin dolgozunk a szoftvernek mind front-end és back-end részen.

1.3. Dolgozat célja

Hármunk munkáját ez a webes alkalmazás fogja összekötni.

A projekt kezdőfelületén a felhasználó beszélgetést kezdhet nagy nyelvi modellekkel támogatott mesterséges intelligencia asszisztenssel, mely ki fogja választani a megfelelő funkciókat és az adott témában fog választ nyújtani. A beszélgetés során a felhasználó kérdések feltevésével férhet hozzá a platform beszélgető felületes funkcióihoz. A többi funkciót szokásos módon a felhasználói felületen való navigálással könnyedén tudja majd elérni. A projekt során minden funkció alapja a mesterséges intelligencia.

1.4. Felépítés és funkciók

- *Egy programozás tanulást segítő funkció, amin keresztül az asszisztens egy programozói kérdésre egy validált kód blokkot képes generálni*: A háttérben a szerver legenerálja a szükséges kódot és addig teszteli ameddig a kód validálás megfelelő eredményt nem produkál
- *Kérdés megválaszolás internetes tartalmak vagy feltöltött dokumentumok alapján*: A kérdésre kereső motor segítségével talált tartalomból vagy a meglévő dokumentumok szövegéből Retrieval Augmented Generation (későbbiekben RAG) segítségével választ ad az asszisztens a feltett kérdésre. A válasz forrását a felhasználó beállíthatja, amennyiben a beállított forrásból nem képes válaszolni az asszisztens, a másik forrás lesz használva.
- *Egy tanulókártya funkcionalitás*: A felhasználónak lehetősége lesz gyűjteményekbe rendezni tanulókártyáit, új tanulókártyákat hozzáadni a meglévő gyűjteményeihez. Ezeket a gyűjteményeket többféleképpen lehet létrehozni. Vagy a felhasználó manuálisan elkészíti őket, vagy pedig különféle mesterséges intelligencia megoldással, internetről, pdf-ből tud generáltatni egy gyűjteményt.
- *Mesterséges intelligencia alapú fogalmazást segítő funkció*: A felhasználó lehetőséget kap a már kész vagy félkész szövegének beküldésére, majd ezt különböző szempontok alapján javíthatja, formázhatja vagy kiegészítheti az asszisztens segítségével.
- *Email író funkció*: Megadott paraméterek, sablonok és kulcsszavak alapján email generálása.
- *Chat Bot felhasználói felületének megvalósítása*: A mesterséges intelligencia alapú asszisztens felhasználói felületének tervezése és implementálása, törekedve egy reszponzív és felhasználó barát felület létrehozására.

1.5. Az implementáció technikai felépítése magas szinten

- Funkciók nagyrészének a logikája a Python szerveren került implementálásra.
- A szerveren egy MongoDB adatbázisban kerülnek mentésre a felhasználó adatai.
- A nagy nyelvi modellek és gépi tanuláshoz szükséges számitásokat a Python szerver a BlackAnt nevű platformon fogja elvégezni.
- A Vue Webes könyvtárban megírt kliens JSON formátumban lévő HTTP hívásokkal fog kommunikálni a Python szerverrel.

1.6. Applikáció rám vonatkozó részei

- AI alapú fogalmazást segítő funkció tervezése és implementálása
- E-mail író funkció tervezése és implementálása
- Chat Bot felhasználói felületének megvalósítása

2. VUE.JS

2.1. Áttekintés

A Vue.js (későbbiekben Vue) [1] egy nyílt forráskódú, progresszív keretrendszer felhasználói felületek és egyoldalas alkalmazások készítéséhez. A szabványos HTML, CSS és JavaScript alapokra épül és olyan deklaratív komponensalapú programozási modellt biztosít, amely segít bármilyen összetettségű felhasználói felület hatékony fejlesztésében. [2]

A Vue-t [1] már az alapjaitól úgy tervezték, hogy teljes mértékben adaptálható lehessen. A központi könyvtár csakis a nézet (view) rétegre összpontosít, és könnyen felvehető és integrálható más könyvtárakkal vagy meglévő projektekkel.

2.2. Progresszív keretrendszer

A Vue [2] egy olyan keretrendszer és ökoszisztéma, amely a frontend fejlesztéshez szükséges általános funkciók többségét lefedi. A web azonban rendkívül változatos, a dolgok, amelyeket a webre építünk, drasztikusan eltérőek lehetnek a formájukban és a méretükben. Ezt szem előtt tartva a Vue-t úgy tervezték, hogy rugalmas és fokozatosan átvehető legyen. A felhasználási esettől függően a Vue különböző módokon használható:

- Statikus HTML bővítése építési lépés nélkül
- Webkomponensként történő beágyazás bármely oldalon
- Egyoldalas alkalmazás (SPA)
- Fullstack / szerveroldali megjelenítés (SSR)
- Jamstack / statikus oldal generálás (SSG)
- Asztali, mobil, WebGL és akár terminál célokra

2.3. Jellemzők

2.3.1. SFC

A legtöbb Vue projektben a komponenseket egy HTML-szerű fájlformátummal, az úgynevezett Single-File Component (más néven *.vue fájlok, rövidítése SFC) segítségével készíthetjük el. A Vue SFC, ahogy a neve is mutatja, egyetlen fájlba foglalja a komponens logikáját (JavaScript), sablonját (HTML) és stílusait (CSS). [2] Erre egy példa:

```vue
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

2-1. ábra - „Single-File Components” [2]

Az SFC a Vue egyik meghatározó jellemzője, és a Vue komponensek írásának ajánlott módja, ha a felhasználási eset megkívánja a build beállításokat. A Vue kezeli az összes build eszköz beállítását a felhasználó számára.

2.3.2. Api

A Vue [2] komponensek két különböző api stílusban írhatók: Options API és Composition API.

Options API

- Az Options API [2] segítségével egy komponens logikáját egy beállítások objektummal határozzuk meg, mely tartalmazza az adatokat, metódusokat és mounted-et. Az Options által meghatározott tulajdonságok ki vannak téve a komponens példányára mutató belső függvényeknek.

Composition API

- A Composition API [2] segítségével egy komponens logikáját importált API-funkciók segítségével határozzuk meg. Az SFC-ben a Composition API-t jellemzően a <script setup>-al használjuk. A setup attribútum egy súgó, amely a Vue-t olyan fordítási idejű átalakításokra készteti, amelyek lehetővé teszik számunkra, hogy a Composition API-t kevesebb forráskóddal használjunk.
- A projekt során legtöbbet Composition API-t használok, leginkább az ismétlődő komponensekhez, mint például a felhasználói felület üzenet buborékjaihoz.

2.4. Használatának oka

A projekt fő célja, hogy egy barátságos, könnyen kezelhető és értelmezhető felületet kapjanak a felhasználók, ahol nem merülnek fel használat béli kérdések. A Vue felsorolt tulajdonságai által ezek az igények mind megvalósíthatók.

3. QUASAR

3.1. Mi a Quasar?

A Quasar [3] egy MIT licenszű, nyílt forráskódú, Vue.js alapú keretrendszer, amely lehetővé teszi, hogy a webfejlesztő gyorsan reszponzív weboldalakat és alkalmazásokat hozhasson létre számos formában:

- SPA-k (Egyoldalas alkalmazások)
- SSR (Szerveroldali renderelt alkalmazás) (+ opcionális PWA kliens átvétele)
- PWA-k (Progresszív webalkalmazás)
- BEX (Böngésző kiterjesztés)
- Mobilalkalmazások (Android, iOS, …) Cordova vagy Capacitor segítségével
- Többplatformos asztali alkalmazások (Electron használatával)

A Quasar használata során nincs szükség további nagy könyvtárakra, mint a Hammer.js, Moment.js vagy a Bootstrap. Ezeket az igényeket belsőleg fedezi, és mindezt kis helyigény mellett.

3.2. Quasar előnyei

- Reszponzív asztali és mobil weboldalak.
- Szinte minden webfejlesztési igényre van egy komponens. Minden komponens kiemelkedő minőségű, teljesítmény orientált és reszponzív.
- Rengeteg funkció azonnal használható formában, melyek nem igényelnek utólagos konfigurációt.
- Alkalmazásbővítmények támogatása.
- Teljes RTL (jobbról balra) támogatás, a fejlesztő által írt weboldal vagy alkalmazás CSS-kódja automatikusan RTL-re konvertálódik RTL nyelvi csomag használatakor.
- A meglévő projekt fokozatos migrálása. A Quasar UMD (Unified Module Definition) verziót kínál, ami azt jelenti, hogy a fejlesztők egy CSS és JS HTML taget adhatnak a meglévő projektükhöz, és máris készenállnak a használatra. Nincs szükség buildelésre.
- Automatizált tesztelés és audit. A Quasar-projektek képesek egység- és végponttól-végpontig tartó tesztelésre, valamint a termékminőségi és biztonsági auditálási eszközök egyre bővülő csomagjára.
- Platformtámogatás széles skálája:
    - Google Chrome, Firefox, Edge, Safari, Opera
    - IOS, Android
    - MacOS, Linux, Windows
- Quasar nyelvi csomagok. Több mint 40 nyelvi csomaggal van felszerelve alapértelmezetten, ráadásul saját nyelvi csomagok hozzáadása is engedélyezett.

- Nagyszerű és részletes dokumentáció.

3.3. Miért használjuk?

A Quasar, a CLI-vel együtt kínált egyszerűsége és teljesítménye miatt rengeteg olyan funkciót tartalmaz, melyek mind arra szolgálnak, hogy megkönnyítsék a fejlesztő életét.

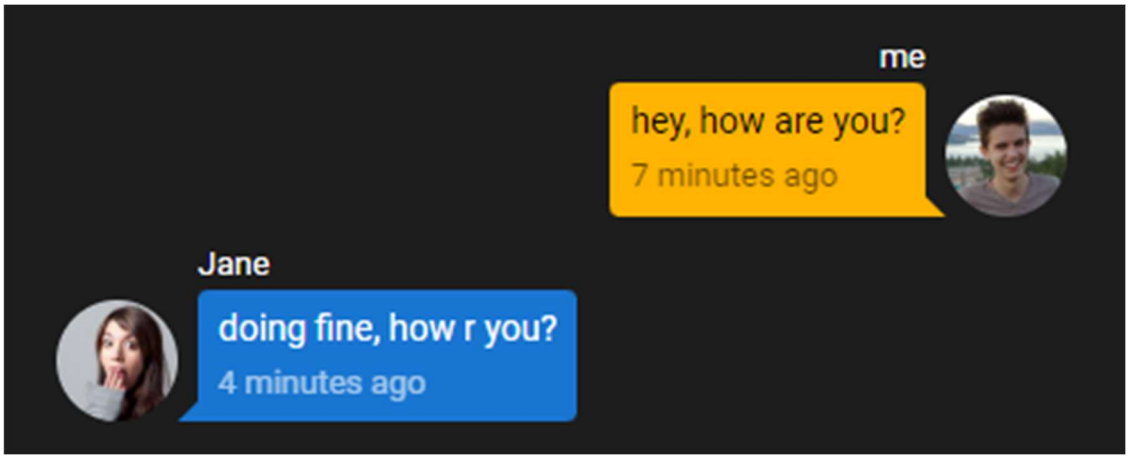
Az előre elkészített komponensek és a velük való fejlesztés egyszerűsége fontos szerepet játszott a Quasar melletti döntésben. Jelen esetben a komponensek könyvtárát saját magam által elkészített univerzális komponensekkel bővítem, ez lényegesen megkönnyíti a későbbi fejlesztéseket.

A teljes körű dokumentáció óriási segítséget nyújt a fejlesztés során, melyben még a legapróbb részletekre is kitértek a készítők. Minden komponens és funkció részletes kidolgozásra került, melyet könnyedén el lehet helyezni a forráskódban és később konfigurálni azokat.

Példa a felhasználói felület egyik legfontosabb komponensére, a szöveg buborékra:

```
<template>
  <div class="q-pa-md row justify-center">
    <div style="width: 100%; max-width: 400px">
      <q-chat-message
        name="me"
        avatar="https://cdn.quasar.dev/img/avatar1.jpg"
        :text="['hey, how are you?']"
        stamp="7 minutes ago"
        sent
        bg-color="amber-7"
      />
      <q-chat-message
        name="Jane"
        avatar="https://cdn.quasar.dev/img/avatar5.jpg"
        :text="['doing fine, how r you?']"
        stamp="4 minutes ago"
        text-color="white"
        bg-color="primary"
      />
    </div>
  </div>
</template>
```

3-1. ábra - Példa komponens kód [9]



3-2. ábra - Példa komponens [9]

4. LLAMA 3

A Llama 3 [4] egy hozzáférhető, nyílt forráskódú nagy nyelvi modell (LLM), amelyet fejlesztők, kutatók és vállalatok számára terveztek, hogy generatív AI ötleteiket felépíthessék, kísérletezhessenek és felelősségteljesen skálázhassák. Egy alaprendszer részeként az innováció alapköveként szolgál a globális közösségben.

A Llama 3 [5] mindkét mesterséges intelligencia alapú funkcióm megvalósítására alkalmas. Képes fordításra, szöveggenerálásra, könnyen kezeli a többlépcsős feladatokat, miközben a kifinomult utólagos képzési folyamatai jelentősen csökkentik a hamis elutasítások arányát, javítják a válaszok összehangolását és növelik a modell válaszainak sokszínűségét. A Llama 3-at [6] párbeszédes felhasználási esetekre optimalizálták, ami számos elérhető, nyílt forráskódú csevegőmodellnél jobb teljesítményt képes nyújtani.

A Llama 3 két méretben érkezik, 8B és 70B paraméterekkel érhető el, előre betanított és utasításra hangolt változatban. Mindkét verzió a jobb következtetési skálázhatóság érdekében Csoportosított-Lekérdezési Figyelmet (GQA) használ. A modell bemenetként kizárólag szöveg formátumot fogad el, kimenetként viszont szöveg és kód generálásra képes. A betanított adatok a 8B verzió esetében 2023. márciusig és a 70B verzió esetében 2023. decemberig érvényesek.

Több mint 15 trillió nyilvánosan elérhető forrásból származó adatmintán képezték le. A finomhangolási adatok között nyilvánosan elérhető utasításadathalmazok, valamint több mint 10 millió ember által kommentált példa is szerepel.

5. AXIOS

Az Axios [7] egy ígéretalapú HTTP-kliens a node.js és a böngésző számára. Izomorf (a böngészőben és a nodejs-ben is futtatható ugyanazzal a kódbázissal). A szerveroldalon a natív node.js http modult használja, míg a kliens (böngésző) XMLHttpRequests-t használ.

A projektben a funkciókat Axios segítségével api hívásokkal lehet elérni a BlackAnt-en futó szerverről.

6. PINIA

A Pinia [8] egy tároló könyvtár a Vue-hoz, lehetővé teszi, hogy egy állapotot megosszon a komponensek / oldalak között. Ennek segítségével nincs az alkalmazás kitéve biztonsági réseknek ellentétben a Vue beépített exportáló funkciójához képest.

A projekt során a legfontosabb szerepét az API hívások tárolásakor fogja betölteni. Az összes API egyetlen Pinia store-ba fog kerülni, amit később bármelyik komponensben meg lehet hívni az importálása után.

Példa kód a Pinia működésére:

