Implementacija RAG modela u web stranicu

Matan, Lovro Luka

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

Lovro Luka Matan

IMPLEMENTACIJA RAG MODELA U WEB STRANICU

Završni rad

Pula, svibanj 2024

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

Lovro Luka Matan

IMPLEMENTACIJA RAG MODELA U WEB STRANICU

Završni rad

JMBAG: 0114035988, redovni student

Studijski smjer: Informatika

Kolegij: Web aplikacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Nikola Tanković

Pula, svibanj 2024

1. Uvod

Umjetna inteligencija se posljednjih godina razvila u jednu od najvažnijih tehnologija koja je promijenila način na koji živimo i radimo. Zahvaljujući napretku u računalnoj snazi, algoritmima dubokog učenja i velikim količinama podataka, AI sustavi sada mogu izvršavati zadatke koji su nekada bili rezervirani isključivo za ljude. Oni prepoznaju obrasce, interpretiraju jezik, pretražuju velike količine podataka i donose odluke koje su prilagođene specifičnim uvjetima. AI-asistent omogućuje korisnicima učitavanje i analiziranje dokumenata kao što su PDF-ovi i CSV datoteke. Aplikacija koristi AI sustav temeljen na tehnologijama Flask-a za backend, MongoDB-a za pohranu podataka, i LlamaIndex-a za procesiranje upita korisnika. Kako bi se pristupilo AI agentu koristi se frontend napravljen u Vue.js-u. Cilj je bio napraviti AI agent koji omogućuje korisnicima da iskoriste umjetnu inteligenciju za analizu i pretragu informacija u dokumentima. AI koristi retrieval-augmented generation (RAG) i druge tehnike kako bi obogatio korisničko iskustvo pružanjem relevantnih i točnih odgovora

2. Korištene tehnologije

U ovom projektu koristi se niz naprednih tehnologija i alata koji omogućuju složene funkcionalnosti, uključujući učitavanje dokumenata, pretragu podataka i generiranje odgovora pomoću umjetne inteligencije. Kombinacija različitih tehnologija osigurava učinkovitost, sigurnost i jednostavnost interakcije s korisnikom.

Glavne tehnologije koje ću detaljno objasniti uključuju retrieval-augmented generation (RAG), koja omogućava pretraživanje i generaciju informacija na temelju stvarnih podataka, te druge alate poput LlamaIndex-a za pretragu dokumenata, OpenAI API-ja za generiranje prirodnog jezika i različite Python biblioteke koje omogućuju obradu datoteka i komunikaciju između korisničkog i poslužiteljskog sloja.

Sljedeći odjeljci pružit će detaljna objašnjenja svake tehnologije, njezine uloge u projektu i kako je implementirana.

2.1 Retrieval-augmented generation (RAG)[1]

Retrieval-Augmented Generation (RAG) je metoda koja kombinira dvije tehnologije pretraživanje informacija (retrieval) i generiranje teksta (generation) koristeći modele za prirodni jezik poput ChatGPT-a [3]. Umjesto da se oslanja isključivo na model velikih jezičnih podataka (kao što je GPT) za odgovaranje na upite, RAG pristup omogućuje sustavu da prvo pretražuje relevantne podatke iz vanjskih izvora [2] (kao što su baze podataka ili dokumenti) i zatim koristi te informacije za generiranje relevantnog odgovora.

Funkcionalnost

Pretraživanje podataka (Retrieval phase):

[1] Retrieval-Augmented Generation (RAG) Koncept: https://blogs.nvidia.com/blog/2020/09/10/retrieval-augmented-generation-bert-gpt-3/

- Kada korisnik postavi upit, sustav prvo pretražuje dostupne podatke (npr. PDF-ove, CSV datoteke, baze podataka) koristeći algoritme za pretragu ili indekse (npr. vektorski indeksi poput LlamaIndex).
- Ovaj korak obično koristi vektorske pretrage ili tradicionalne pretraživačke alate (npr. Elasticsearch) kako bi pronašao relevantne dokumente ili podatke koji odgovaraju upitu korisnika.

Generiranje odgovora (Generation phase):

- Nakon što su relevantni podaci pronađeni, ti podaci se predaju jezičnom modelu (npr. ChatGPT) kako bi generirao koherentan i informativan odgovor na temelju pretraženih informacija.
- Ova kombinacija omogućava modelu da generira odgovore koji su ne samo kontekstualno točni već i temeljeni na stvarnim podacima.

Primjena

RAG pristup koristi se za kombiniranje pretrage po PDF-ovima i CSV datotekama s generiranjem odgovora pomoću AI modela. To omogućava korisnicima da postave upit o sadržaju učitanih dokumenata, a sustav će prvo pretražiti sadržaj dokumenata i zatim generirati odgovor.

Primjer

PDF-ovi i CSV datoteke se prvo pretražuju kako bi se pronašli relevantni podaci na temelju korisničkog upita.
AI model (npr. ChatGPT) zatim koristi te podatke za generiranje odgovora.

2.2 LlamaIndex [2]

LlamaIndex je alat koji omogućuje integraciju velikih jezičnih modela (LLMs) s vanjskim bazama podataka, dokumentima i strukturiranim podacima. Svrha ovog alata je pomoći korisnicima da pretražuju i dobiju odgovore na pitanja iz tih vanjskih izvora podataka.

Funkcionalnost

Kreiranje vektorskih indeksa (Indexing phase):

- Dokumenti (npr. PDF-ovi, CSV-ovi) se prvo obrađuju i pretvaraju u vektore koji predstavljaju informacije unutar tih dokumenata.
- Ovi vektori su temelj pretrage, omogućujući modelu da pronađe relevantne dijelove teksta na temelju sličnosti između korisničkog upita i vektora dokumenata.

Upit i pretraga (Querying phase):

- Kada korisnik postavi upit, LlamaIndex pretražuje kreirane vektorske indekse kako bi pronašao dijelove teksta koji su relevantni za korisnikov upit.
- Relevantni dijelovi dokumenata se vraćaju i šalju AI modelu (npr. GPT-3.5-turbo) kako bi generirao odgovor na temelju tih informacija.

