# Razvoj i implementacija AI agenta temeljenog na velikim jezičnim modelima

**Radulović, Bojan**

**Master's thesis / Diplomski rad (sveučilišni)**

**2025**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Rijeka, Faculty of Informatics and Digital Technologies / Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:195:516338

*Rights / Prava:* Zaštićeno autorskim pravom. / In copyright.

*Download date / Datum preuzimanja:* **2026-11-10**

*Repository / Repozitorij:*

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies / Repozitorij Fakulteta informatike i digitalnih tehnologija Sveučilišta u Rijeci

Dabar

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni diplomski studij Informatika

Bojan Radulović

# Razvoj i implementacija AI agenta temeljenog na velikim jezičnim modelima

Diplomski rad

Mentor: prof. dr. sc. Ana Meštrović

Rijeka, 9. srpnja 2025.

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija
uniri

Rijeka, 2.6.2025.

# Zadatak za diplomski rad

Pristupnik/ica: Bojan Radulović

Naziv diplomskog rada: Razvoj i implementacija AI agenta temeljenog na velikim jezičnim modelima

Naziv diplomskog rada na eng. jeziku: Development and Implementation of an AI Agent Based on Large Language Models

Sadržaj zadatka:
Veliki jezični modeli omogućuju razvoj naprednih AI agenata koji korisnicima mogu asistirati u obavljanju digitalnih zadataka. U ovom diplomskom radu potrebno je razviti aplikaciju koja implementira agente sposobne za razne zadatke kao što su npr. slanje e-pošte, generiranje slika, izvođenje Python kalkulacija i pretraživanje weba. U radu je potrebno opisati teorijsku pozadinu, a potom korištene tehnologije, arhitekturu i funkcionalnosti razvijenog agenta i sustava.

Mentor/ica:
Prof. dr. sc. Ana Meštrović

Voditeljica za diplomske radove:
Izv. prof. dr. sc. Lucia Načinović Prskalo

Komentor/ica:

Zadatak preuzet (datum): 3.6.2025.

(potpis pristupnika/ce)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR5324020061100977786

## Sažetak

Cilj ovog diplomskog rada implementacija je i razvoj agenta temeljenog na velikim jezičnim modelima (engl. *large language models*, LLM) i umjetnoj inteligenciji (engl. *artificial intelligence*, AI) čija je funkcija asistiranje korisnika prilikom obavljanja raznih digitalnih zadataka. Funkcionalnosti modela uključuju slanje elektroničke pošte, generiranje slika, izvršavanje kalkulacija koristeći Python programski kôd i pretraživanje web-a.

U ovu svrhu, izrađen je sustav temeljen na AI agentima naziva Lumen koji se sastoji od korisničkog sučelja izrađenog primjenom Reacta te poslužiteljskog dijela implantiranog pomoću programskog jezika Python koji koristi tehnologije poput LangChain-a, FastAPI-a, RabbitMQ-a itd. te mikroservisnu arhitekturu. Sustav sadrži i funkcionalnosti prepoznavanja govora (engl. *speech to text*, STT) te generiranja glasovnih odgovora na korisnička pitanja (engl. *text to speech*, TTS).

Prilikom izrade ovog sustava, nastojalo se gdje god je moguće koristiti lokalno pokrenute modele i programska rješenja otvorenog koda. Korišten je LLM otvorenog koda Solar 10.7B Instruct v1.0, Kokoro TTS model, SearXNG za pretraživanje web-a, SDXL-Lightning model za generiranje slika itd.