```
// stores/counter.js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
  state: () => {
    return { count: 0 }
  },
  // could also be defined as
  // state: () => ({ count: 0 })
  actions: {
    increment() {
      this.count++
    },
  },
})
```

6-1. ábra - Pinia működése példa [8]

7. MONGODB

7.1. Mi a MongoDB?

A MongoDB [10] egy dokumentum-adatbázis, amelyet az alkalmazások fejlesztésének és méretezésének megkönnyítésére terveztek. A MongoDB a következő környezetekben futtatható:

- MongoDB Atlas: A teljes mértékben menedzselt szolgáltatás a MongoDB felhőben történő telepítéséhez.
- MongoDB Enterprise: A MongoDB előfizetéses, önmenedzselt változata.
- MongoDB Community: A MongoDB forrásából elérhető, ingyenesen használható és önmenedzselt változata.

Használati lehetőségek:

- Az adatok tárolása és lekérdezése.
- Adatok átalakítása aggregációkkal.
- Biztonságos hozzáférés az adatokhoz.
- Adatbázis telepítése és skálázása.

7.2. PyMongo

A PyMongo [11] egy Python disztribúció, amely a MongoDB-vel való munkához szükséges eszközöket tartalmazza, ez az ajánlott módja a Pythonból való munkának a MongoDB-vel. [12] Szinkron API-t biztosít a MongoDB adatbázisokkal való munkához, amely összhangban van a többi MongoDB nyelvi illesztőprogrammal.

7.3. Használatának oka

A Python szerver a BlackAnt-en fog futni és ez áll rendelkezésre. A MongoDB egyszerűen használható, könnyű rá csatlakozni. Hosszútávon nem lesz ideális megoldás, viszont a szakdolgozat során nem lesz annyi adat az adatbázisban, hogy bármilyen problémába ütközzek.

8. PYTHON

8.1. Mi a Python?

A Python [13] egy interpretált, objektumorientált, magas szintű, dinamikus szemantikájú programozási nyelv. Magas szintű, beépített adatszerkezetei, valamint dinamikus típus- és kötéskezelése rendkívül vonzóvá teszi gyors alkalmazásfejlesztés (Rapid Application Development) céljára, illetve szkriptnyelvként vagy „ragasztónyelvként” is használható meglévő komponensek összekapcsolásával. A Python egyszerű, könnyen elsajátítható szintaxisa a kód olvashatóságát helyezi előtérbe, ezáltal csökkenti a programkarbantartás költségeit. A Python támogatja a modulokat és csomagokat, ami elősegíti a program modularitását és a kód újra felhasználását. A Python interpreter-je és kiterjedt szabványos könyvtára forrás- vagy bináris formában ingyenesen elérhető minden jelentősebb platformon.

8.2. Flask

A Flask [14] egy könnyű WSGI (Web Server Gateway Interface) webalkalmazás-keretrendszer, amelyet úgy terveztek meg, hogy gyors és egyszerű legyen az indítás, ugyanakkor képes legyen bonyolult alkalmazások kezelésére is. Eredetileg a Werkzeug és Jinja egyszerű csomagjaként indult, de mára az egyik legnépszerűbb Python-alapú webalkalmazás-keretrendszerré vált.

A Flask ajánlásokat kínál, de nem kényszerít semmilyen függőséget vagy projektstruktúrát. A Fejlesztőre van bízva, hogy milyen eszközöket és könyvtárakat választ. A közösség által biztosított számos kiterjesztés lehetővé teszi az új funkciók könnyű hozzáadását, ami még rugalmasabbá és vonzóbbá teszi a fejlesztők számára.

```
# save this as app.py
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"
```

```
$ flask run
  * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

8-1. ábra - "A Simple Example" [14]

8.3. Használatának oka

Használatának az oka a mesterséges intelligencia alapú szerverünk hatékony implementálása volt. Az AI modelleket és az ahhoz szükséges csomagokat kizárólag Python segítségével lehet futtatni. Szerencsére egyetemi tanulmányaink során beleláthattunk a Python működésébe, természetesen voltak újdonságok és még nem használt kódrészletek, de óriási meglepetéseket nem tartogatott számunkra.

9. BLACKANT

9.1. Mi a BlackAnt?

A BlackAnt [15] platform egy olyan keretrendszer, amely megoldást kínál arra a problémára, hogy a felhasználóknak számos erőforrásigényes számítást (különösen mesterséges intelligenciát, mélytanulási és gépi tanulási algoritmusokat) kell biztosítaniuk vagy használniuk magas rendelkezésre állás mellett, miközben nem éri meg számukra olyan fejlesztőket alkalmazni, akik rendelkeznek a szükséges szakértelemmel, illetve nem célszerű helyi erőforrásokat fenntartani egy vagy néhány ilyen szolgáltatáshoz.

A BlackAnt rendszer célja, hogy kutatók és olyan felhasználók számára nyújtson lehetőséget magas rendelkezésre állású szolgáltatások fejlesztésére és futtatására, akik nem rendelkeznek a szükséges szakértelemmel. A BlackAnt egy felhőalapú platform, ezért nincs szükség helyi erőforrások biztosítására.

9.2. Használatának oka

A szakdolgozati projektünk szerverét a BlackAnt fogja futtatni a feljebb már említett szempontok miatt. Segít abban, hogy a nap bármely pontján elérhető legyen a szerver és stabilan fusson bármi féle hardveres komplikáció nélkül.

10.FUNKCIÓK TERVEZÉSE

10.1. Chat bot felhasználói felülete

A weboldal lelke a chat bot, ahol számos funkció elérhetővé válik a felhasználó számára, mint például a későbbiekben említett fogalmazást segítő és email író funkció.

A felhasználói felületet a már fentebb kifejtett Vue.js és Quasar framework segítségével hoztam létre.

A tanulás során lehetőség van több beszélgető ablak megnyitására, ha egyszerre szeretnénk különböző témákban beszélgetni a mesterséges intelligenciával. Ez lehetővé teszi, hogy az egyes funkciókat akár egyidőben is lehessen használni. Egy új beszélgető ablak megnyitásakor a mesterséges intelligencia, más néven „HelperBot" megkérdezi a felhasználót, hogy miben segíthet és milyen témákról szeretne tanulni. Kezdetekben a HelperBot angol nyelven fog kommunikálni, később igény esetén ez bővítésre kerülhet. A beszélgető ablak mellett a felhasználó számára rendelkezésre áll egy legördülő lista, melyben ki tudja választani a HelperBot fő funkcióját, legyen ez tanulás, fogalmazás vagy email írás, az utóbbi kettőről a következő pontban lehet olvasni bővebben.

Mint azt már említettem az ablak megnyitásakor a HelperBot el is kezdi a beszélgetést, a felhasználónak már csak egy feladata marad, hogy feltegye a kérdést, hogy milyen témában szeretne tanulni. A HelperBot működéséről Pálosi Ákos szakdolgozatában lehet többet olvasni. Amíg a válaszra várunk ezt jelzi nekünk az oldal egy gondolkodó jelzéssel. A már legenerált válaszokat, de a saját kérdéseinket is lehetőségünk van a vágólapra másolni a megfelelő szövegbuborékra való kattintással. Egér rávitelével egy felugró információs szöveg tájékoztatja a felhasználót ennek működéséről.

A felhasználói felületet Figma-ban terveztem meg. Azért erre esett a választás, mert nagyon hasonló komponensekkel lehet benne dolgozni, mint a Quasar esetén és ez megkönnyíti a későbbi fejlesztési folyamatokat. Pasztell színkombinációt adtam a weboldalnak, igyekeztem a színek megválasztásánál törekedni arra, hogy a fontos részeket figyelemfelkeltően jelenítsem meg, mégis kíméletes legyen a szemnek.

10.2. Funkciók

A szakdolgozatom során a felhasználói felület mellett két fontos szerepet játszó funkció kidolgozása a feladatom, ezek a fogalmazást segítő és az email író. Mindkét esetben a felhasználónak a legördülő funkcióválasztó listából van lehetősége eldönteni, hogy aktuálisan mit szeretne használni, alapértelmezetten a tanuló felület jelenik meg új beszélgető ablak megnyitásakor.

Modell választásnál a Llama 3-ra esett a választás, amit fentebb már kifejtettem. Ígéretesnek tűnik a gyorsasága, a betanított adatok mennyisége és a bő konfigurációs lehetőségei miatt.

10.2.1. Fogalmazást segítő

A funkció választó listából a felhasználónak lehetősége van a fogalmazást segítő funkció kiválasztására. Ekkor az oldalon a beszélgető felület eltűnik és a helyére egy kinézetben nagyon hasonló, de működésben lényegesen eltérő felületet kap.

A HelperBot a felhasználó üdvözlése után felsorolja az elérhető lehetőségeket:

- Szöveg kiegészítése
- Nyelvtani hibák javítása
- Újra fogalmazás

A szükséges segítség kiválasztása utáni lépés a szöveg beillesztése, amit rövid várakozás után kiegészítve, javítva vagy újra gondolva tudhatunk magunkévá. A visszakapott szöveg ugyanúgy, mint a Chat bot-nál, egy kattintással másolható a vágólapra.

A felhasználó számára rendelkezésre állnak beállítási lehetőségek, ha nem lenne megelégedve a kapott válasszal. Itt testre tudja szabni, hogy mennyire legyen hosszú vagy rövid, mennyire legyen bonyolult a szöveg megfogalmazása. Beállítások mentése után a HelperBot újra lefuttatja a modellt és a visszakapott szöveg már teljesen a felhasználó igényeire lesz szabva.

10.2.2. Email író

Az előzőkhez hasonlóan ehhez a funkcióhoz is a listából lehet hozzáférni.

Az email író funkció használatakor a HelperBot végig kíséri a felhasználót a beállítási lépéseken, hogy egy teljesen testreszabott emailt kapjon, kérdései a következők:

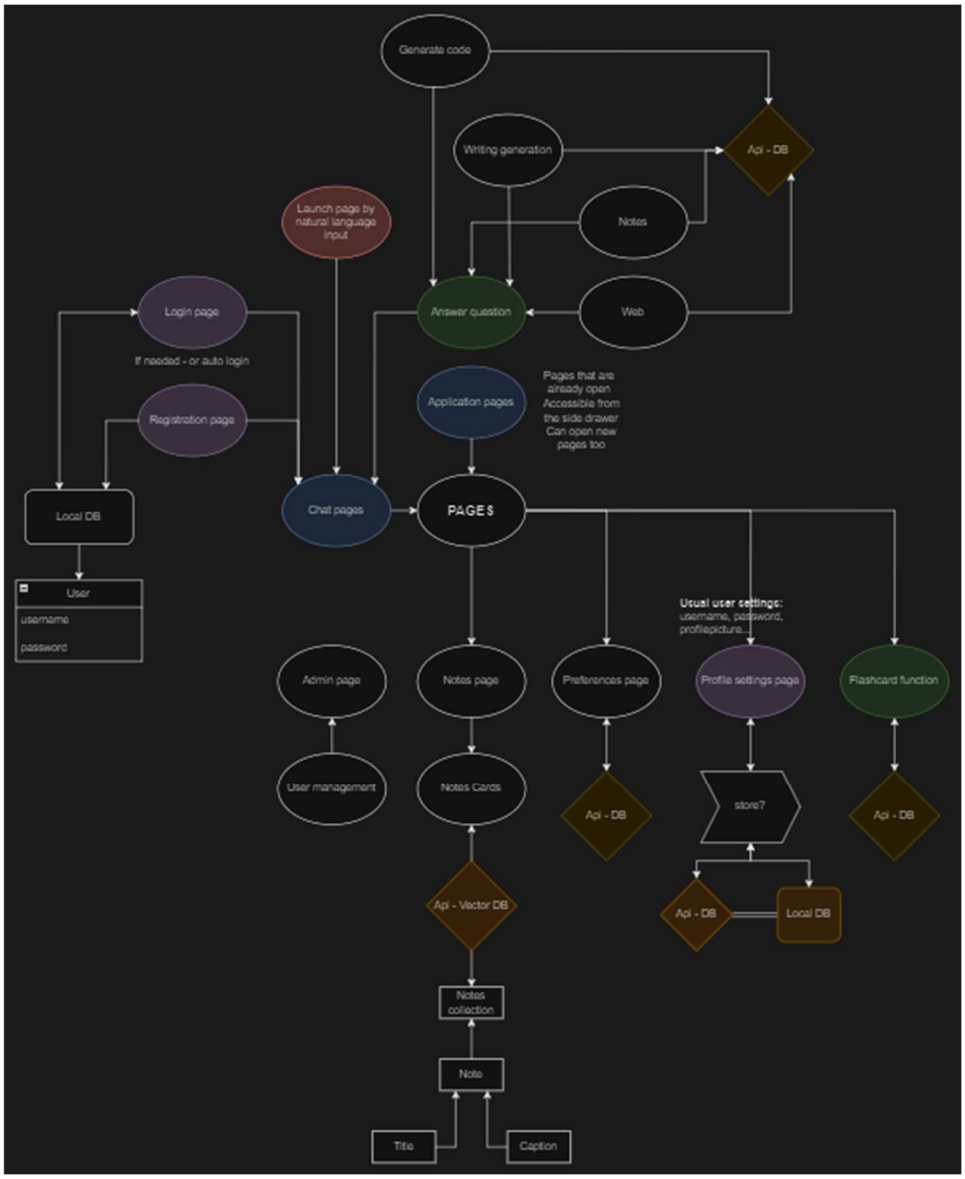
1. Milyen típusú legyen az email?
    a. Formális
    b. Informális
2. Mennyire legyen bonyolult a megfogalmazása?
    a. Skála 1-től 5-ig, 1 a legegyszerűbb és 5 a legbonyolultabb.
3. Hogy hívják a címzettet?
4. Hogy hívják a feladót?
5. Mi a témája?
    a. Rövid, kulcsszavas megfogalmazás a témáról.