Primjena

LlamaIndex se koristi za pretraživanje podataka u PDF i CSV datotekama. Nakon što su dokumenti učitani, oni se pretvaraju u vektore koji omogućuju brzo pretraživanje relevantnih informacija. Korisnik može postaviti pitanje vezano uz sadržaj dokumenata, a LlamaIndex pretražuje te dokumente i vraća najrelevantnije informacije koje se zatim koriste za generiranje odgovora.

Primjer koda za pretragu PDF-ova pomoću LlamaIndex-a:

[2] LlamaIndex : https://gpt-index.readthedocs.io/en/latest/

```python
def load_pdfs_from_gridfs(pdf_ids):
    engines = {}
    for pdf_id in pdf_ids:
        result = pdf_collection.find_one({"_id": pdf_id})
        if result is None:
            logging.error(f"No document found with _id {pdf_id}")
            continue

        file_name = result['filename']
        pdf_name = os.path.splitext(file_name)[0]

        # Check if the file exists in GridFS
        if not fs.exists(pdf_id):
            logging.error(f"File with _id {pdf_id} does not exist in GridFS")
            continue

        try:
            pdf_bytes = fs.get(pdf_id).read()
            pdf_text = extract_text_from_pdf(pdf_bytes)
            index = get_index([pdf_text], pdf_name)
            engines[pdf_name] = index.as_query_engine()
        except Exception as e:
            logging.error(f"An error occurred while processing PDF with _id {pdf_id}: {str(e)}")
    return engines
```

Slika1: Primjer koda za pretragu PDF-ova pomoću LlamaIndex-a

Ovdje se PDF-ovi pretvaraju u indekse koje LlamaIndex koristi za pretragu i generiranje odgovora.

2.3 ChatGPT API[3]

ChatGPT API je sučelje koje omogućuje integraciju velikih jezičnih modela poput GPT-3.5 ili GPT-4 u aplikacije. API omogućava aplikacijama da šalju tekstualne upite modelu i dobivaju generirane odgovore. ChatGPT API omogućuje aplikaciji generiranje odgovora na temelju prirodnog jezika, pružajući korisnicima ljudski čitljive odgovore [4]. Ovaj pristup temelji se na Transformer arhitekturi koja je prvi put opisana u radu Vaswani i dr. [4].

[3] OpenAI API : https://platform.openai.com/docs/

Funkcionalnost

Slanje upita (Request):

- Aplikacija šalje POST zahtjev ChatGPT API-u s tekstualnim upitom. Ovaj upit može biti jednostavno pitanje ili složena naredba.

Obrada upita (Processing):

- Model analizira upit koristeći svoj model za obradu prirodnog jezika (Language Model) koji je treniran na velikim količinama tekstualnih podataka.

Generiranje odgovora (Response):

- Na temelju analize upita, API generira odgovor i vraća ga aplikaciji u tekstualnom formatu.

Primjena

ChatGPT API za generiranje odgovora na korisnička pitanja se koristi nakon što LlamaIndex pretraži relevantne podatke, pa ChatGPT koristi te podatke kako bi generirao odgovor koji odgovara na pitanje korisnika.

Primjer

```
llm = OpenAI(model="gpt-3.5-turbo")
agent = ReActAgent.from_tools(tools, llm=llm, verbose=True, context=context)

while (prompt := input("Enter a prompt (q to quit): ")) != "q":
    result = agent.query(prompt)
    print(result)
```

Slika2:Primjer ChatGPT API u projektu

2.4 Flask[4]

Flask je Python framework za razvoj web aplikacija. Jedna od njegovih glavnih prednosti je jednostavnost i fleksibilnost, što ga čini idealnim za izgradnju REST API-ja koje se koriste u aplikaciji.

[4] Flask :https://flask.palletsprojects.com/

Funkcionalnost

Definiranje ruta (Routing):

- Flask koristi `@app.route()` kako bi definirao rute ili "putove" unutar aplikacije. Ove rute povezuju URL-ove s funkcijama koje obrađuju zahtjeve.

Obrada zahtjeva (Request handling):

- Kada korisnik pošalje HTTP zahtjev (npr. POST, GET), Flask ga obrađuje i vraća odgovarajući odgovor. Zahtjev može sadržavati podatke u obliku JSON-a, oblika ili kao multimedijalni sadržaj poput PDF datoteka.

Slanje odgovora (Response):

- Nakon obrade, Flask vraća odgovor (obično JSON format) klijentu. To može biti status uspjeha, podaci iz baze ili poruka o grešci.

Primjena

Flask je korišten za definiranje API endpointova za prijavu, registraciju korisnika, učitavanje datoteka i postavljanje upita AI-u.

Primjer Flask rute za prijavu

```python
@app.route('/api/login', methods=['POST'])
def login():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')

    if not username or not password:
        return jsonify({'error': 'Missing fields'}), 400

    user = users_collection.find_one({'username': username})

    if user and bcrypt.check_password_hash(user['password'], password):
        access_token = create_access_token(identity={'username': username, 'email': user['email']},
                                           expires_delta=datetime.timedelta(hours=1))
        return jsonify({'access_token': access_token}), 200
    else:
        return jsonify({'error': 'Invalid credentials'}), 401
```

Slika3:Primjer Flask rute

2.5 PyPDF2[5]

PyPDF2 je Python knjižnica koja omogućuje rad s PDF datotekama. Omogućuje različite operacije nad PDF-ovima, uključujući ekstrakciju teksta, spajanje i razdvajanje stranica te manipulaciju metapodacima PDF datoteka [1].

PyPDF2 se koristi za ekstrakciju teksta iz PDF dokumenata koje korisnici učitavaju. Ova funkcionalnost je ključna za omogućavanje pretraživanja i analize sadržaja unutar PDF-ova putem AI asistenta.