**Ključne riječi:** umjetna inteligencija; agent; veliki jezični modeli; React; otvoreni kod; LangChain; Python; Apache; FastAPI; RabbitMQ

# SADRŽAJ

1. Uvod ..... 1
2. Teorijska podloga ..... 2
2.1. Generativna umjetna inteligencija ..... 2
2.1.1. Upiti i inženjering upita ..... 3
2.2. Veliki jezični modeli ..... 5
2.2.1. Nedostatci velikih jezičnih modela ..... 7
2.3. AI agenti ..... 9
2.3.1. Arhitektura AI agenata ..... 10
3. Razvoj i implementacija sustava Lumen ..... 13
3.1. Razvoj poslužiteljske strane sustava ..... 13
3.1.1. Pregled arhitekture sustava ..... 13
3.1.2. Baza podataka ..... 14
3.1.3. FastAPI poslužitelj ..... 16
3.1.4. AI Agent i veliki jezični model ..... 18
3.1.5. Alati ..... 19
3.1.6. TTS i STT funkcionalnosti ..... 26
3.1.7. RabbitMQ ..... 26
3.2. Implementacija korisničkog sučelja ..... 27
3.2.1. Sučelje za prijavu ..... 27
3.2.2. Oporavak lozinke ..... 28
3.2.3. Administratorsko sučelje ..... 32
3.2.4. Sučelje za razgovor s agentom ..... 33
3.3. Objavljivanje sustava ..... 39
3.4. Mogućnost unaprjeđenja sustava ..... 40
3.5. Specifikacije sustava ..... 41
4. Zaključak ..... 42
Literatura ..... 43
Popis slika ..... 45
Popis priloga ..... 46

# 1. Uvod

Razvoj umjetne inteligencije, pogotovo generativne umjetne inteligencije i velikih jezičnih modela, u zadnjih je nekoliko godina uvelike promijenio način na koji korisnici konzumiraju informacije te komuniciraju s digitalnim sustavima. Agenti temeljeni na ovim tehnologijama postaju sve prisutniji, kako u svakodnevnim životima korisnika, tako i u poslovnim procesima. Ovaj trend vidljiv je i na primjeru najpopularnijeg AI alata ChatGPT-a [1]. ChatGPT, koji osim korištenja samih LLM-ova korisnicima nudi i brojne agente, mjesečno bilježi više od 5 milijardi posjeta, što ga čini jednim od najposjećenijih web alata na svijetu [2].

Međutim, korištenje takvih web alata zatvorenog koda često otvara važna pitanja vezana uz privatnost, sigurnost i kontrolu nad podacima. Pri uporabi ovih alata, korisnici ne mogu uvijek na lak način i sa sigurnošću doći do informacija o tome kako se njihovi podaci prikupljaju, pohranjuj i obrađuju. Također, prilikom slanja informacija putem web-a, uvijek postoji i intrinzični rizik od neovlaštenog pristupa, presretanja ili manipulacije tijekom prijenosa podataka. Osim toga, korisnici nemaju mogućnost utjecaja na sigurnosne protokole i mjere zaštite koje takvi alati primjenjuju, što može dodatno povećati rizik od nepredviđenih propusta ili curenja podataka te korisnike ostavlja ovisnim o rješenjima pružatelja usluge. Ovo posebno dolazi do izražaja u kontekstu poslovne primjene, gdje obrada osjetljivih informacija zahtijeva visoku razinu povjerenja i transparentnosti.

Potaknut ovim izazovima, cilj ovog rada razviti je sustav temeljen na AI agentima Lumen koristeći lokalno pokrenute alate otvorenog koda. Lumen korisnicima pruža neke od najpopularnijih funkcionalnosti AI agenata poput obrade prirodnog jezika putem STT i TTS funkcionalnosti, slanje elektroničke pošte, generiranje slika, pretraživanje web-a i izvršavanje kalkulacija koristeći programski jezik Python. Ovaj pristup pruža veću razinu sigurnosti, fleksibilnosti i neovisnosti, jer korisnici zadržavaju potpunu kontrolu nad svojim podacima unutar vlastite infrastrukture.

# 2. Teorijska podloga

Razvoj složenih AI agentnih sustava zahtjeva dobro razumijevanje temeljnih pojmova i koncepata iz područja generativne umjetne inteligencije. U ovom poglavlju obrađena je teorijska osnova potrebna za razumijevanje načina rada sustava Lumen, uključujući pregled područja generativne umjetne inteligencije, velikih jezičnih modela i AI agenata.