Minden választ üzenet formájában kell megadni, ezeket majd a modell fogja értelmezni és feldolgozni. Mint a fogalmazás segítőnél, itt is lehetőség van az email újra generálására, további beállítások megadására.

Mindkét esetben fontos számomra a felhasználók visszajelzése, ezért is tartom jelentősnek az újra generálás lehetőségét. Ezeket a visszajelzéseket egy log-fájlba fogom menteni, hogy hozzájáruljon a későbbi javítások és fejlesztések könnyebb kezeléséhez.

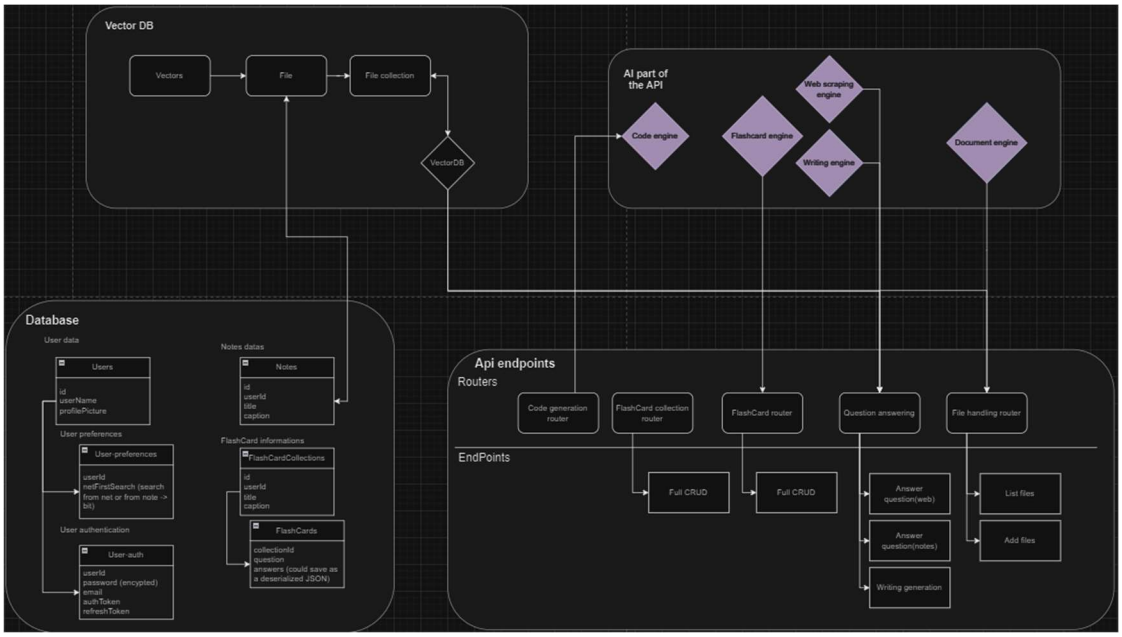
11.RENDSZER DIAGRAMOK

11.1. Magas szintű front-end architektúra

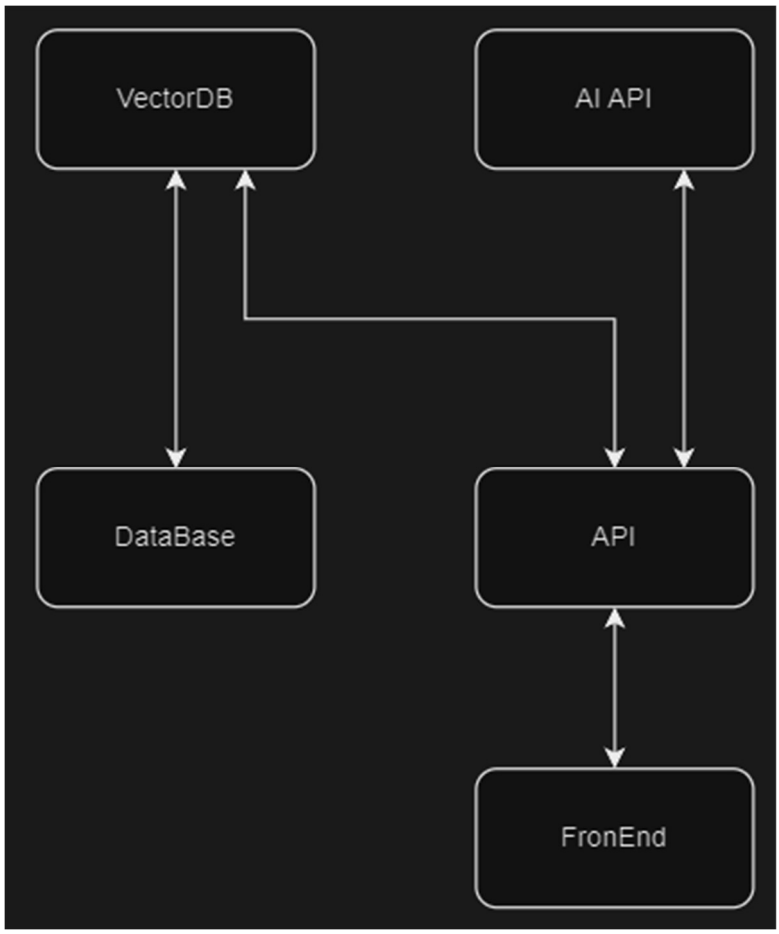


11-1. ábra - Magas szintű front-end architektúra

11.2. Kapcsolatok



11-2. ábra - Kapcsolatok



11-3. ábra - Kapcsolatok 2

12.ÖSSZEHASONLÍTÁS HASONLÓ RENDSZERREL

Ebben a fejezetben az AI-meghajtott tanulást segítő platformot fogom összehasonlítani a Coral AI-al, és a következő szempontok alapján kiértékelni:

- Funkciók
- Támogatott nyelvek
- Összegzés

12.1. Coral AI

12.1.1. Mi a Coral AI?

„A Coral AI-nál [16] arra összpontosítunk, hogy a legjobb mesterséges intelligencia-szoftvert hozzuk létre az Ön fájljainak olvasásához és elemzéséhez.

Platformunk a nagyméretű nyelvi modellek erejét kihasználva segít Önnek gyorsan összefoglalni, lekérdezni és kulcsfontosságú információkat kinyerni az összes fájltípusból.

Akár kutató, szakember vagy diák, a Coral AI egyszerűsíti a dokumentumokkal való interakciót, időt takarít meg, és segít a legfontosabb dolgokra összpontosítani.

Hisszük, hogy kevesebb időt kell töltenie az ismétlődő, értelmetlen feladatokkal, és több időt kell fordítania arra, amiben a legjobb - innovációra, tanulásra és alkotásra.”

12.1.2. Funkciók

A Coral AI a következő funkciókkal szolgál [17]:

- *Összefoglalók készítése*: Összefoglaló készítése hosszú dokumentumokból másodpercek alatt.
- *Keresés az összes dokumentumban*: Korlátlan számú dokumentum feltöltés, keresés.
- *Idézetek beszerzése*: Minden válaszhoz kattintható oldalszám hivatkozás.
- *Címkézett dokumentumok*: Dokumentumok címkézése és csoportokba rendezése.
- *Feltöltés .pdf, .docx, .pptx, .txt + több más formátum*: Több, mint 20 féle fájlformátum támogatás és 200 MB-os méret limit.
- *Hozzáférés 5 élvonalbeli AI modellhez*: A Coral AI több AI modellt kínál, köztük a GPT-4o és a Claude 3.5 Sonnet modelleket.
- *Grafikák és képek elemzése*: Kérdések feltétele és magyarázatok generálása diagramok és képek alapján.
- *Hangfájlok átírása és csevegés*: Hangfájlok átírása, összefoglalása és csevegés.

A funkciókat tekintve vannak hasonlóságok, de eltérések is. Mi nem kizárólag a fájlokkal történő munkálatokra fókuszálunk, hanem input adatok alapján válaszok generálására.

Dézsi Csaba szakdolgozata tartalmazza a pdf feldolgozásra vonatkozó részeket. Több olyan funkciót használ a Coral AI, amit még a tervezés előtt kitűztünk jövőbeli céloknak, mint például a videó formátumok feldolgozása.

12.1.3. Támogatott nyelvek

A Coral AI [18] számos nyelvet támogat alapértelmezetten, melyeket két részre lehet osztani.

Dokumentumok:

Több mint 90 nyelven működik, többek között spanyol, kínai, hindi, francia, arab, német és portugál. Különböző nyelveken írt dokumentumokat lehet feltölteni, különböző nyelveken tehetőek fel a kérdések, és fordításra is alkalmas.

Audió fájlok:

Hang alapú formátumok esetén se rövid a lista, angol, spanyol, francia, német, olasz, portugál, holland, afrikaans, albán, amhara, arab, örmény, asszam, azerbajdzsáni, baskír, baszk, fehérorosz, bengáli, bosnyák, bolgár, bosnyák, breton, bolgár nyelven működik, burmai, katalán, kínai, horvát, cseh, dán, észt, észt, feröer, finn, galíciai, grúz, görög, gudzsarati, haiti, hawaii, hausa, héber, hindi, magyar, izlandi, indonéz, japán, jávai, kannada, kazah, khmer, koreai, laoszi, latin, lett, lingala, litván, luxemburgi, makedón, malagaszkári, maláj, malajálam, máltai, maori, marathi, mongol, nepáli, norvég, norvég nynorsk, okcitán, panjabi, pástu, perzsa, lengyel, román, orosz, szanszkrit, szerb, shona, szindhi, szingaléz, szlovák, szlovén, szomáli, szundan, szuahéli, svéd, tagalog, tadzsik, tamil, tatár, telugu, thai, tibeti, török, türkmén, ukrán, urdu, üzbég, vietnami, walesi, jiddis, joruba.

Jelenleg a programunk kizárólag angol nyelvet támogat, de szintén célunk, ha nem is ennyire tágan, de bővíteni a nyelvi lehetőségeket, ezzel minél több felhasználóhoz eljuttatva az AI-meghajtott tanulást segítő platformot.

12.1.4. Összegzés

Azt be kell látni, hogy a Coral AI egy kiforrott és sokoldalú felület, rengeteg jó megoldással, viszont nem fed le mindent, mivel kizárólag dokumentumok feldolgozására és felhasználására alkalmas. Ebben tudnánk többet mutatni, mint a jelen példában említett konkurensünk, hogy nem lenne limitálva a felhasználó és mindent egy platformon belül érne el.

A Coral AI jó példát mutat arra, hogyan is képzeljük el a munkánkat a jövőben. Tanulmányozása során számos további ötlet született meg, amit mi is tudnánk alkalmazni és kamatoztatni a jövőben, ezzel kibővítve és javítva a weboldalunk használhatóságát, amivel egy komplett és átfogó felhasználói élményt tudnánk biztosítani.

13. VÁLTOZTATÁSOK

A következő fejezetben szembetűnő lehet néhány változtatás a tervezéshez képest, ezeknek különféle okait szeretném összegezni pár mondatban.

13.1. Felhasználói felület

Az egyik leginkább szembetűnő változtatás a funkció választó legörülő lista működése, itt már nincs lehetősége a felhasználónak manuálisan váltani a funkcionalitások között, mivel sikerült megoldani a dinamikus váltásokat, amiket kontextus formájában kap meg a front-end a szerverről az előzetesen bevitt üzenet alapján, tehát ez kizárólag információt ad arról, hogy éppen mire is használjuk a HelperBot-ot.