Primjena

Kada korisnik učita PDF datoteku, backend koristi PyPDF2 za čitanje i ekstrakciju teksta iz svake stranice dokumenta

Ekstrahirani tekst zatim se koristi za kreiranje pretraživog indeksa pomoću LlamaIndex, čineći ga dostupnim za kasnije upite korisnika.

Prednosti

Jednostavna ekstrakcija teksta: Omogućuje jednostavan način za dohvaćanje i obradu teksta iz PDF-ova, bez obzira na njihovu veličinu.

Kompatibilnost: Podržava širok spektar PDF datoteka, što olakšava rad s različitim vrstama dokumenata koje korisnici mogu učitati.

2.6 MongoDB i GridFS[6]

MongoDB je NoSQL baza podataka koja pohranjuje podatke u obliku dokumenata sličnih JSON-u, što omogućuje fleksibilno i skalabilno rukovanje podacima [4]. Umjesto pohrane podataka u tradicionalnim tablicama (kao što je slučaj u relacijskim bazama podataka),

[5] PyPDF2 : https://pypdf2.readthedocs.io/en/latest/
[6] MongoDB i GridFS : https://www.mongodb.com/docs/drivers/gridfs/

MongoDB koristi kolekcije i dokumente, što je vrlo pogodno za pohranu nestrukturiranih ili polustrukturiranih podataka.

Primjena

MongoDB se koristi za pohranu korisničkih podataka (poput podataka za prijavu), kao i za pohranu i upravljanje velikim datotekama poput PDF-ova putem GridFS-a.

GridFS je sustav za pohranu datoteka unutar MongoDB-a koji omogućuje pohranu i dohvaćanje velikih datoteka (većih od ograničenja veličine BSON dokumenta). GridFS dijeli datoteke na manje dijelove i pohranjuje ih u kolekcije, što omogućuje učinkovito rukovanje čak i vrlo velikim datotekama.

Pohrana PDF-ova: Kada korisnik učita PDF datoteku putem korisničkog sučelja, backend koristi GridFS za pohranu datoteke u MongoDB. Ovo se radi pomoću funkcije `fs.put()` koja dijeli PDF na manje dijelove i pohranjuje ih u bazu podataka.

Ekstrakcija i pretraživanje podataka: Nakon pohrane, backend koristi PyPDF2 za ekstrakciju teksta iz PDF-a i LlamaIndex za stvaranje indeksa koji omogućuje pretraživanje sadržaja datoteke.

Brzi dohvat podataka: Kada korisnik postavi pitanje vezano za sadržaj PDF-a, aplikacija koristi GridFS za brzo dohvaćanje potrebnih dijelova datoteke i pretraživanje relevantnih informacija.

Prednosti

Fleksibilno pohranjivanje podataka: Omogućava jednostavnu pohranu nestrukturiranih podataka, uključujući dokumente i binarne podatke (npr., PDF-ove).

Upravljanje velikim datotekama: GridFS omogućuje pohranu velikih datoteka razdjeljujući ih na manje dijelove, što olakšava njihovo dohvaćanje i obradu [4].

Skalabilnost: MongoDB je dizajniran za skaliranje i može rukovati velikim količinama podataka, što je idealno za aplikacije koje pohranjuju i analiziraju velike datoteke.

3. Kod

Ovaj odjeljak pokriva implementaciju koda projekta i pruža pregled glavnih komponenti koje čine aplikaciju. Projekt je izgrađen koristeći korisnički sloj temeljen na Vue.js za interakciju s korisnicima, te poslužiteljski sloj implementiran pomoću Flask frameworka, koji upravlja autentifikacijom, pohranom podataka, analizom dokumenata i generiranjem odgovora putem AI-a.

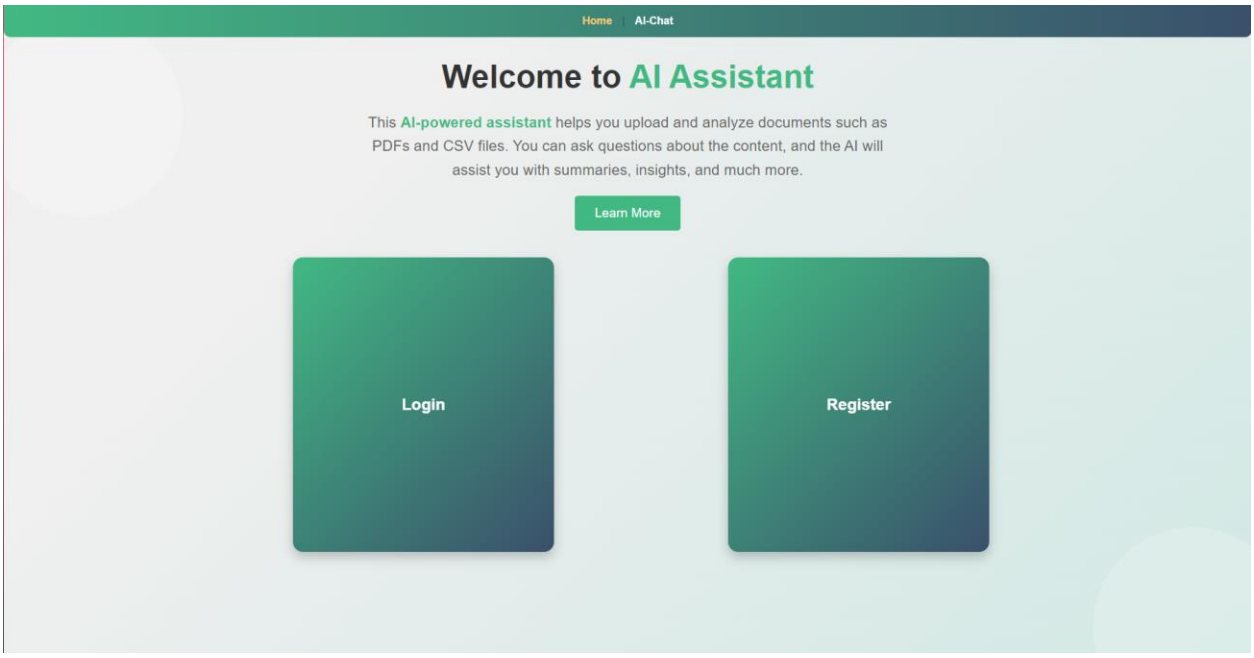
Kroz ovaj odjeljak, detaljno ću analizirati kako su implementirane ključne funkcionalnosti poput prijave, registracije, upravljanja datotekama i komunikacije s AI modelom. Započeti ćemo s korisničkim slojem aplikacije, gdje su komponente Vue.js-a odgovorne za interakciju s korisnikom, a zatim ćemo prijeći na backend i logiku koja stoji iza rukovanja korisničkim upitima i upravljanja dokumentima.