## 2.1. Generativna umjetna inteligencija

Generativna umjetna inteligencija označava područje umjetne inteligencije usmjereno na stvaranje novog sadržaja temeljenog na naučenim uzorcima iz postojećih podataka [3]. Ovi sustavi oslanjaju se na duboke generativne modele čiji je cilj prepoznati i naučiti složene uzorke u podacima na kojima su trenirani te generirati nove podatke koji vjerno odražavaju originalne uzorke.

U pozadini ovih modela najčešće se nalaze duboke neuronske mreže. Duboke neuronske mreže predstavljaju naprednu vrstu umjetnih neuronskih mreža sastavljenih od velikog broja međusobno povezanih slojeva neurona. Svaki sloj dubokih neuronskih mreža podatke obrađuje na sve višoj razini apstrakcije, omogućavajući sustavu prepoznavanje kompleksnih obrazaca, struktura i ovisnosti unutar podataka. Ovaj tip neuronskih mreža posebno je učinkovit u učenju višedimenzionalnih i nelinearnih odnosa u podacima, što ih čini vrlo pogodnim za primjenu u zadacima kojima se bavi područje generativne umjetne inteligencije, poput generiranja teksta, slika, zvuka, videozapisa, animacija itd.

Primjeri generativnih modela uključuju generativne suparničke mreže (engl. *Generative Adversarial Network*, GAN), varijacijske autoenkodere (engl. *Variational Autoencoders*, VAE), transformerske modele i difuzijske modele. Generativne suparničke mreže se sastoje od dvije neuralne mreže: generatora, koji stvara podatke, i diskriminatora, koji procjenjuje jesu li ti podaci stvarni ili generirani. Ove mreže se treniraju u suparničkom odnosu, što dovodi do poboljšanja kvalitete generiranih rezultata. Kroz proces treniranja, generator postepeno generira sve više uvjerljive podatke, dok diskriminator postaje sve bolji u raspoznavanju stvarnih i generiranih podataka. GAN-ovi su posebno uspješni u području generiranja slika, animacija i dubokih lažnjaka (engl. *deepfakes*). Varijacijski autoenkoderi probabilistički su modeli koji uče sažeti ulazne podatke u latentni prostor niže dimenzionalnosti te ih rekonstruirati iz takve reprezentacije. Učenjem iz podataka i njihovih latentnih reprezentacija, ovi modeli mogu generirati nove, slične, podatke. Često se primjenjuju kod generiranje slika i sustava za preporuke. Transformeri predstavljaju vrstu neuronske mreže koja primjenjuje mehanizam samo-pažnje (engl. *self-attention*) za hvatanje dugoročnih ovisnosti u podacima. Koriste se kod zadataka obrade prirodnog jezika te su temelj velikih jezičnih modela. Difuzijski modeli su generativni modeli najčešće korišteni za generiranje slika temeljeni na procesu uklanjanja šuma. Generacija slike započinje šumom (npr. Gaussovom distribucijom) koji se postepeno, kroz niz koraka, čisti kako bi se dobila detaljna i realistična slika.

Od navedenih vrsta generativnih modela, sustav Lumen primjenjuje velike jezične modele i TTS modele temeljene na transformerima te difuzijske modele za generiranje slika. Veliki jezični modeli predstavljaju temelj agenta na kojem je zasnovan sustav Lumen. Agent pomoću velikih jezičnih modela obrađuje korisnikov upit, odabire potrebne radnje za ostvarivanje upita, generira odgovor itd. TTS modeli koriste se za pretvorbu generiranog teksta u govor, što omogućuje prirodniju interakciju s korisnicima putem glasa. Difuzijski modeli korišteni su kod implementacije funkcionalnosti generiranja slika. Ovi modeli na temelju korisnikovog upita generiraju traženu sliku.