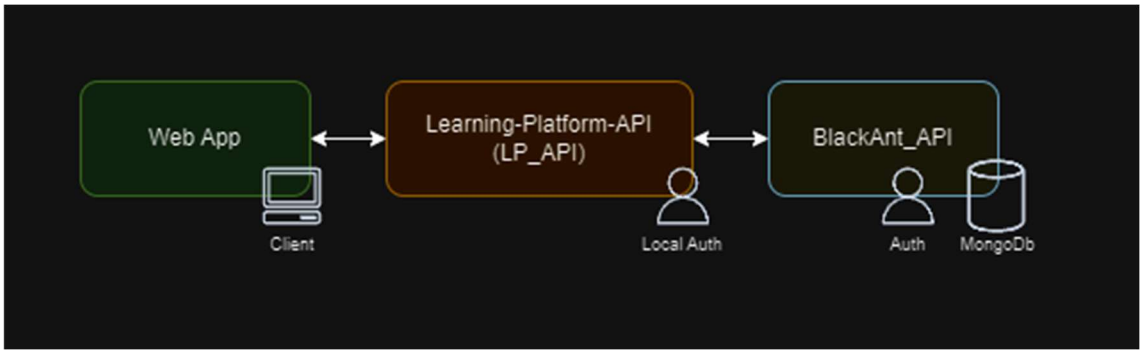
Másik nagy változtatás a fogalmazást segítő funkció részekre bontása. A UI-on továbbra is „wording helper-ként” lesz látható viszont a működése szempontjából fontos volt elszeparálni három részre. Külön kontrollereket kapott minden fogalmazással kapcsolatos funkció, ezt részletesebben bemutatom a következő fejezetben.

Email-ek generálása során elvetettem azt az ötletet, hogy skálán lehessen kiválasztani a megfogalmazás bonyolultságát, mivel ezt a szöveg rövid megfogalmazásakor meg tudjuk adni a modellnek igény szerint. Helyette fontosabbnak tartottam az email hosszának beállítási lehetőségét. A tesztelés során soha nem kaptam kétszer ugyanolyan eredményt éppen ezért nem lett volna érdemi szerepe az újra generálás gombnak, ha a kapott válasz nem elégíti ki a felhasználó igényeit egyszerűen tud pontosítani a bevitt adatokon és egy új üzenetváltás formájában legenerálni a javított értékekkel rendelkező emailt.

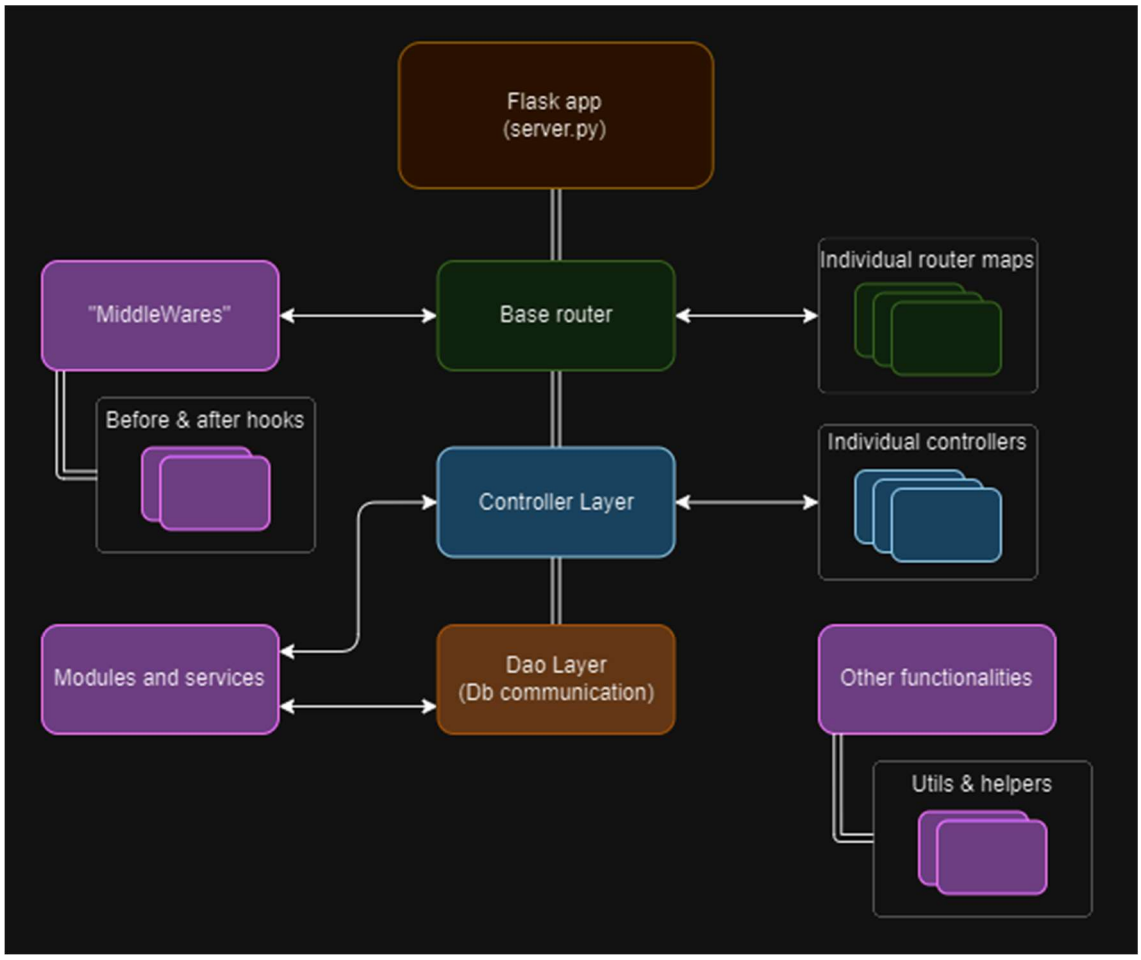
13.2. Back-end

Tervezés során még PostgreSQL adatbázist szerettünk volna használni, de mivel a BlackAnt-en MongoDB állt alapértelmezetten a rendelkezésünkre ezért arra esett a választás. Mivel a szakdolgozati munkánk során nem dolgozunk sok adattal, nincs több száz felhasználónk, ezért nem lett volna indokolt más adatbázis használata, így sem ütköztünk átláthatósági problémákba.

Az adatbázis mellett a szerveren is kellett eszközölni változtatásokat a tervezetthez képest. A következő képek mutatják be a jelenlegi architektúrát:



13-1. ábra – Kapcsolatok 3



13-2. ábra - Kapcsolatok 4

14.MEGVALÓSÍTÁS

A munkafolyamatok bemutatását kezdetben felhasználói szemszögből fogom szemléltetni, így egy átfogó és átlátható képet kaphatunk a program működéséről. A szakdolgozatomat két fő részre lehet osztani, front-end és back-and.

14.1. Felhasználói felület

A felhasználói felület tervezésekor és implementálásakor a legfőbb szempont az volt, hogy a felhasználónak egy informatív, reszponzív és barátságos élményben legyen része a weboldal használatakor.

Legelőször egy üres Quasar projektet hoztam létre, ami legenerált számomra egy sablont, ami alapján sokkal egyszerűbb volt a fejlesztés kezdete, ez a következőket tartalmazta:

- Futtatáshoz szükséges fájlok
- IndexPage
- MainLayout
- Router

Második lépésként telepítésre került a Pinia, aminek segítségével létrehoztam a „chat-store.js-t”, amely tartalmazza a defineStoret az Axios instance és api-k számára, fontosabb változók is itt kerültek tárolásra, amik esetenként több oldalon keresztül is előfordulhatnak.

Ahogy azt már a projekt tervezésénél bemutattam a front-end Axios segítségével kommunikál a szerverrel, minden api response egy Json objektum, amely tartalmazza bejelentkezés esetén a felhasználó számára a jwt-tokent, beszélgetés esetén minden egyes felhasználó által bevitt adatot, a legenerált eredményt és a státuszváltozókat.

14.1.1. Bejelentkező és regisztrációs oldal

A weboldal megnyitásakor legelőször a „LoginPage-re” kerülünk, ahol lehetőség van mind bejelentkezésre és regisztrációra egyaránt. Regisztrációkor az új felhasználónak meg kell adnia a következő adatokat:

- Felhasználónév
- Email cím
- Jelszó
- Jelszó megerősítése

Bejelentkezés esetében:

- Felhasználónév
- Jelszó

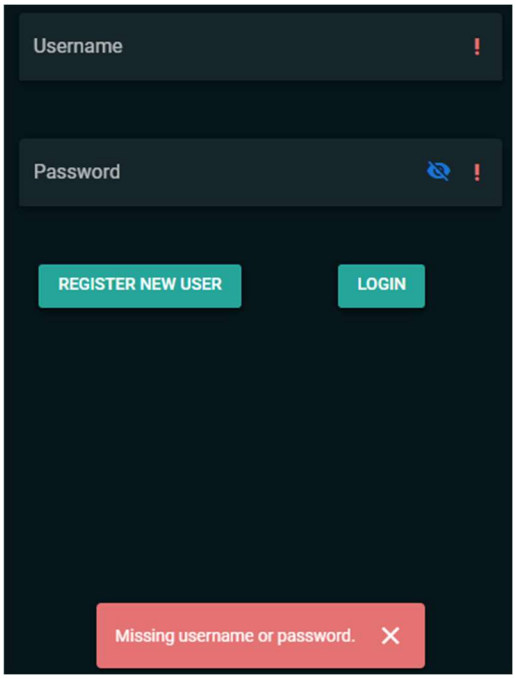
Mindkét esetben a jelszó mező rejtett karaktereket használ így csak a szövegdoboz jobb oldalán elhelyezett „szem” ikon megnyomásakor válik láthatóvá, ezzel megőrizve a jelszavak biztonságát. Természetesen minden jelszó kizárólag titkosítás után kerül az adatbázisba:

```
def hash_password(self, password):  1 usage  KBartu
    # Generate a salt and hash the password
    salt = bcrypt.gensalt()
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), salt)
    return hashed_password
```

14-1. ábra - Jelszó titkosítás

Sikeres regisztráció után az újonnan létrehozott felhasználó bekerül egy MongoDB adatbázisba a titkosított jelszóval.

Szintén mindkét esetben, ha bármelyik mező nincs kitöltve, de mégis megpróbál regisztrálni vagy bejelentkezni a felhasználó egy felugró értesítést kap, tájékoztatva a hiányzó információk megadásáról, ezt minden érintett mező egy piros felkiáltójellel is jelzi. Hasonló értesítést kap, ha a regisztrációnál nem egyezne meg a megismételt jelszó az eredetivel, vagy a kiválasztott felhasználónév és email cím már foglalt.



14-2. ábra - Bejelentkezés

Sikeres bejelentkezést követően minden felhasználó kap egy „jwt-token-t”, aminek segítségével éri el az egyes funkciókat és oldalakat. Ennek hiányában semmihez sincs hozzáférése, kivéve a belépést és a regisztrációt. Minden token 1 órán keresztül biztosítja

a weboldal hozzáférhetőségét, ezt követően újbóli bejelentkezés szükséges. A következő kódrészlet mutatja be az egyszerű, de hasznos megoldást erre a problémára:

```
if (!Cookies.get('jwt')){
  $router.push('/').then(() => {
    location.reload()
  })
}
```

14-3. ábra - JWT token ellenőrzés

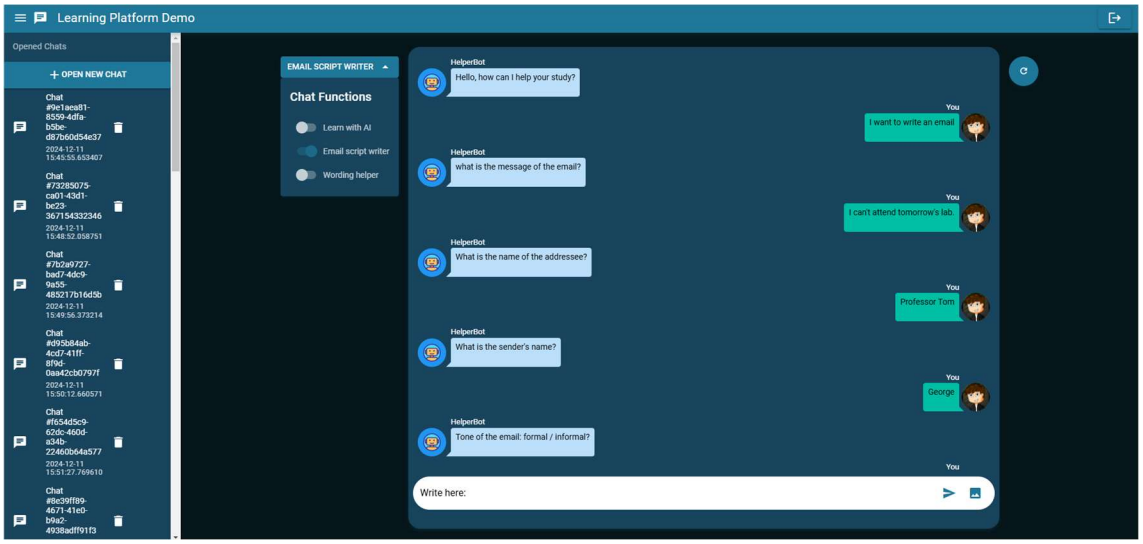
14.1.2. Főoldal

A felhasználó számára sikeres bejelentkezést követően láthatóvá is válik a főoldal, vagyis a ChatBot felülete. Legfelül található a „MainLayout”, ami a Quasar-nak egy fő jellegzetessége és egy keretet alkot az oldalak körül. Ezt átalakítva biztosítok egy kijelentkezési lehetőséget, új beszélgető felületek megnyitását és további funkciók és oldalak eléréséhez egy baloldalról kinyíló menüt.

Az oldal közepén helyezkedik el a projekt egyik legfontosabb eleme, ami összeköti a különböző funkcionalitásokat, a beszélgető felület, ahol a HelperBot egyből üdvözli is a felhasználót, várva az utasításait. Az ablak legalján kapott helyet a beviteli mező, melynek segítségével tudjuk kérdéseinket feltenni. Itt két gomb található, a baloldali az üzenet elküldésére, míg a jobb oldali képek csatolására szolgál, utóbbinak Dézsi Csaba szakdolgozatában lesz jelentősége a „flashcardok-nál”.

Jobb oldalon található egy újra töltés gomb, melynek lenyomásakor az eddigi beszélgetés törlődik így egy új, tiszta felületet kapva folytathatja a felhasználó a beszélgetést.