3.1 Korisnički sloj

Korisnički sloj predstavlja frontend aplikacije izgrađene pomoću Vue.js frameworka. Ovaj sloj osigurava interakciju s korisnicima kroz intuitivno korisničko sučelje. Sadrži komponente koje omogućuju prijavu i registraciju korisnika, postavljanje upita AI asistentu, kao i učitavanje i pretraživanje PDF datoteka. Korisnički sloj šalje zahtjeve poslužiteljskom sloju putem API poziva te prikazuje odgovore koje generira umjetna inteligencija, čineći aplikaciju lako razumljivom i jednostavnom za korištenje

Komponenta HomeView

Ova komponenta pruža korisničko sučelje za login i registraciju korisnika. Kartični dizajn omogućuje korisnicima da jednostavno prebacuju između kartica za prijavu i registraciju.



Slika4:Komponenta HomePageView

Vue.js omogućuje jednostavnu implementaciju dvosmjerne vezanosti podataka (two-way data binding), što znači da se promjene u unosnim poljima automatski ažuriraju u podacima pohranjenima u komponenti [2]. Na primjer, `v-model` direktiva koristi se za vezivanje vrijednosti polja za unos s odgovarajućim varijablama (`loginEmailUsername`, `loginPassword`, itd.) u skriptama.

Glavne funkcionalnosti

toggleFlip(): Funkcija koja prebacuje između login i register kartice.

login(): Prikuplja podatke o prijavi (korisničko ime i lozinku) i šalje POST zahtjev prema backendu za autentifikaciju.

register(): Omogućuje korisnicima registraciju unosom emaila, korisničkog imena, lozinke i potvrde lozinke. Provjerava da su lozinke iste prije slanja zahtjeva za registraciju na backend.

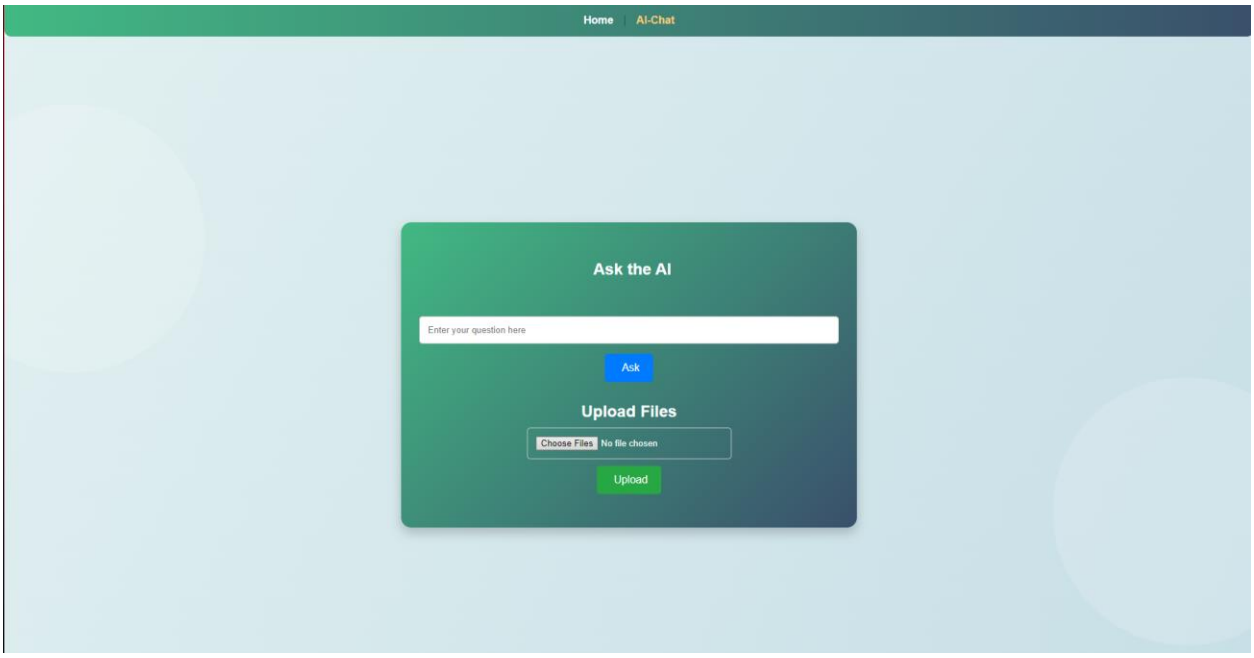
Kod za registraciju korisnika

```
async register() {
  if (this.registerPassword === this.registerPasswordConfirm) {
    const userData = {
      email: this.registerEmail,
      username: this.registerUsername,
      password: this.registerPassword,
    };

    try {
      const response = await fetch("http://localhost:5000/api/register", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(userData),
      });

      const result = await response.json();

      if (response.ok) {
        console.log("User registered successfully:", result);
        alert("User registered successfully!");
        this.$router.push("/chat");
      } else {
        console.error("Error registering user:", result.error);
        alert("Error registering user: " + result.error);
      }
    } catch (error) {
      console.error("Error during registration:", error);
      alert("An error occurred during registration. Please try again.");
    }
  } else {
    alert("Passwords do not match!");
  }
},
```

Slika5:Kod za registraciju korisnika

Komponenta AiChatView

Ova komponenta omogućuje korisnicima postavljanje pitanja AI asistentu i učitavanje PDF datoteka za analizu. Također sadrži unosno polje i gumb za slanje upita, kao i sekciju za prikaz odgovora koje generira AI.