U posljednje vrijeme, u području generativne umjetne inteligencije, posebno se ističu temeljni modeli (engl. *foundation models*). Temeljni modeli odnose se na bilo koji model treniran na širokom skupu podataka, najčešće korištenjem metoda samo-nadziranog učenja (engl. *self-supervised learning*). Temeljni modeli mogu se prilagoditi za rješavanje velikog broja različitih zadataka koristeći metode finog podešavanja (engl. *fine-tuning*) [4]. Ovi modeli nisu trenirani isključivo za jedan specifičan zadatak, već su u stanju generalizirati znanje i uz minimalno fino podešavanje obavljati čitav niz različitih zadataka. Temeljni modeli čine osnovu mnogih sustava temeljnih na umjetnoj inteligenciji. Njihova je primjena izrazito široka i pokriva gotovo sve domene koje uključuju obradu podataka. U području računalnog vida i obrade slika, koriste se za stvaranje fotorealističnih vizuala, umjetničkih ilustracija, 3D modela i animacija, kao i za automatsku obradu fotografija ili rekonstrukciju oštećenih dijelova slika. Kod domene obrade zvuka, ovi modeli koriste se za generiranje glasa na temelju teksta, generiranje glazbe u određenom stilu, sinteziranje raznih zvukova iz prirode itd. Još jedna uzbudljiva primjena temeljnih modela je primjena u području biomedicine i farmacije. Modeli trenirani na strukturama proteina koriste se za predviđanje načina presavijanja lanaca aminokiselina, što je ključno za razvoj novih lijekova i terapija. Ipak, najširа i najpopularnija primjena temeljnih modela je u području obrade prirodnog jezika. Veliki jezični modeli, koji su pod kategorija temeljnih modela, pronašli su primjenu u širokom spektru zadataka koji uključuju razumijevanje, generiranje i prevođenje teksta.

### 2.1.1. Upiti i inženjering upita

Za uspješno korištenje punog potencijala generativnih modela, važno je razumjeti koncepte vezane uz upite (engl. *prompt*) i inženjering upita (engl. *prompt engineering*). Upiti predstavljaju ulazne podatke modela, najčešće u tekstualnom obliku. Korisnik ove upite daje modelu kako bi izazvao željeni izlaz, bilo to generiranje određenog sadržaja, dobivanje odgovara na pitanja, rješavanje nekog problema i sl. Kvaliteta i struktura upita znatno utječe na kvalitetu izlaza modela, što je dovelo do razvoja područja inženjeringa upita. Inženjering upita je proces oblikovanja i optimizacije upita kako bi se maksimizirala učinkovitost, relevantnost i točnost izlaznih podataka generativnih modela [5]. Riječ je o vještini koja zahtijeva duboko razumijevanje načina na koji modeli procesuiraju jezik, kao i eksperimentalni pristup u kojem se različiti oblici uputa testiraju i poboljšavaju. Inženjering upita uključuje manipulaciju formulacije upita, dodavanje primjera unutar upita, strukturiranje upita kao niza naredbi, postavljanje ograničenja, definiranje stila odgovora, definiranje uloge modela i mnoge druge postupke s ciljem optimizacije odgovora modela.