Bal oldalt kapott helyet a funkció indikátor, amivel nyomon tudjuk követni, hogy épp melyik funkcióját használjuk a weboldalnak, tervezéskor az volt a célom, hogy a felhasználó manuálisan is képes legyen váltani az egyes funkciók között, viszont sikerült ezt dinamikusan megoldani, és jelenleg a weboldal a beküldött szövegek alapján képes eldönteni, hogy éppen mit szeretne csinálni a kérdező.



14-4. ábra - Fő oldal

Beszélgetéskor minden kérdés és válasz egy kattintással a vágólapra másolható, megkönnyítve az utólagos felhasználást, ha rávisszük az egeret bármelyik üzenetre ezt egy „tooltip” fogja jelezni.

Minden egyes kérdés egy api hívás a szerver felé és minden kapott válasz egy api válasz, ha a felhasználó feltette kérdését, egy várakozó indikátort fog látni, amíg meg nem érkezik a válasz a szervertől. Abban az esetben, ha nem sikerülne a szervernek feldolgozni a kérdést erről a felhasználó egy felugró szöveg formájában fog értesítést kapni.

Bejelentkezést követően egy üres felületet kap a felhasználó, ahol egyből fel tudja tenni a kérdéseit. Amint megtörtént az első szöveg bevitele az adatbázisban létrehoz a szerver egy új „conversation” objektumot, aminek az id-ját egyből vissza is küldi a front-endre. Erre az id-ra hivatkozva kerülnek eltárolásra az előzmények.

Minden adott generáláshoz szükséges beérkező adat egy „steps” objektumban kerül eltárolásra, amely ingázik a front- és back-end között. Amíg ezt az AI modell meg nem kapja, minden api hívásnál válaszként érkezik a front-endre, ahol az aktuális input belekerül. Sikeres generálás során minden előzmény eltárolásra kerül a MongoDB adatbázisunkban, amelyekhez minden felhasználó egyszerűen, utólag is hozzáfér. A bal oldalt kinyíló menüben igény szerint folytatni lehet régebbi beszélgetéseket, újakat létrehozni, vagy akár törölni is azokat.

Ahogy azt már említettem a tervezés részben, minden megnyitott beszélgető felület különböző funkciókat tud használni, vagyis nem lesz a felhasználó limitálva, hogy egy adott pillanatban csak az egyikhez fér hozzá.

A forráskód átláthatóságának érdekében igyekeztem minden nagyobb kódrészletet komponensek formájában eltárolni, ami a kódok újra felhasználást is elősegítette, ez

leginkább a szövegbuborékok esetében volt jelentős, mivel nincs limitálva a beszélgetés hossza, ezért elég hamar kialakulhatnak hosszú beszélgetések.

Hosszas fejtörést követően sikerült megoldani, hogy a hosszabb beszélgetéseket reszponzívan lekövesse a beszélgető ablak. Minden buborék egy görgethető listába kerül, ami sajnos magától nem ugrik a legutoljára hozzáadott elemhez, ezért egy egyszerű „ -et” létrehoztam egy id-val, amire hivatkozva minden input megadása és a szerverről érkezett válasz esetén manuálisan legörget a legaljára.

14.2. Szerver oldali megvalósítás

A szerver implementációja kívül esett a hatáskörömön, de volt nekem is részem annak fejlesztésében. Rövid összefoglaló a szerverről:

Az AI modellek miatt természetesen Python szervert használunk, ami Flask API segítségével kommunikál a front-end-el.

A szerver lelke a „FunctionChooser” (Funkció választó), melyet Pálosi Ákos alkotott meg. Ez egy olyan mesterséges intelligencia alapú middleware-ként lett implementálva, amely minden bejövő kérést átirányít a megfelelő végpontra a megadott input alapján. Ha a kérés a „FORCE_ROUTE=true” paramétert tartalmazza, akár lekérdezési paraméterként vagy a JSON body-ban, akkor az átirányítás megtörténik. Az átirányítás egy függvénytérkép segítségével valósul meg, melyet a következő kép illusztrál az email vezérlőhöz tartozó függvénytérképpel, erről bővebben Pálosi Ákos szakdolgozatában esik szó:

```
Context.EMAIL.value: {
    "controller": EmailController.EmailController,
    "pipeline_desc": "Drafting an email based on multiple parameters",
    "steps": {
        "1": {
            "step_input_desc": "message of the email",
            "step_input_name": "message",
            "examples": [
                {
                    "example_prompt": "Write an email about quantum physics",
                    "example_answer": "quantum physics",
                }
            ],
            "url_to_redirect": "EmailContext_step1"
        },
        "2": {
            "step_input_desc": "The person who the email is addressed to",
            "step_input_name": "addressee's name",
            "examples": [
                {
                    "example_prompt": "write me an email from Emily to Joseph",
                    "example_answer": "Joseph",
                }
            ],
            "url_to_redirect": "EmailContext_step2"
        },
        "3": {
            "step_input_desc": "The person who is sending the email",
            "step_input_name": "sender's name",
            "examples": [
                {
                    "example_prompt": "write me an email from Walter to Jasmine",
                    "example_answer": "Walter",
                }
            ],
            "url_to_redirect": "EmailContext_step3"
        },
```

14-5. ábra - Függvénytérkép példa 1

```
        "4": {
            "step_input_desc": "The tone of the email, the choices are formal, informal",
            "step_input_name": "tone",
            "examples": [
                {
                    "example_prompt": "Write me funny letter about history",
                    "example_answer": "informal",
                }
            ],
            "url_to_redirect": "EmailContext_step4"
        },
        "5": {
            "step_input_desc": "The length of the email in number of words.",
            "step_input_name": "words",
            "examples": [
                {
                    "example_prompt": "Write an email in 200 words",
                    "example_answer": "200",
                }
            ],
            "url_to_redirect": "EmailContext_step5"
        }
    },
    "after_steps": "EmailContext_generate_email",
    "functions": {},
},
```

14-6. ábra - Függvénytérkép példa 2

Minden függvénytérkép-hez tartozik egy kontroller, ami az ott látható függvényeket tartalmazza, nekem a funkcióim megvalósításához három kontrollert volt szükséges létrehoznom:

- *EmailController* – email generálás és lépései.
- *NovelComplementController* – Szöveg kiegészítés és lépései.
- *NovelCorrectionController* – Szöveg nyelvtani hibáinak javítása és szöveg újra fogalmazása.

A következő EmailController példa egy rövid részletet mutat be a kontrollerek felépítéséről:

```python
class EmailController(StepsBaseController):  10 usages  KBartu +2 *
    def __init__(self, request):  dezsi_csaba
        super().__init__(request)

    @staticmethod  1 usage (1 dynamic)  KBartu
    def validate_step(step: int, input: str) -> bool:
        return True

    def EmailContext_step1(self):  KBartu +1 *
        args = self._req.args
        steps = args["steps"]
        input = args["input"]
        conversation_id = args["conversation_id"]
        question = "what is the message of the email?"
        return jsonify({
            "question": question,
            "steps": json.loads(steps),
            "current_step": "1",
            "context": "email",
            "conversationId": conversation_id,
            "input": input,
        })

    def EmailContext_step2(self):  KBartu +1
        args = self._req.args
        conversation_id = args["conversation_id"]
        steps = args["steps"]
        input = args["input"]
        question = "What is the name of the addressee?"
        return jsonify({
            "question": question,
            "steps": json.loads(steps),
            "current_step": "2",
            "context": "email",
            "conversationId": conversation_id,
            "input": input,

        })
```

14-7. ábra - Kontroller példa

14.2.1. EmailController

Az EmailController tartalmazza az email generálás menetéhez szükséges függvényeket, amelyek sorrendben a következők:

- EmailContext_step1 – Üzenet rövid leírása
- EmailContext_step2 – Címzett neve
- EmailContext_step3 – Feladó neve
- EmailContext_step4 – Hangvétel (formális / informális)
- EmailContext_step5 – Email hossza (szavak száma)
- EmailContext_generate_email – A felhasználó adatait integrálja a „prompt-ba”, amit átad a Llama3-as modellnek, generálás után visszaküldi a végső eredményt a felhasználónak.

Az email generálás során használt prompt template:

- Input adatok:
  - "subject", "addressee", "sender", "tone", "num_of_words"
- Template:
  - Write an email in maximum {num_of_words} words in a paragraph form with the following details.
  - Avoid generating restricted words.
  - Return only the answer.
  - Addressee: {addressee}
  - Sender: {sender}
  - Email tone: {tone}
  - Email Content:
  - Subject: {subject}
  - Dear {addressee},

Minden lépésnél addig várakozik, amíg meg nem kapja a következő választ, amint minden szükséges információ a birtokába kerül megkezdődik a generálás. A FunctionChooser segítségével egyszerre több lépés is elvégezhető, viszont egyenként is használhatók.

14.2.2. NovelComplementController

A NovelComplementController tartalmazza a szöveg kiegészítés menetéhez szükséges függvényeket:

- NovelComplementContext_step1 – Eredeti szöveg megadása
- NovelComplementContext_step2 – Rövid leírás a kiegészítendő szövegről
- NovelComplementContext_step3 – A kiegészítés hossza

- NovelComplement – Kiegészített szöveg generálása után visszaküldi a felhasználó számára az eredményt.

A szöveg kiegészítés lényegében ugyan úgy működik, mint az email generálás, a felhasználó lépésenként, vagy egyetlen szöveg formájában megadhatja a modell számára szükséges információkat.

A szöveg kiegészítés során használt prompt template:

- Input adatok:
  - "text", "complement_text", "words"
- Template:
  - Keep the original text.
  - Complement the text following the standards in {words} words.
  - Text: {text}
  - Complement: {complement_text}

14.2.3. NovelCorrectionController

Ez nagyon hasonló, mint az előbb említett kontrollerek, viszont részben eltér. Itt nincsenek lépések, mivel nincs szükségünk több adatra a felhasználótól, kizárólag a szövegre, ezáltal teljes mértékben el tudjuk hagyni a lépéseket és egyből az eredményre koncentrálhatunk:

- NovelGrammarCorrection – Ez a függvény a bemenetként kapott szöveget a „prompt-tal” együtt átadja a modellnek, amely visszaküldi a felhasználó számára a már kijavított, nyelvtani hibáktól mentes szöveget.
- NovelRephrase – Ebben az esetben a felhasználó egy átfogalmazott szöveget kap vissza, lényegbeli eltérések nélkül és megőrizve a szöveg hosszát.

A nyelvtani hibák kijavításához használt prompt template:

- Input adatok:
  - "text", "num_of_words"
- Template:
  - Correct the following text frammatically woithout changing the meaning of the text, the generated text shouldn’t be longer than {num_of_words} words.
  - Return only the corrected text without any explanation or other compliments.
  - The text for the grammatical correction:
  - {text}

A szöveg átfogalmazásához használt prompt template:

- Input adatok:
    - "text"
- Template:
    - Rewrite the text following the standards.
    - Do not make the text longer.
    - Text: {text}
    - START OF THE GENERATED TEXT:

14.2.4. Map-ok

Minden kontrollernek szüksége van egy „map” fájlra, ami tartalmazza az adott végpontokat az összes lépéshez, a FunctionChooser ez alapján fogja tudni, hogy melyik végpontok közül választhat. Minden kontroller map fájlja a RoutingMap-ban kerül definiálásra.

14.2.5. AI modell

A funkcióim megvalósításához a tervezésnél már említett Llama3-as modellt használtam, pontosabban a Meta Llama3 8B Instruct modellt. Implementálás során egy notebook fájlban hoztam létre, ami lényegesen leegyszerűsítette az egyes funkciók tesztelését. Minden funkcióhoz létrehoztam egy „prompt template-t”, ami tartalmazza a modell számára az utasításokat és a felhasználótól érkezett input adatokat, ez az előző részekben volt látható. Minden prompt template megírásakor ügyeltem arra, hogy minél egyszerűbb és célravezetőbb legyen a megfogalmazása, a minél kevesebb utómunka reményében. Sajnos nem minden esetben sikerült olyan eredményeket visszakapnom, ami alkalmas lenne a felhasználóhoz való visszajuttatásra, ezért az eredmények egy részét le kellett vágnom, hogy csak a releváns információkat kapják meg, ennek megkönnyítése céljából olyan válaszokat generáltatok a modellel, hogy fix pontokat kapjak, amik alapján megtörténhet a szöveg tartalmi kiemelése.

Minden egyes generálás során az elkészült szöveg átesik egy felülvizsgálaton, ami tartalmazza az említett felesleges részek levágását, és mivel az eredményeket leginkább diákok fogják kapni tanulmányi céllal, ezért fontosnak tartottam, hogy egy „forbidden_phrases” nevű listában összegyűjtsem a nemkívánatos szavakat, amik szűrésre kerülnek, ezzel védve a felhasználókat.

A modell a következő paraméterekkel és beállításokkal rendelkezik:

```python
def load_model(self):  1 usage  Ákos
    os.environ['HF_HOME'] = '/models'
    compute_dtypes = getattr(torch, "float16")
    quant_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_use_double_quant=False,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_compute_dtype=compute_dtypes
    )
    model_id = self.model_name
    #"meta-llama/Meta-Llama-3-8B-Instruct"
    access_token = "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"

    tokenizer = AutoTokenizer.from_pretrained(model_id, token=access_token)
    model = AutoModelForCausalLM.from_pretrained(model_id,
                                                 token=access_token,
                                                 quantization_config=quant_config, device_map="auto")
    pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max_length=1000, length_penalty=1.5)
    hf = HuggingFacePipeline(pipeline=pipe)
    self.hf = hf
```

14-8. ábra - Modell betöltés

A modell konfigurálásakor törekedtem a gyorsaságra és a helyes eredmények visszaadására, de kompromisszumot kellett kötnöm. Amikor jónak mondható eredményt kaptam sajnos erőteljesen lelassult a futási idő, ellenkező esetben gyorsan kaptam hibás eredményeket. Jelen állapotban lassan, de használható értékekkel szolgál a modell.