Slika6:Komponenta AIChatView

Komponenta koristi `axios` biblioteku za slanje HTTP zahtjeva backendu. Funkcija `uploadFiles()` koristi JavaScript-ov `FormData` objekt za stvaranje multipart/form-data zahtjeva koji se šalje na `/api/upload_files` endpoint. Ova vrsta zahtjeva omogućuje prijenos binarnih podataka (datoteka) uz dodatne informacije, kao što su korisnički token za autentifikaciju.

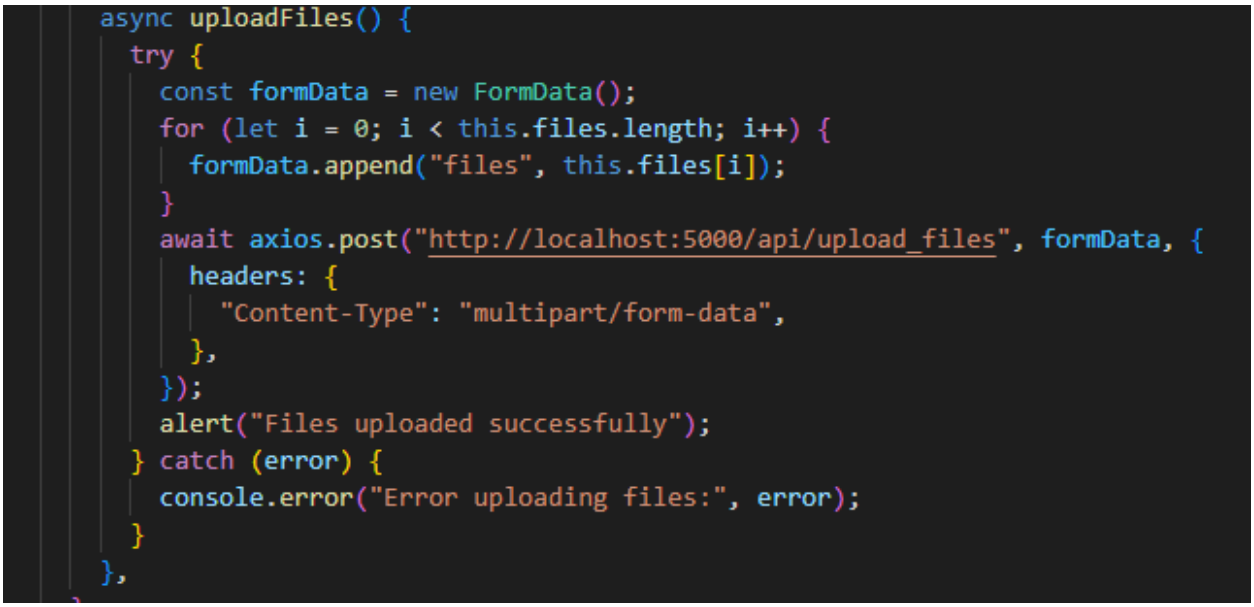
Glavne funkcionalnosti

askAi(): Funkcija koja šalje pitanje korisnika na backend, gdje AI obrađuje upit koristeći podatke iz PDF-ova ili drugih dokumenata.

onFileChange(): Prikuplja datoteke koje korisnik želi učitati i sprema ih u lokalnu varijablu.

uploadFiles(): Prikupljene datoteke se šalju na backend koristeći FormData objekt. Backend koristi GridFS za pohranu PDF-ova u MongoDB.

Kod za upload datoteka



```
async uploadFiles() {
  try {
    const formData = new FormData();
    for (let i = 0; i < this.files.length; i++) {
      formData.append("files", this.files[i]);
    }
    await axios.post("http://localhost:5000/api/upload_files", formData, {
      headers: {
        "Content-Type": "multipart/form-data",
      },
    });
    alert("Files uploaded successfully");
  } catch (error) {
    console.error("Error uploading files:", error);
  }
},
```

Slika7:Kod za unošenje datoteka

Objašnjenje

`onFileChange(event)`: Koristi se za praćenje promjene u unosu datoteka. Svaki put kada korisnik odabere novu datoteku, datoteke se pohranjuju u `this.files`.

`uploadFiles()`: Kreira novi `FormData` objekt i kroz petlju dodaje svaku odabranu datoteku. Potom šalje zahtjev backendu putem POST metode, uz zaglavlje "multipart/form-data", kako bi se datoteke pravilno prenijele.

3.2 Poslužiteljski sloj

Poslužiteljski sloj upravlja logikom aplikacije, od autentifikacije korisnika, do obrade upita korisnika putem AI-a. Također, ima ključnu funkcionalnost za učitavanje i pohranu PDF datoteka koristeći MongoDB GridFS. Poslužiteljski sloj koristi Flask framework kako bi osigurao REST API za komunikaciju između korisničkog sučelja i pozadinskog sustava [5]

Sigurnost aplikacije je osigurana pomoću JWT (JSON Web Token) tehnologije. Kada se korisnik uspješno prijavi, backend generira JWT token koristeći tajni ključ. Taj token se

vraća korisniku i pohranjuje na klijentskoj strani (npr., u localStorage). Svi daljnji zahtjevi prema API-ju uključuju ovaj token u zaglavlju zahtjeva za provjeru autentičnosti.