Razlikujemo više vrsta upita, ovisno o njihovoj kompleksnosti i cilju. Najosnovniji oblik upita predstavljaju jednostavni upiti (engl. *zero-shot prompts*) koji se najčešće sastoje od jedne ili nekoliko rečenica te direktno opisuju zadatak. Primjerice: „Prevedi ovaj tekst na hrvatski.“ Ova vrsta upita korisna je kada korisnik želi brzo i jednostavno dobiti odgovor, osobito u slučajevima kada je zadatak jednoznačan i ne zahtijeva dodatni kontekst ili složeno razmišljanje. Iako jednostavni i laki za postavljanje te učinkoviti kod jednostavnijih zadataka, ovakvi upiti često nisu dovoljno precizni za složenije zadatke ili za generiranje odgovara u nekom specifičnom formatu. Model zbog nedostatka dodatnih smjernica može proizvesti odgovore koji nisu usklađeni s očekivanjima korisnika. Za složenije upite kod kojih je potrebno dodatno usmjeravanje modela ka željenom formatu odgovora, koriste se upiti s jednim ili više primjera (engl. *one-shot prompts* i *few-shot prompts*). Kod ove vrste upita, korisnici u upit uključuju i primjere ulaza i željenih izlaza kako bi se usmjerilo model prema generiranju odgovara u željenom obrascu. Time model dobiva dodatan kontekst o korisnikovim željama što često rezultira točnijim i preciznijim odgovorima modela koji su bolje usklađeni s korisnikovim očekivanjima. Također, za kompleksnije zadatke za čije je rješavanje potrebno više koraka, često se koriste i upiti lanca misli (engl. *chain-of-thought prompts*). Lanac misli predstavlja kompleksniju vrstu upita kod koje se model potiče da eksplicitno predstavi svoj misaoni proces te dođe do rješenja kroz veći broj koraka. Ovakva struktura upita omogućava modelu da razmišlja na glas, čime se smanjuje mogućnost pogrešnog odgovora i povećava razumljivost rezultata. Lanac misli može pomoći i korisniku da lakše razumije proces zaključivanja modela, što je osobito korisno u edukativnim kontekstima. Često su korišteni i upiti stabla misli (engl. *tree-of-thoughts prompts*). Ova tehnika slična je lancu misli, ali umjesto linearnog niza koraka koristi grananje. Ovaj pristup modelu omogućava da istovremeno razmotri i evaluira više alternativnih putanja misli te dođe do zaključka o najboljem toku za dolazak do rješenja na korisnikov upit. Kod ove tehnike model istražuje više različitih strategija razmišljanja prije konvergiranja na onu za koju smatra da će uroditi najboljim rezultatom. Svaka od navedenih vrsta upita dolazi s prednostima i manama. Područje inženjeringa upita se među ostalim bavim i odabirom optimalne vrste upite za dobivanje najboljeg mogućeg odgovora modela. Proces pronalaska optimalnog upita može se potpomognuti i tehnikom automatskih upita (engl. *auto-prompting*). Ova tehnika temelji se na automatskom generiranju, prilagođavanju i optimizaciji upita. Kod automatskih upita, umjesto da korisnik sam smišlja najbolji mogući upit, taj proces obavljaju specijalizirani modeli ili algoritmi. Ovi modeli i algoritmi automatski sami kreiraju ili modificiraju upite na temelju specifičnih kriterija ili povratnih informacija, čime se postiže bolja usklađenost odgovora modela s korisnikovim očekivanjima. Još jedna važna i često korištena vrsta upita su sistemski upiti (engl. *system prompts*). Sistemski upiti dostavljaju se modelu kao dodatak korisnikovom upitu, a služe za definiranje osnovnih pravila ili uputa koje model treba slijediti prilikom generiranja odgovora na korisnikov upit. Pomoću ove vrste upita definira se stil, ton, format i sl. za odgovore modela. Oni mogu uključivati smjernice o tome kako model treba tretirati korisničke upite, koje informacije treba izbjegavati ili naglašavati, kakav ton treba koristiti te kako se model treba ponašati u slučaju nejasnih ili dvosmislenih zahtjeva. Sistemski upiti najčešće se implementiraju kao početni ulaz u sustav prije svakog korisničkog upita. Time se osigurava da su pravila i smjernice trajno prisutne u ulaznim podacima modela. Sistemski upiti često služe

i za prilagodbu modela specifičnim domenama ili korisničkim potrebama. Na primjer, u edukativnom kontekstu, sistemski upit može zahtijevati od modela da odgovara na način prilagođen učenicima određene dobi, koristeći jednostavan jezik i objašnjenja. Sistemski upiti značajno utječu na cjelokupnu kvalitetu odgovora modela te je zato važno da su precizni i kvalitetni.

U kontekstu agentnih sustava, kao što je sustav Lumen, posebno značajan oblik upita predstavlja ReAct obrazac (od engl. *Reasoning and Acting*). ReAct obrazac predstavlja niz uputa koje se prosljeđuju agentu te pobliže objašnjavaju način na koji agent mora generirati odgovore i pozivati dostupne alate. Praćenjem ovog obrasca, agent dobiva sposobnost razmišljanja o akcijama koje treba provesti, izvršavanja tih akcija te promatranja rezultata svojih akcija na temelju kojih odlučuje o sljedećim koracima potrebnim za uspješno ispunjavanje korisnikovog upita. Tipičan ReAct format sadrži sljedeća polja:

- Thought (misao) – agent razmišlja naglas i objašnjava svoj trenutni zaključak ili plan za sljedeći korak.
- Action (akcija) – agent bira koji alat ili funkciju će koristiti.
- Action Input (ulazni podaci akcije) – ulaz koji se šalje odabranom alatu. Konkretan upit ili podatak.
- Observation (opservacija) – rezultat akcije, odnosno odgovor koji je agent dobio od alata.
- Final Answer (završni odgovor) – završni odgovor koji agent daje korisniku nakon što je prikupio dovoljno informacija.