Az egyes kontrollerekből érkező „prompt_template” és „input_variables” a „generate” függvénybe kerül, itt adom át azokat a modell számára, ami a meghívás pillanatában megkapja a szükséges információkat a kiválasztott funkció válaszának generálásához:

```python
def generate(self, input_variables:list, prompt_template:str, input_data:dict):  Ákos +1
    self.logger.error("---------------------")
    self.logger.error(prompt_template)
    self.logger.error(input_data)
    self.logger.error(input_variables)
    self.logger.error("---------------------")

    prompt_template = PromptTemplate(
        input_variables=input_variables,
        template=prompt_template
    )

    chain = prompt_template | self.hf
    ...
    generation_result = chain.invoke(input_data)
    generation_content = generation_result.strip()

    forbidden_phrases = []
    pattern = r'|'.join([re.escape(phrase) for phrase in forbidden_phrases])
    generation_content = re.sub(pattern,  repl: '', generation_content, flags=re.IGNORECASE)

    return generation_content
```

14-9. ábra - Modell generálás

A szerver indításakor a modell egyből betöltődik, megakadályozva a felület lelassulását és ezzel felgyorsítva a felhasználói élményt.

15.TESZTELÉS

Tesztelés során nem szeretnék kizárólag számokat feltűntetni, ezért példákon keresztül fogom tesztelni és bemutatni a program működését, mivel a Llama3-as modellem nem feltétlen szokott egymás után két egyforma választ generálni, ezért nehéz lenne prediktálni a válaszokat, vagy különféle metrikákkal bemutatni azokat. A tesztelés felépítése a következő lesz:

- Bemeneti paraméterek, lépések válaszai.
- Kapott válasz bemutatása.
- Konklúzió

Mivel saját számítógépem videókártyáját használja a modell, ezért a generálási időt nem tartom releváns adatnak, éppen ezért ez nem kerül prezentálásra.

Először a modell működését tesztelem a funkciókon keresztül, utána a felhasználói felületet.

15.1. Email generálás

Az email generálást egy olyan példán keresztül fogom tesztelni és bemutatni, ahol egyenként megyek végig a kérdéseken, így tesztelve a kontrollerem is, ahol ezek feldolgozásra kerülnek. Akkor tudjuk, hogy a funkció választó helyesen működik, hogy a kontextus nélkül beírt szövegünk alapján („I want to write an email”) jelen esetben a funkció indikátort email-re állította a response alapján.

Bemenetek, kapott válaszok:

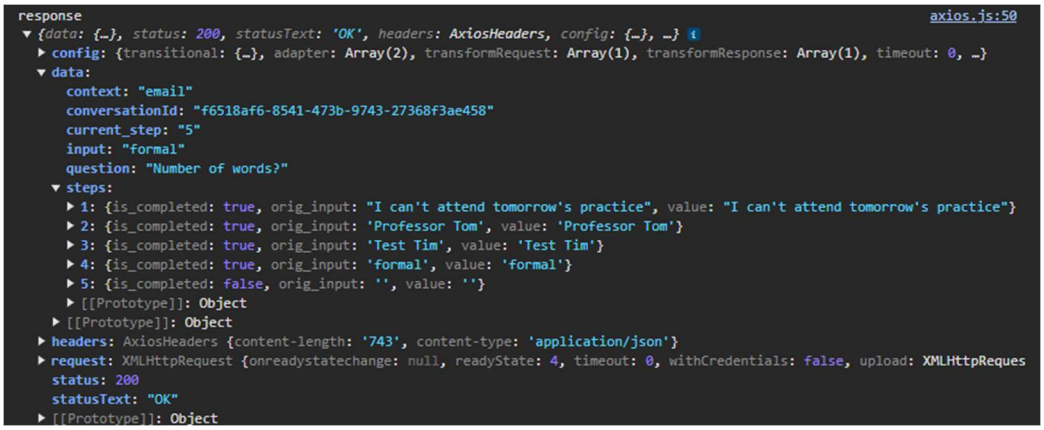
- Bemenet: „I want to write an email” – Válasz: „What is the message of the email?”
- Bemenet: „I can't attend tomorrow's practice” – Válasz: „What is the name of the addressee?”
- Bemenet: „Professor Tom” – Válasz: „What is the sender’s name?”
- Bemenet: „Test Tim” – Válasz: „Tone of the email: formal / informal?”
- Bemenet: „Formal” – Válasz: „Number of words?”
- Bemenet: „30”

Generált válasz:

- „Dear Professor Tom, I regret to inform you that I will be unable to attend tomorrow's practice. I will keep you updated on my availability. Thank you for your understanding. Best regards, Test Tim”

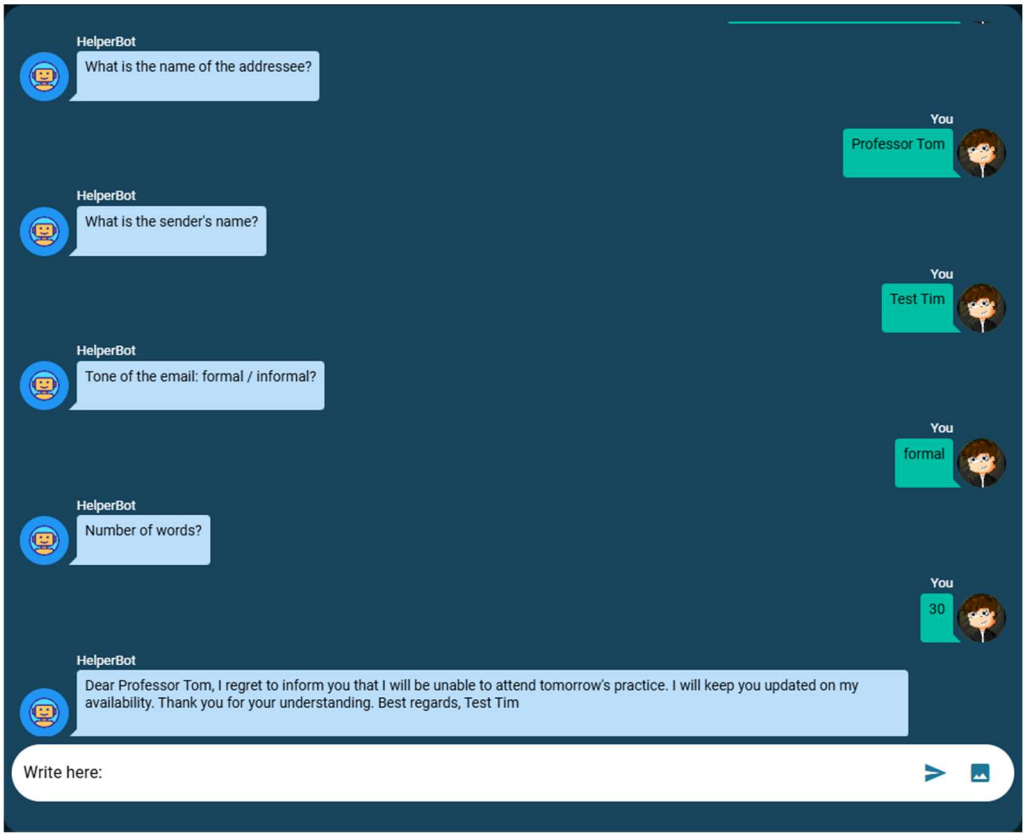
Ahogy azt a válaszban is láthatjuk, a modell megfelelően működött, a bevitt adatok alapján legenerálta a szöveget, megtartva a címzettet és a feladót, betartotta a formai követelményeket, arról írt, amiről szerettük volna és a szavak száma is megfelelő. Itt kiemelném, hogy a megfogalmazás miatt egy – két szó eltérés lehetséges a kívánttól és

az üdvözlés nem számít bele a 30 szóba, mivel azt manuálisan adtam hozzá a template-hez, hogy a modell tudja, hogy honnantól generálja az emailt.



15-1. ábra - Email teszt 1

Ezen a képen egy példa látható az api válaszokra, jelen esetben a szöveg hosszának beállítása előtti állapotában. Minden lépés elmentésre kerül, a Json objektum véglegesítése után kerülnek be az adatok a template-be, amit a Llama3 kiértékel és feldolgoz. A lépéseket, vagyis az email generálás tesztelésének menetét a következő kép illusztrálja:



15-2. ábra - Email teszt 2

A képen azt is megfigyelhetjük, hogy sikeresen működik a túlcsordulás ellen létrehozott funkcióm és automatikusan legörgette a beszélgetést a legaljára.

15.2. Szöveg kiegészítés

A szöveg kiegészítés menete nem tér el az email generálástól, ugyan úgy lépésenként fogom tesztelni és a végeredményt kiértékelni. Egy rövidebb történetet fogok megadni a modellnek, ami alapján kiderül, hogy tudja értelmezni a szövegkörnyezetet és betartja a szavak számát.

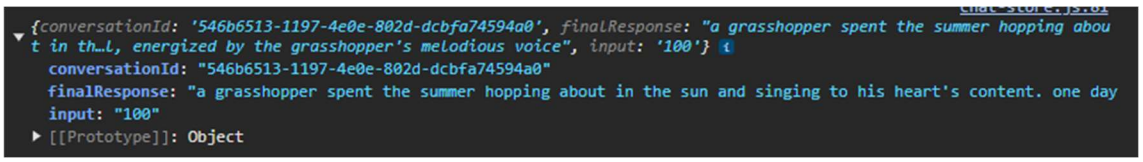
Bemenetek, kapott válaszok:

- Bemenet: „Complete my novel with extra sentences, A grasshopper spent the summer hopping about in the sun and singing to his heart's content. One day, an ant went hurrying by, looking very hot and weary.” – Válasz: „Enter your short complement.”
- Bemenet: „What did the ant do?” – Válasz: „Enter the number of words to complete your novel.”
- Bemenet: „100”

Generált válasz:

- „a grasshopper spent the summer hopping about in the sun and singing to his heart's content. one day an ant went hurrying by looking very hot and weary.
- The ant stopped for a moment, took a deep breath, and said, "Oh, you're a grasshopper, aren't you? I've been working hard all day, gathering crumbs for my anthill. I'm exhausted. But I must admit, your singing is lovely. It's a welcome respite from my toilsome labor." The grasshopper smiled and continued to sing, feeling grateful for the ant's appreciation. The ant listened, feeling refreshed, and then returned to his anthill, energized by the grasshopper's melodious voice”

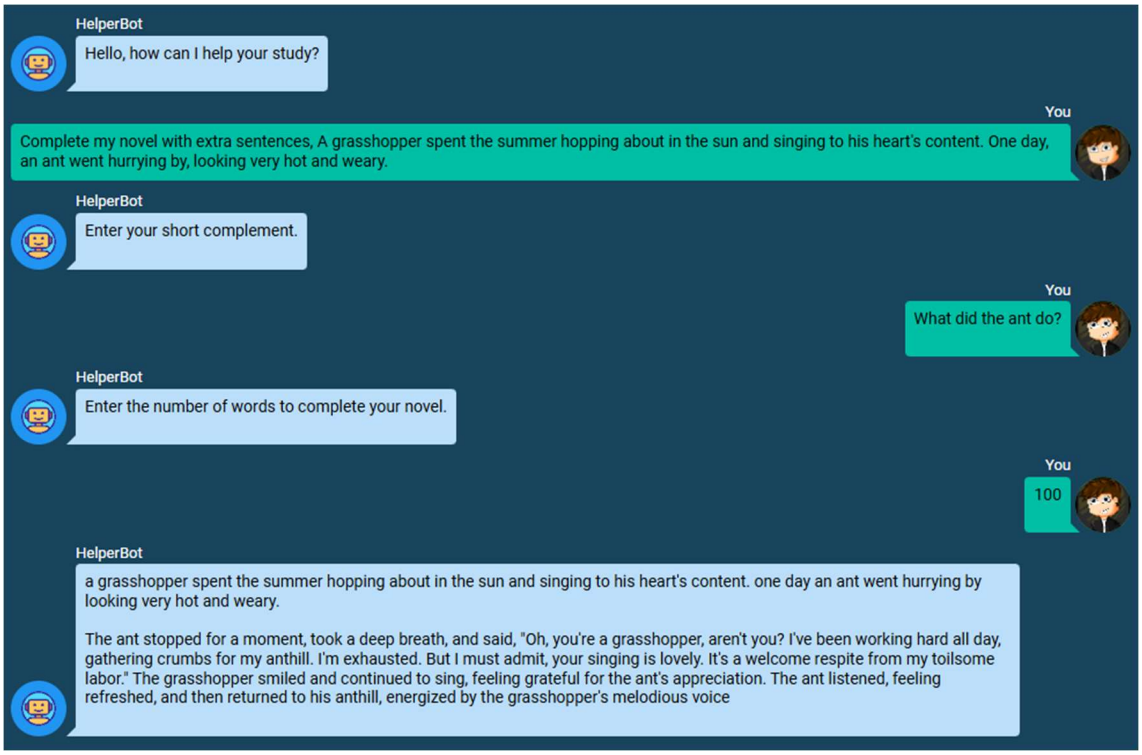
Ahogy az a visszakapott válaszból is látszik egyetlen probléma, hogy a mondatot nem nagy betűvel kezdi, miközben az inputban nagy „A” -val kezdtem a mondatot, ezt leszámítva helyesen felismerte a szövegkörnyezetet és ennek megfelelően kiegészítette a kérdés alapján, betartva a megadott szavak számát.



15-3. ábra - Szöveg kiegészítés teszt 1

Az email generálás tesztelésekor mutattam api példát a lépésekre, ezért most a végső válasz eredményét prezentálom. Ebben az esetben az utolsó kérdésre adott választ láthatjuk az id és a végeredmény mellett.