Primjer AI-agenta

```
PS C:\Users\lovro\Desktop\AIAGENT> python main.py
Enter a prompt (q to quit): What is the capitol of Slovenia?
> Running step a7776650-65f5-4c83-a136-5b7c737ef59f. Step input: What is the capitol of Slovenia?
Thought: The user is asking about the capital of Slovenia, so I can provide this information using a tool.
Action: Croatia_data
Action Input: {'input': 'capital of Slovenia'}
Observation: Ljubljana
> Running step 338ef72d-6eb9-4e2e-a417-f4593d247b43. Step input: None
Thought: I can answer without using any more tools. I'll use the user's language to answer
Answer: Ljubljana
Ljubljana
Enter a prompt (q to quit):
```

Slika8:Primjer interakcije s AI agentom

Učitavanje CSV datoteka

Koristi se `PandasQueryEngine` za pretraživanje podataka iz CSV datoteka. Ove datoteke su definirane na početku, a query engine se automatski stvaraju za svaku od njih.

```python
def load_csvs(csv_paths):
    engines = {}
    for csv_path in csv_paths:
        csv_name = os.path.basename(csv_path).split('.')[0]
        df = pd.read_csv(csv_path)
        query_engine = PandasQueryEngine(df=df, verbose=True, instruction_str=instruction_str)
        query_engine.update_prompts({"pandas_prompt": new_prompt})
        engines[csv_name] = query_engine
    return engines
```

Slika9:Kod za učitavanje CSV datoteka

Učitavanje PDF datoteka

Koristi se `PDFReader` za čitanje PDF-ova, a `VectorStoreIndex` za stvaranje indeksa koji omogućuje pretragu tih datoteka.

```python
def load_pdfs(pdf_paths):
    engines = {}
    for pdf_path in pdf_paths:
        pdf_name = os.path.basename(pdf_path).split('.')[0]
        pdf_data = PDFReader().load_data(file=pdf_path)
        engines[pdf_name] = get_index(pdf_data, pdf_name)
    return engines
```

Slika10:Kod za unošenje PDF datoteka

Ovdje se učitavaju sve PDF datoteke i dodjeljuju im se engine za pretragu.

Server.py

Ovaj file upravlja glavnim API endpointovima za autentifikaciju, postavljanje upita i učitavanje datoteka.

Učitavanje datoteka

Endpoint `/api/upload_files` omogućava korisnicima učitavanje PDF datoteka. Svaka datoteka se pohranjuje u MongoDB GridFS, a metapodaci se spremaju u kolekciju `PDFs`.

```python
@app.route('/api/upload_files', methods=['POST'])
def upload_files():
    if 'files' not in request.files:
        return jsonify({'error': 'No files part in the request'}), 400

    files = request.files.getlist('files')
    for file in files:
        if file:
            file_id = fs.put(file, filename=file.filename)
            # Store file metadata in MongoDB
            file_metadata = {
                "filename": file.filename,
                "file_id": file_id,
                "type": "pdf"
            }
            pdf_collection.insert_one(file_metadata)

            # Update engines with the new file content
            pdf_engines.update(load_pdfs_from_gridfs([file_id]))

    return jsonify({'message': 'Files uploaded successfully'}), 200
```

Slika11:Ruta za unošenje PDF datoteka

Objašnjenje

Provjera postoji li `files` u zahtjevu. Ako ne postoji, vraća se pogreška.
Ako datoteka postoji, ona se pohranjuje u GridFS pomoću `fs.put()`. Također, metapodaci poput imena datoteke i tipa pohranjuju se u kolekciji `PDFs`.
Nakon pohrane, novi engine se kreira za svaku učitanu PDF datoteku, omogućujući korisnicima da postavljaju upite na temelju sadržaja tog PDF-a.

Postavljanje upita AI-u

Endpoint `/api/ask_ai` omogućuje korisnicima da postave pitanje vezano uz sadržaj PDF-a ili CSV-a. AI asistent zatim obrađuje pitanje i vraća odgovor.

```python
@app.route('/api/ask_ai', methods=['POST'])
def ask_ai():
    data = request.get_json()
    question = data.get('question')
    if not question:
        return jsonify({'error': 'Question is required'}), 400
    try:
        result = agent.query(question)
        return jsonify({'answer': str(result)})  # Ensure result is converted to string
    except Exception as e:
        logging.error(f"Error processing the request: {str(e)}", exc_info=True)
        return jsonify({'error': str(e)}), 500
```

Slika12:Ruta za postavljanje pitanja AI

Objašnjenje

Funkcija prima pitanje korisnika u JSON formatu i koristi `agent.query()` za slanje tog pitanja AI agentu.
Ako postoji bilo kakva greška tijekom obrade pitanja, vraća se odgovarajuća poruka o pogrešci.

Učitavanje PDF-ova iz GridFS-a

Funkcija za učitavanje PDF-ova iz MongoDB GridFS i generiranje query engine-ova za njih.

```python
def load_pdfs_from_gridfs(pdf_ids):
    engines = {}
    for pdf_id in pdf_ids:
        result = pdf_collection.find_one({"_id": pdf_id})
        if result is None:
            logging.error(f"No document found with _id {pdf_id}")
            continue

        file_name = result['filename']
        pdf_name = os.path.splitext(file_name)[0]

        # Check if the file exists in GridFS
        if not fs.exists(pdf_id):
            logging.error(f"File with _id {pdf_id} does not exist in GridFS")
            continue

        try:
            pdf_bytes = fs.get(pdf_id).read()
            pdf_text = extract_text_from_pdf(pdf_bytes)
            index = get_index([pdf_text], pdf_name)
            engines[pdf_name] = index.as_query_engine()
        except Exception as e:
            logging.error(f"An error occurred while processing PDF with _id {pdf_id}: {str(e)}")
    return engines
```

Slika13:Kod za učitavanje PDF datoteka s baze podataka

Objašnjenje

Funkcija koristi `find_one()` za dohvaćanje metapodataka PDF datoteke pohranjene u MongoDB. Ako datoteka postoji, njezin sadržaj se učitava pomoću `fs.get().read()`. Nakon toga, koristi se funkcija `extract_text_from_pdf()` koja koristi PyPDF2 za ekstrakciju teksta iz PDF-a.
Konačno, generira se query engine koji omogućuje pretraživanje tog PDF-a.