Agent iterativno po potrebi koristi sve ili neke od prvih četiri navedenih polja obrasca dok ne zaključi da je korisnikov upit ispunjen. U tom trenutku agent generira izlaz koji sadrži završni odgovor, što označava kraj izvođenja agenta, te se tada završni odgovor dostavlja korisniku.

## 2.2. Veliki jezični modeli

Veliki jezični modeli predstavljaju oblik generativne umjetne inteligencije koji se fokusira na obradu, razumijevanje i generiranje prirodnog jezika pomoću dubokih neuronskih mreža. Temeljeni su na višeslojnoj arhitekturi transformera [6]. Razvoj velikih jezičnih modela predstavlja jedan od najvećih iskoraka današnjice u području obrade prirodnog jezika.

Jedan od glavnih problema obrade prirodnog jezika, pa tako i razvoj velikih jezičnih modela za koji je potrebna ogromna količina tekstualnih podataka, predstavlja ekstrakcija korisnih i strojno obradivih informacija iz nestrukturiranih tekstualnih podataka. Putem raznih internetskih izvora, knjiga, članaka i sl., javno je dostupna ogromna količina teksta pisanog prirodnim jezikom koja predstavlja bogat izvor informacija. Ovi tekstualni podaci su po svojoj prirodi diskretni i rijetki (engl. sparse) što ih čini izazovnima za obradu. Za potrebe strojne obrade, tekstualni podaci moraju se transformirati u strojno čitljive reprezentacije, najčešće u obliku numeričkih vektora koji opisuju značajke teksta. Kvaliteta tih reprezentacija uvelike

utječe na uspješnost daljnjih zadataka obrade prirodnog jezika. Tradicionalne metode kao što su one-hot kodiranje i učestalost pojma - obrnuta učestalost dokumenta (engl. *Term Frequency - Inverse Document Frequency*, TF-IDF) bile su prvi koraci u pretvaranju teksta u kvantitativni oblik. Ipak, ograničenja ovih metoda vidljiva su kod visoke dimenzionalnosti i rijetkosti vektora nastalih putem ovih metoda te njihovoj ograničenoj sposobnosti zadržavanja semantičkih i sintaktičkih informacija [7]. Razvojem dubokog učenja, pojavljuju se nove metode vektorske reprezentacije teksta koristeći AI modele temeljene na neuronskim mrežama. Ovi modeli automatski uče koje značajke teksta su relevantne te kako očuvati i reprezentirati semantičke sličnosti među tekstualnim jedinicama u vektorskom prostoru. Ovo omogućuje reprezentiranje tekst u obliku kontinuiranih i gustih vektora zvanim ugrađenim vektorima (engl. *embeddings*). Ugrađeni vektori su se pokazali prikladnijima za zadatke obrade prirodnog jezika, pogotovo kod velikih jezičnih modela, zbog svoje manje dimenzionalnosti, boljeg očuvanja semantičkih odnosa, veće sposobnosti generalizacije itd.

Pojednostavljeno, jezični modeli funkcioniraju tako da pokušavaju predvidjeti najvjerojatniju sljedeću riječ u rečenici na temelju prethodnog konteksta. Primjerice, ovisno o naučenim obrascima iz korpusa teksta na kojemu je model treniran, za rečenicu „Mačka se boji“ model bi kao najvjerojatniju sljedeću riječ mogao odabrati riječi poput „psa“, „vode“, „laveža“ itd. Kod velikih jezičnih modela, ovaj zadatak se rješava pomoću dubokih neuronskih mreža s milijardama parametara. Ovo velikim jezičnim modela dopušta učenje složenih jezičnih obrazaca, gramatičkih struktura, značenja riječi itd.