Hogyan is nézett ki ez a teszteset a front-enden:



15-4. ábra - Szöveg kiegészítés teszt 2

15.3. Nyelvtani hibák javítása

A nyelvtani hibák javítása esetén egyszerűbb a dolgom, mivel csakis egy bemenetre van szükség a kívánt végeredmény eléréséhez.

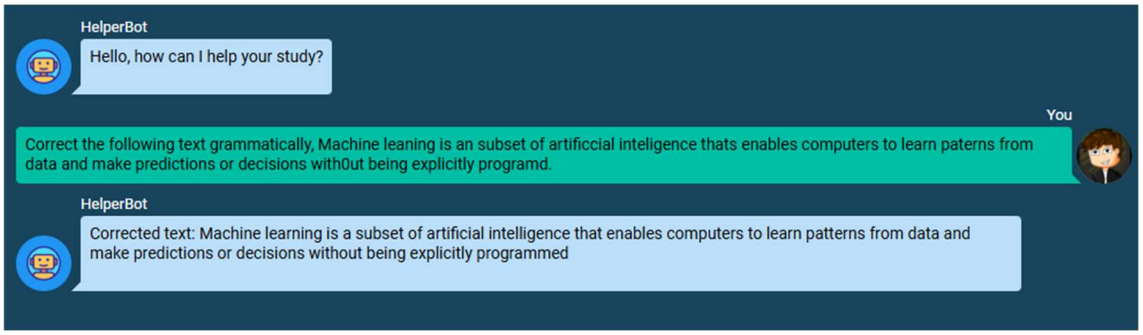
Bemenet:

- „Correct the following text grammatically, Machine leaning is an subset of artificcial inteligence thats enables computers to learn paterns from data and make predictions or decisions with0ut being explicitly programd.„

Generált válasz:

- „Corrected text: Machine learning is a subset of artificial intelligence that enables computers to learn patterns from data and make predictions or decisions without being explicitly programmed”

A következő kép bemutatja a nyelvtani hibák javításának menetét felhasználói szemszögből:



15-5. ábra - Nyelvtani hibák teszt 1

Elmondhatjuk a generált válaszról, hogy a szó elírásokat könnyedén felismeri a modell és javítja azokat bármiféle nehézség nélkül, viszont a következő tesztesetben nehezebb dolga lesz, mivel már nyelvtani hibákkal is meg kell küzdenie.

Bemenet:

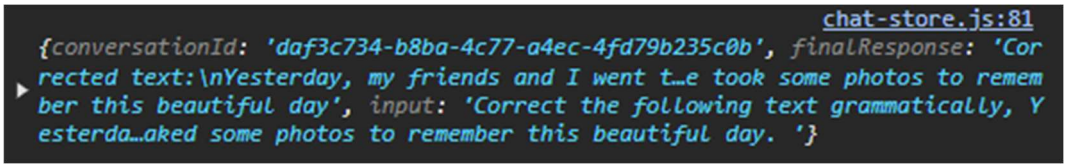
- „Correct the following text grammatically, Yesterday, me and my friends goes to the park for a picnic. We bringed sandwiches, chips, and juice. The weather was very nice and the sun was shinning bright. Everyone enjoy to play games and talking. At the end, we taked some photos to remember this beautiful day. „

Generált válasz:

- „Corrected text: Yesterday, my friends and I went to the park for a picnic. We brought sandwiches, chips, and juice. The weather was very nice, and the sun was shining brightly. Everyone enjoyed playing games and talking. At the end, we took some photos to remember this beautiful day”

15-6. ábra - Nyelvtani hibák teszt 2

Bonyolultabb hibáknál is a várt eredményt hozta, gond nélkül elbánt a nyelvtani hibákkal is. A következő kép a szerverről érkező api válasz, mivel nincsenek lépések, ezért kizárólag a „conversationId-t”, a megadott inputot és a végső eredményt tartalmazza:



15-7. ábra - Nyelvtani hibák teszt 3

15.4. Szöveg átfogalmazás

Szöveg átfogalmazásakor sincs több adatra szükség, mint magára a szövegre.

Bemenet:

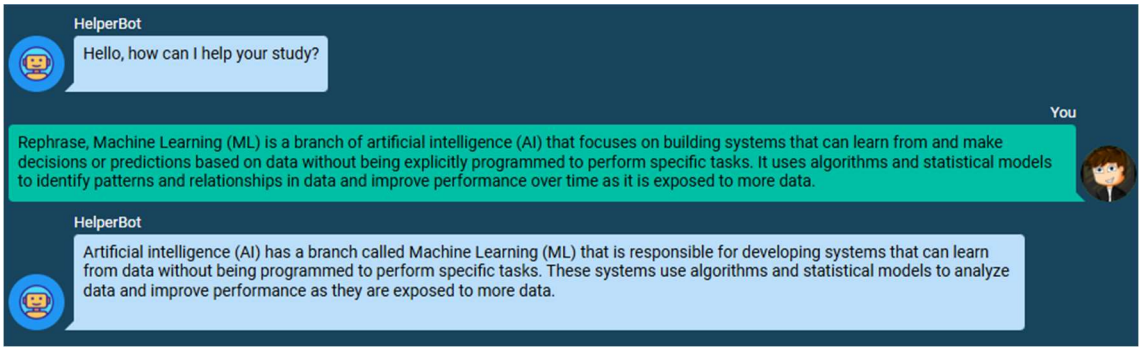
- „Rephrase, Machine Learning (ML) is a branch of artificial intelligence (AI) that focuses on building systems that can learn from and make decisions or predictions based on data without being explicitly programmed to perform specific tasks. It uses algorithms and statistical models to identify patterns and relationships in data and improve performance over time as it is exposed to more data."

Generált válasz:

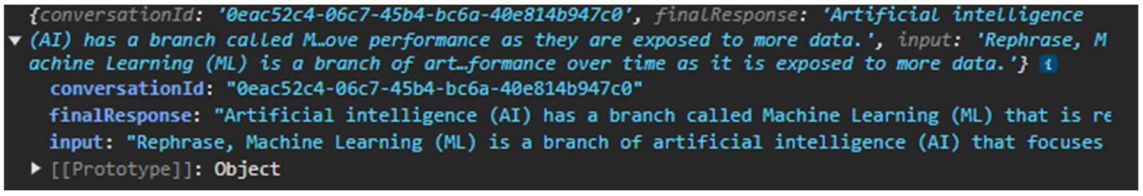
- „Artificial intelligence (AI) has a branch called Machine Learning (ML) that is responsible for developing systems that can learn from data without being programmed to perform specific tasks. These systems use algorithms and statistical models to analyze data and improve performance as they are exposed to more data.

Ebben a tesztesetben is látható, hogy a modell sikeresen megértette a feladatát és tartalmi változtatás nélkül újra megfogalmazta a kívánt szöveget.

Felhasználói oldalon a már megszokott struktúrával találkozhatunk, ettől az api válasz sem kivétel:



15-8. ábra - Átfogalmazás teszt 1

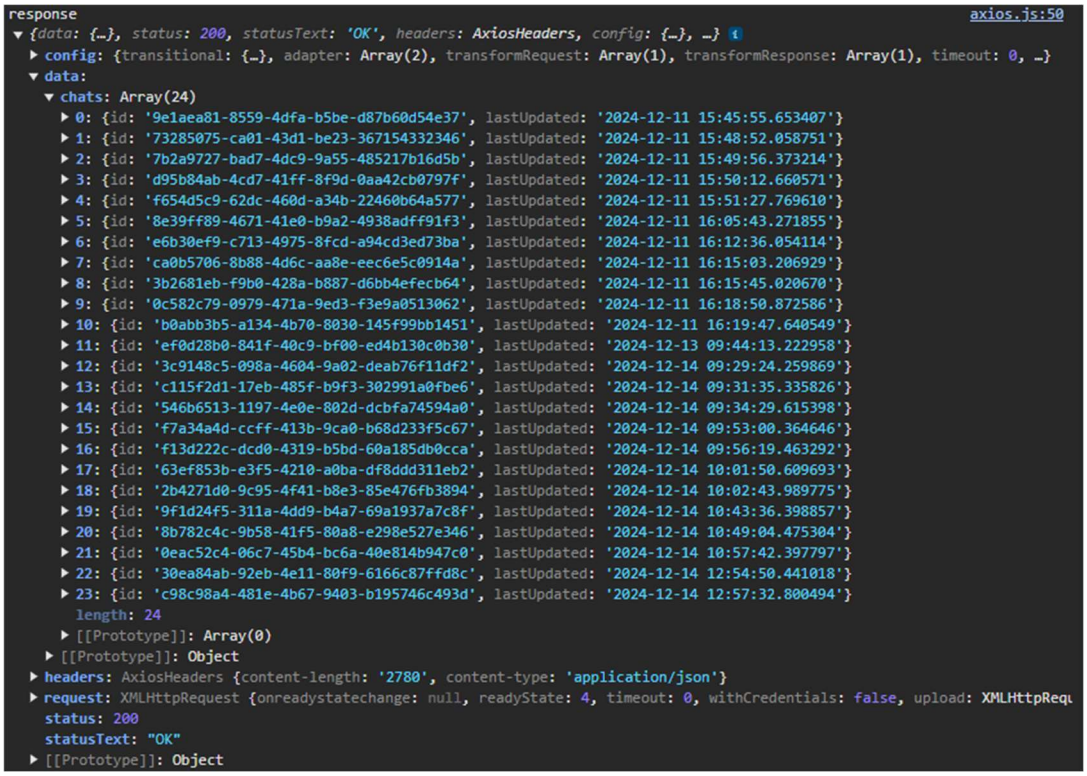


15-9. ábra - Átfogalmazás teszt 2

15.5. Felhasználói felület

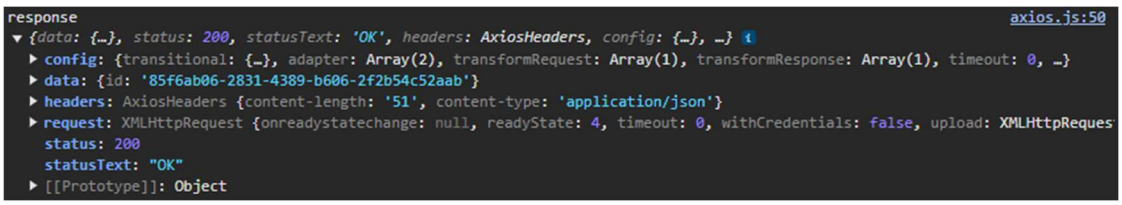
A funkciók tesztelése során maga a beszélgető felület is tesztelésre került, ezért ebben a fejezetben erre nem fogok bővebben kitérni. Jelen esetben a beszélgetési előzményekre, új beszélgetés létrehozására és a törlésre fókuszálok, melyek sikeres működésére a legjobb példa a szerverről érkező válaszok.

Beszélgetési előzmények:



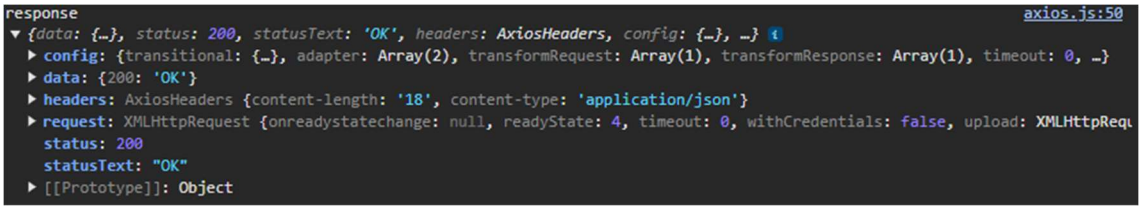
15-10. ábra - Beszélgetési előzmények teszt

Beszélgetés létrehozása:



15-11. ábra - Beszélgetés létrehozás teszt

Beszélgetés törlése:



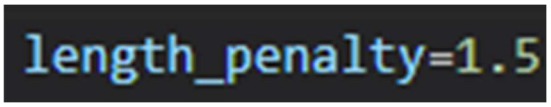
15-12. ábra - Beszélgetés törlés teszt

16.NEHÉZSÉGEK, FELMERÜLT PROBLÉMÁK

16.1. Llama 3 konfiguráció

A legnagyobb problémát a Meta Llama 3 modell konfigurációja jelentette. Sajnos, mint minden modellt számos tényezővel számolni kell, ha megfelelően szeretnénk beállítani a paramétereket, ez az én helyzetemben sem volt másképp. Kezdetekben elég hamar sikerült rátalálnom egy akkor még használhatónak vélt beállításra, amely a helyzethez képest viszonylag gyorsan képes volt válaszokat generálni, viszont elég hamar problémákba ütköztem.

- A modell nem tudta rendesen kezelni, ha olyan feladatokat kapott, melyek megszabják a kimeneti érték hosszát. Legtöbb esetben vagy figyelmen kívül hagyta a szöveg hosszát érintő utasítás részt, vagy egy számára szimpatikusnak vélt szövegrészletet ismételt addig, amíg el nem érte a megadott limitet, például email generálás esetén ez a végső elköszönést jelentette. Nagyon sok megoldással próbálkoztam, minden elérhető paramétert átállítottam egyenként, de egyik változtatás sem hozta meg a kívánt eredményt, a jelenlegi 4bit-es verzió helyett 8bit-re is átállítottam, hátha „okosabban” fogja értelmezni az utasításokat, de ez sem segített. Sajnos egy olyan megoldást találtam, ami bár segít a problémán, viszont szembe ötlően lelassítja a program futását, ez azt jelenti, hogy az a folyamat, ami eddig 30 másodpercig tartott jelenleg 2-3 percet is igénybe vehet. A paraméter a következő képen látható:



16-1. ábra - Nehézségek példa

- Másik nagy probléma a prompt template-k értelmezése volt. Többféleképpen is próbálkoztam, de azt vettem észre, hogy nem tesz jót a modellnek, ha túl részletes leírást adok meg neki. Volt olyan eset, amikor válaszolt a kérdésemre, viszont a következő híváskor már ő kérdezett tőlem. Ennek megoldására azt találtam ki, hogy inkább egyszerűbben értelmezhető utasításokat adok meg és utólagosan korrigálom az eredményeket.