4. Funkcionalnost

Ovaj odjeljak opisuje glavne funkcionalnosti aplikacije, uključujući načine na koje korisnici mogu komunicirati s njom. Aplikacija kombinira mogućnosti korisničke autentifikacije, obrade i pohrane dokumenata te interakcije s AI asistentom. Kroz jednostavno korisničko sučelje, korisnicima je omogućeno učitavanje PDF datoteka, postavljanje upita vezanih uz sadržaj tih dokumenata te primanje odgovora generiranih uz pomoć umjetne inteligencije.

U nastavku su opisani ključni koraci i instrukcije kako aplikacija radi i kako korisnici mogu koristiti sve njezine funkcionalnosti.

4.1 Instrukcije

Prijava i Registracija

- Korisnik pokreće aplikaciju i ima opciju prijaviti se ili registrirati. Nakon uspješne prijave, korisnik dobiva JWT token, što mu omogućava pristup ostalim funkcijama unutar aplikacije.

Učitavanje PDF datoteka

- Nakon prijave, korisnik može učitati PDF datoteke kroz sučelje u komponenti AiChatView. Komponenta koristi funkciju `uploadFiles()` za prijenos datoteka na backend. Backend koristi GridFS za pohranu datoteka, a zatim obrađuje njihov sadržaj pomoću PyPDF2 i LlamaIndex-a, omogućavajući njihovo kasnije pretraživanje.

Postavljanje upita

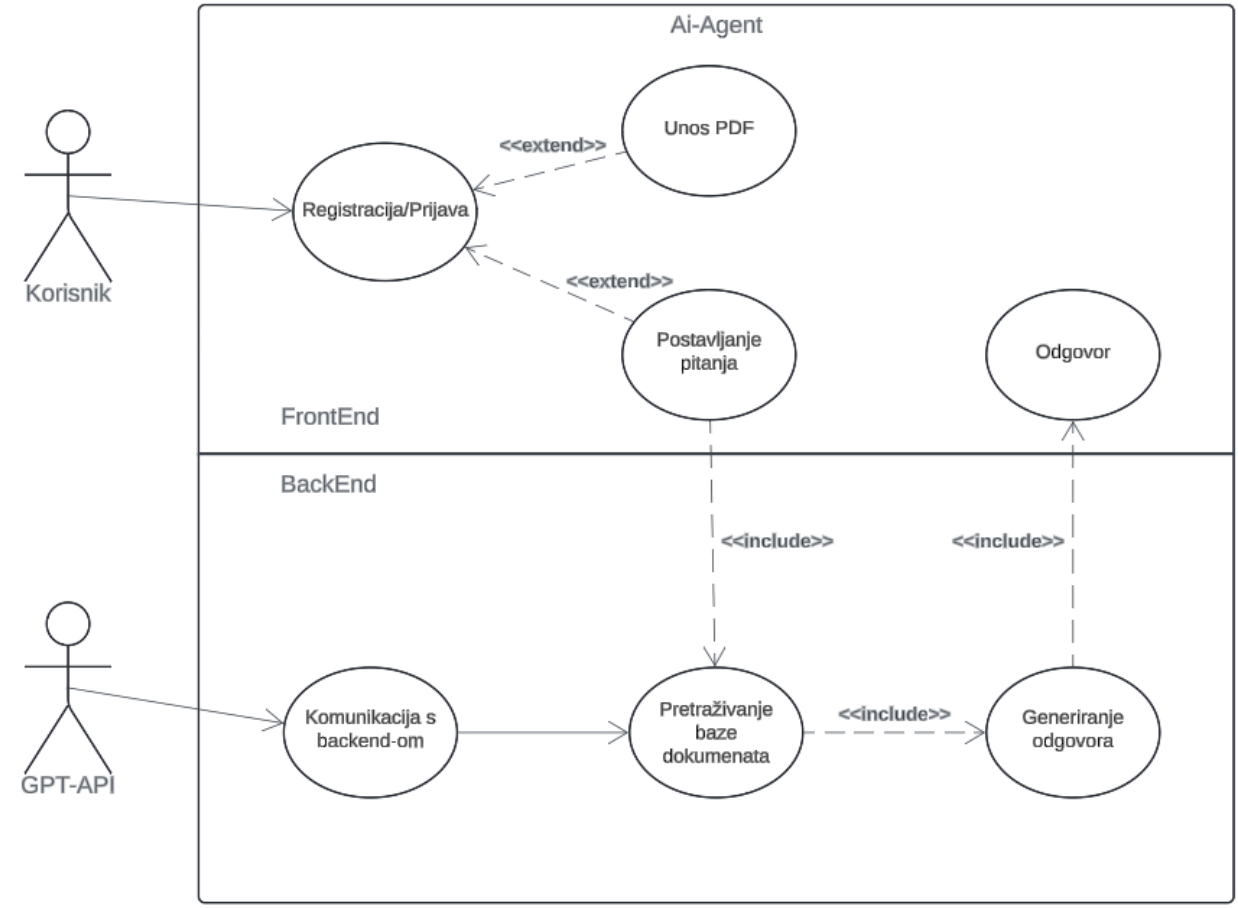
- Korisnik postavlja pitanja vezana uz učitane PDF ili CSV dokumente. Kada korisnik postavi upit, aplikacija koristi Retrieval-Augmented Generation (RAG) pristup za dohvaćanje relevantnih podataka iz datoteka. To omogućuje AI modelu da koristi informacije iz dokumenata za generiranje odgovora, pružajući tako korisnicima točne i ažurirane informacije.

- Ako se pojavi pogreška prilikom učitavanja datoteka ili obrade upita (npr., neuspješan dohvat iz MongoDB-a ili pogrešan unos korisnika), aplikacija vraća odgovarajuće poruke o grešci koje korisniku pomažu u razumijevanju problema i njegovom rješavanju.

Prikaz odgovora

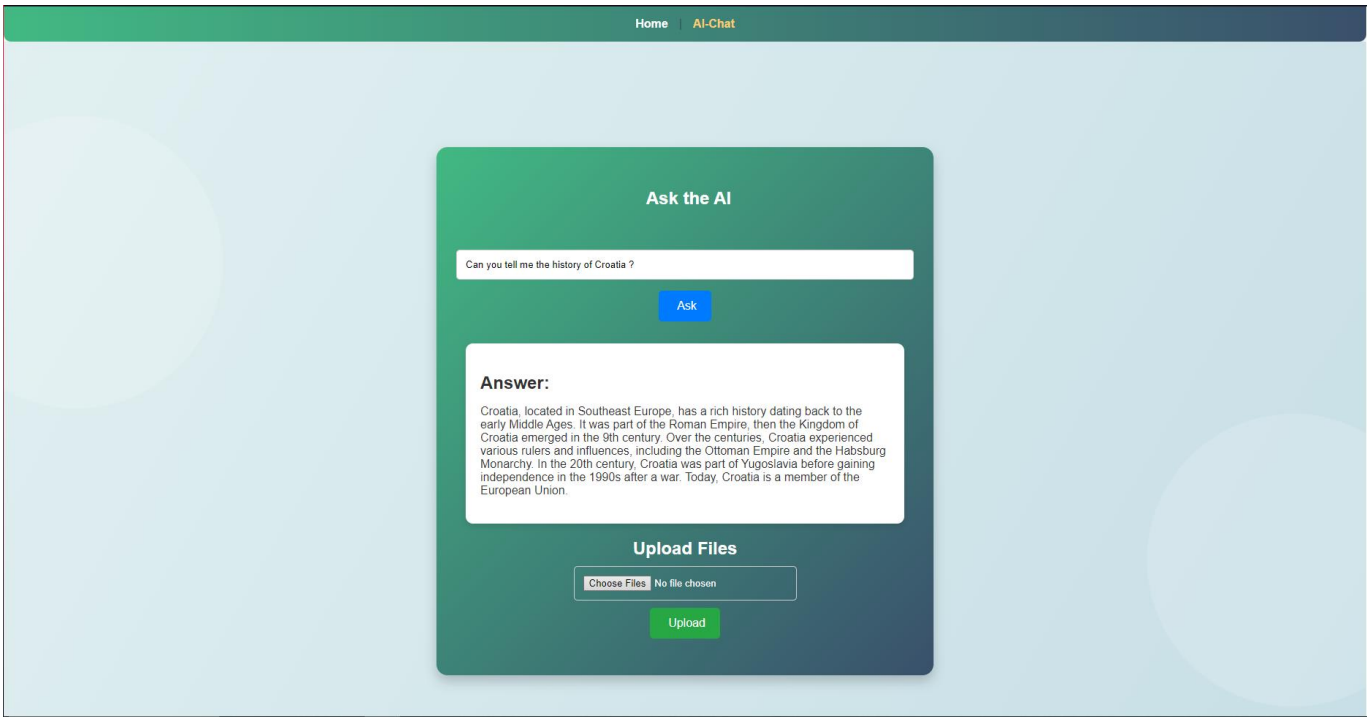
- Kada se pronađe odgovor, koristi se ChatGPT API za generiranje ljudski čitljivog odgovora na temelju lokalnih i vanjskih podataka. Odgovor se zatim vraća korisniku i prikazuje u prozoru za razgovor.

4.1.1 Dijagram



Slika14:Dijagram

4.1.2 Prikaz aplikacije



Slika15:Prikaz aplikacije

5. Zaključak

Tehnologije kao što su RAG, LlamaIndex, PandasQueryEngine i ChatGPT API igraju ključnu ulogu u projektu, omogućujući složenu analizu podataka i generiraju odgovore na temelju vanjskim izvorima. RAG omogućava kombinaciju dohvaćenih podataka s AI modelom za generiranje kontekstualno točnih odgovora, dok LlamaIndex i PandasQueryEngine omogućuju efikasno pretraživanje i analizu podataka iz dokumenata. Sve te komponente zajedno čine AI-asistent fleksibilnim alatom za rad s različitim vrstama podataka. Budući rad na aplikaciji može uključivati proširenje podrške za dodatne vrste datoteka, optimizaciju performansi pretrage te unapređenje korisničkog sučelja za bolju interakciju. Implementacija naprednijih sigurnosnih protokola za prijenos osjetljivih podataka i dodavanje funkcija za naprednu analizu sadržaja dokumenata također su potencijalna područja za razvoj.

Sažetak na hrvatskom:

Ova aplikacija omogućuje korisnicima učitavanje PDF datoteka, pretraživanje njihovog sadržaja te postavljanje upita AI asistentu koji koristi Retrieval-Augmented Generation (RAG) za obogaćivanje odgovora dohvaćenim podacima. Time se korisnicima pružaju točni i relevantni odgovori na temelju stvarnih podataka.

Ključne riječi: AI,Vue.js,MongoDB,Flask,Python,RAG

Summary in English:

This application allows users to upload PDF files, search their content, and submit questions to an AI assistant that uses Retrieval-Augmented Generation (RAG) to enhance responses with relevant data. This provides users with accurate and contextually relevant answers based on real-world information.

Keywords: AI,Vue.js,MongoDB,Flask,Python,RAG

Literatura

1. S. Bird, E. Klein i E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, 1. izd., Sebastopol, SAD: O'Reilly Media, 2009.
2. R. Baeza-Yates i B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology Behind Search*, 2. izd., Boston, SAD: Addison-Wesley, 2011.
3. P. Lewis, E. Perez, A. Piktus i dr., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv preprint arXiv:2005.11401*, svibanj 2020. [Online]. Dostupno: https://arxiv.org/abs/2005.11401. [Pristupljeno: 8. kolovoza2024.].
4. A. Vaswani, N. Shazeer, N. Parmar i dr., "Attention Is All You Need," *arXiv preprint arXiv:1706.03762*, lipanj 2017. [Online]. Dostupno: https://arxiv.org/abs/1706.03762. [Pristupljeno: 30. srpnja 2024.].
5. A. Paszke, S. Gross, F. Massa i dr., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv preprint arXiv:1912.01703*, prosinac 2019. [Online]. Dostupno: https://arxiv.org/abs/1912.01703. [Pristupljeno: 2. rujna 2024.].

Popis slika