16.2. Limitációk

Mind a Funkcióválasztónál és a fő modellem a Llama3-nál is érezhetőek teljesítmény béli problémák, viszont most leginkább a Llama3-ra térnék ki. Ez több tényezőnek is betudható, legyen ez akár hardveres vagy szoftveres megkötés.

Saját számítógépemen folyt a fejlesztés, éppen ezért a legelső szembeötlő negatívum a modell tárhelyigénye volt. Igaz, hogy a Llama3 8B instruct modellt használtam, ami

fokozottan kevesebb tárhelyet foglal el, mint a teljes méretű társa, viszont így is könnyedén belecsúszott a több tíz gigabájtba, ami a Docker-rel együtt megrémítő számokat volt képes produkálni. Tesztelés során nem egyszer szembesültem azzal, hogy már csak egy – két gigabájt tárhelyem maradt a meghajtón, ekkor mindig le kellett állítanom a folyamatokat és kétségbeesetten felszabadítani egy kis helyet.

A második nehézség, ami elég hamar felfedte magát a modellek sebessége. Ezt már az előző pontban bemutattam, hogy ez nem feltétlenül csak hardveres probléma, viszont az is megnehezítette a fejlesztési folyamatokat, hogy a projekt összes modellje a videókártyát használja a visszaadott értékek generálásához. Nincs rossz videókártya a számítógépemben, viszont csak ez az egyetlen elektronikus eszközöm volt képes futtatni a programot lokálisan, éppen ezért helyhez is voltam kötve.

17.TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Véleményem szerint rengeteg potenciált tartogat még magában az AI tanuló platform, az eddig megvalósított program tökéletes alapot nyújt a bővítéshez és a továbbfejlesztéshez. Természetesen jelen esetben az időkorlát is jelentős szerepet játszott a platform limitált működésében, viszont kibővített, komplexebb megoldásokkal lényegesen lehet javítani a teljesítményén és sokszínűségén.

17.1. UI

Elsősorban a felhasználói felületre térek ki, itt a már jól ismertetett Vue framework-öt használtam a Quasar-ral együtt. Jelen pillanatban ez tűnt a legalkalmasabbnak a szakdolgozatom implementálásához, mivel ezzel volt már előzetes ismeretem. Abban az esetben, ha nem sürgetett volna a határidő biztosan más framework-öt használtam volna, például a React-et. A React-tel sokkal komplexebb és esztétikusabb megjelenést tudtam volna adni a felhasználói felületnek, viszont nem lett volna időm megtanulni a működését ennyi idő alatt és érdembeli programot felmutatni.

17.2. Adatbázis

Lokális teszteléshez és az éles verzióhoz is MongoDB adatbázist használunk, mivel a BlackAnt-en ez áll a rendelkezésünkre. A jövőben szerintem fontos lenne egy átláthatóbb és jobban strukturált adatbázist használni, például MySql-t vagy PostgreSQL-t, mivel sok felhalmozódott adat esetén könnyebbek lennének az utólagos munkák. Az implementáció és tesztelés során nem ütköztünk átláthatósági problémákba, mivel nem halmozódott fel annyi adat, de a jövőben fontos lesz lecserélni az adatbázist.

17.3. Modellek

AI modellek továbbfejlesztési lehetőségeinél a legfontosabb paraméter a fent említett nehézségek kiküszöbölése lenne, ez tartalmazza a szoftveres és hardveres megkötéseket. Erősebb és gyorsabb szerveren lényegesen nagyobb lenne a futási teljesítmény.

A modellek paraméterein is lehet még optimalizálni, előfordul, hogy sok használat után elkezd nem szokványos válaszokat generálni.

Kiemelkedő fejlesztés nyújtana a Llama3 8B verziója helyett egy nagyobb, több adaton tanított társára váltani. Ezzel a fejlesztéssel sokkal komplexebb és kifinomultabb válaszokat tudnánk generálni a felhasználók számára.

17.4. Funkciók

Projekt társaimmal nagyon sokat ötleteltünk, hogy milyen funkciókat valósítsunk meg a szakdolgozat munkánk alatt, mivel egyetemi éveink során többször éreztük hiányát egy

ennyire átfogó és sokoldalú platformnak. Jelenleg implementált funkcionalitásaink mellett számos nagyszerű gondolat született:

- *Videó alapján jegyzetkészítés*: Jelenleg kizárólag szöveges állományokkal dolgozunk, éppen ezért miért ne lehetne videókkal is? Az elképzelés hasonló, mint a szövegek generálásánál, a felhasználó elküldi a HelperBot számára a videóhoz tartozó url-t, amiből egy pontokra szedett összesítést vagy egy komplex, tanulható anyagot kap. A beszámoló vagy a tananyag legenerálása előtt beállítási paraméterek is eszközölhetőek, érintve a végeredmény hosszát és komplexitását, illetve mely részekre térjen ki részletesebben.
- *Kézzel írott szövegek*: Projektmunkánk során egy olyan alkalmazást fejlesztünk, ami felismeri olvashatatlan pdf fájlok szövegét, legyen ez kézzel vagy géppel írott formátumban. Képes elkülöníteni a matematikai részeket a szöveges részektől és latex formátumban visszaadni. Arra gondoltunk nem lenne nehéz majd a jövőben összeolvasztani a két munkánkat, mivel óriási hasznát vehetnénk. Tananyagok vagy tanuló kártyák generálásánál saját füzetünk beszkennelésével kaphatnánk meg az eredményt, vagy akár egy új kibővített jegyzettel gazdagodhatnánk. Jelentősége nem merül ki ennyiben, mivel a szövegalkotó és javító funkciókhoz is lehetne használni. A beszkennelt oldalak feltöltése után az AI kiegészíti, kijavítja vagy átfogalmazza a tanórák alatt sietve leírt jegyzetünket.

17.5. Platformok

A jövőben nem szeretném, hogy a felhasználók számítógéphez legyenek kötve, ha használni szeretnék az AI tanuló platformot. Minden lehetőséget le szeretnék fedni, legyen szó desktop, android vagy ios alapú eszközökről. Ennek a fejlesztésnek nem kizárólag kényelmi szerepe lenne, hanem kibővítené a felhasználói bázist.

17.6. Tesztelés

Jelen pillanatban a tesztelést kizárólag manuálisan tudtam megoldani, nem tudok pontossági méréseket végrehajtani a modellemen, mivel dinamikusan változnak a kimeneti értékek éppen ezért előre soha nem lehet megmondani mi lesz a válasz, maximum következtetni lehet a bemenetként megkapott utasítások alapján.

A jövőben sokat segítene a tesztelés felgyorsításában, vagy akár kiváltásában egy automatizált tesztelés. Úgy gondolom erre tökéletes megoldást nyújtana a Selenium, ami a manuális tesztelést végezné el automatizálva. Képes lenne előre definiált bemeneti értékek alapján kipróbálni minden egyes funkciót és a legenerált válaszokat elmenteni, az előforduló hibákat pedig logolni.

18.TARTALMI ÖSSZEFOGLALÓ

18.1. Magyar nyelven

A szakdolgozati dokumentációm egy betekintést nyújtott az AI-meghajtott tanulást segítő platform általam fejlesztett részeinek fejlesztési menetébe és működésébe, egy átfogó képet adva a platform fontosságáról és a benne rejlő potenciálról. Az olvasó megismerhette a felhasználói felületet, az email író, a szöveg kiegészítő, a szöveg javító és átfogalmazó funkciók működését felhasználói és szerveroldalról egyaránt. Bemutatásra kerültek a projekt alatt használt keretrendszerek, modellek és minden az implementálás alatt használt fejlesztési eszköz. Tesztesetekkel alá lett támasztva minden funkció helyes működése, ezzel egyaránt bemutatva a szöveggenerálások lépéseinek menetét.

Természetesen, mint minden projekt esetében találkoztunk mi is nehézségekkel és hátráltató tényezőkkel, de sikerült mindenre megoldást találni, akár átmenetit vagy véglegest.

Nem győzöm hangsúlyozni a weboldalban rejlő potenciált, amit a továbbfejlesztési lehetőségeknél ki is fejtettem. Ez számunkra nem csak egy szakmai dolgozat volt, hanem egy olyan tanulási és szakmai fejlődés lehetősége, ami elengedhetetlen a mai világban. A projekt fejlesztése nem áll meg a bemutató után, minden jelenlegi hibáját és hiányosságát javítani fogjuk és mindenképp tovább szeretnénk lendíteni a szakdolgozati státuszból.

Véleményem szerint teljes mértékben sikerült az elvártaknak megfelelni és minden kitűzött célt teljesíteni.

18.2. Angol nyelven

My thesis documentation provided an insight into the development process and operation of the parts of the AI-powered learning platform I developed, giving an overview of the platform's importance and potential. The reader was able to learn about the user interface, the functionality of the email writer, the text completion, the text correction and rephrasing functions from both the user and server side. The frameworks, models and all development tools used during the project implementation were presented. Test cases have been used to demonstrate the correct functioning of each function, showing the steps of the text generation process.

Of course, as with all projects, we have encountered difficulties and obstacles, but we have managed to find solutions to all of them, whether temporary or permanent.

I cannot stress enough the potential of the website, which I have explained in the section on further development. For us, this was not just a thesis, but an opportunity for learning and professional development, which is essential in today's world. The development of

the project will not stop after the presentation, we will improve all its current flaws and shortcomings and we definitely want to boost it further from the thesis status.

In my opinion, we have fully met the expectations and achieved all the objectives set.

19.IRODALOMJEGYZÉK

[1] „Introduction” - https://v2.vuejs.org/v2/guide/, Utoljára megtekintve: 2024.04.08

[2] „The Progressive Framework” - https://vuejs.org/guide/introduction.html, Utoljára megtekintve: 2024.04.08

[3] „Why Quasar?” - https://quasar.dev/introduction-to-quasar, Utoljára megtekintve: 2024.04.16

[4] „Llama models and tools” - https://llama.meta.com/, Utoljára megtekintve: 2024.04.22

[5] „Enhanced performance” - https://llama.meta.com/llama3, Utoljára megtekintve: 2024.04.22

[6] „Model Card” - https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md, Utoljára megtekintve: 2024.04.22

[7] „Getting Started” - https://axios-http.com/docs/intro, Utoljára megtekintve: 2024.05.13

[8] „Introduction” - https://pinia.vuejs.org/introduction.html, Utoljára megtekintve: 2024.05.15

[9] „Chat Message” - https://quasar.dev/vue-components/chat, Utoljára megtekintve: 2024.05.15

[10] „What is MongoDB?” - https://www.mongodb.com/docs/manual/, Utoljára megtekintve: 2024.11.25

[11] „MongoDB PyMongo Documentation” - https://www.mongodb.com/docs/languages/python/pymongo-driver/current/, Utoljára megtekintve: 2024.11.25

[12] „MongoDB with Python” - https://www.mongodb.com/docs/languages/python/, Utoljára megtekintve: 2024.11.25

[13] „What is Python? Executive Summary” - https://www.python.org/doc/essays/blurb/, Utoljára megtekintve: 2024.11.30

[14] „Flask” - https://github.com/pallets/flask/, Utoljára megtekintve: 2024.11.30

[15] Balázs, M., Eigner, G. (2024). BlackAnt—High Available Micro-Service Based Cloud Platform with Dynamical Scaling. In: Kovács, L., Haidegger, T., Szakál, A. (eds) Recent Advances in Intelligent Engineering. Topics in Intelligent Engineering and Informatics, vol 18. Springer, Cham. https://doi.org/10.1007/978-3-031-58257-8_12, Utoljára megtekintve: 2024.12.10

[16] „Our Mission” - https://www.getcoralai.com/about-us?gc_id=21579256258&h_ga_id=166062671163&h_ad_id=710692591647&h_keyword_id=kwd-328172635649&h_keyword=study%20ai&h_placement=&gad_source=1&gclid=CjwKCAiAjeW6BhBAEiwAdKltMvEBRTbuXAKW6pYCwOF-

n4Z4gPpe4KDBYqJy51Zswv-ep781plIgIRoCZTsQAvD_BwE&referrer=https%3A%2F%2Fwww.google.com%2F, Utoljára megtekintve: 2024.12.11

[17] „Features” - https://www.getcoralai.com/features?gc_id=21579256258&h_ga_id=166062671163&h_ad_id=710692591647&h_keyword_id=kwd-328172635649&h_keyword=study%20ai&h_placement=&gad_source=1&gclid=CjwKCAiAjeW6BhBAEiwAdKltMvEBRTbuXAKW6pYCwOF-n4Z4gPpe4KDBYqJy51Zswv-ep781plIgIRoCZTsQAvD_BwE&referrer=https%3A%2F%2Fwww.google.com%2F, Utoljára megtekintve: 2024.12.11

[18] „Does Coral AI work in other languages?” - https://www.getcoralai.com/?gc_id=21579256258&h_ga_id=166062671163&h_ad_id=710692591647&h_keyword_id=kwd-328172635649&h_keyword=study%20ai&h_placement=&gad_source=1&gclid=CjwKCAiAjeW6BhBAEiwAdKltMvEBRTbuXAKW6pYCwOF-n4Z4gPpe4KDBYqJy51Zswv-ep781plIgIRoCZTsQAvD_BwE, Utoljára megtekintve: 2024.12.11

20.ÁBRAJEGYZÉK