Proces treniranja velikih jezičnih modela složen je postupak od više koraka koji zahtjeva ogromne količine tekstualnih podataka. Treniranje najčešće uključuje sve ili neke od sljedećih faza: predtreniranje, fino podešavanje, učenje putem uputa (engl. *instruction tuning*) i treniranje pomoću ljudskog povratnog odgovora (engl. *Reinforcement Learning from Human Feedback*, RLHF). Kod predtreniranja, model uči opće obrasce u jeziku koristeći goleme količine nestrukturiranog teksta. Model se dodatno trenira na specifičnim skupovima podataka ili za izvršavanje specifičnih zadataka pomoću finog podešavanja. Kako bi model naučio bolje i pravilnije slijediti eksplicitne korisničke upute, primjenjuje se tehnika učenja prema uputama. Nadalje, koristi se proces treniranja pomoću ljudskog povratnog odgovora, gdje se tekst koji generira veliki jezični model ručno pregledava te se modelu daje povratna informacija o mogućnostima poboljšanja.

Među najpoznatijim velikim jezičnim modelima ističu se GPT (Generative Pre-trained Transformer) modeli koje razvija tvrtka OpenAI. Ovi modeli trenirani su na golemim količinama tekstualnih podataka i koriste se u širokom spektru primjena, uključujući automatizirano pisanje, analiziranje i prevođenje tekstova. Njihova sposobnost razumijevanja i generiranja prirodnog jezika omogućuje primjenu u mnogim industrijama poput obrazovanja, korisničke podrške, programiranja itd. Sposobni su generirati visokokvalitetne tekstove, odgovarati na složena pitanja i voditi smislene razgovore s korisnicima [8].

Drugi popularni primjer velikih jezičnih modela Google-ov je BERT (Bidirectional Encoder Representations from Transformers) model otvorenog koda. BERT koristi dvosmjerno kodiranje koje omogućava odlično razumijevanje konteksta unutar rečenica. Često se koristi za zadatke klasifikacije teksta, prepoznavanja entiteta i odgovaranja na pitanja. BERT je postao

temelj za razvoj brojnih varijacija i unaprijeđenih modela poput RoBERTa, DistilBERT i ALBERT, koji su optimizirani za bržu obradu, manju memorijsku potrošnju ili bolju točnost [9].

Sve veću popularnost stječu i drugi veliki jezični modeli otvorenog koda poput Llama, Falcon i Mistral modela. Llama (Large Language Model Meta AI) modeli, koje je razvila tvrtka Meta, poznati su po svojoj efikasnosti i visokokvalitetnoj generaciji teksta uz relativno male hardverske zahtjeve [10]. Falcon modeli, razvijeni od strane Technology Innovation Institute-a, posebno se ističu u zadacima generiranja teksta te su optimizirani za upotrebu u raznim zadacima u istraživačkim i industrijskim okruženjima [11]. Mistral modele razvija tvrtka Mistral AI. Cilj Mistral AI-a razviti je snažne, transparentne i učinkovite javno dostupne velike jezične modele [12]. Zbog svoje otvorene licence i odličnog omjera performansi i efikasnosti, Mistral modeli popularan su izbor među istraživačima i razvojnim timovima koji žele koristiti moćne jezične modele bez oslanjanja na komercijalne API-je.

U kontekstu sustava Lumen, korišten je predtreniran veliki jezični model otvorenog koda Solar 10.7B Instruct [13]. Model sadrži preko deset milijardi parametara te je napravljen s ciljem što boljeg razumijevanja i praćenja korisničkih uputa. Model je treniran na velikom korpusu raznovrsnog tekstualnog materijala te je dodatno fino podešen putem tehnike učenja prema uputama. Ovo ga čini pogodnim za implementaciju u agentni sustav, gdje je vrlo bitno pravilno praćenje uputa prilikom pozivanja dostupnih alata. U agentnim sustavima, ključno je koristiti veliki jezični model koji može pouzdano razumjeti kada i kako pozvati odgovarajući alat na temelju korisnikovog upita. Također, prilikom odabira modela bilo je važno odabrati efikasan model relativno male veličine zbog ograničenih računalnih resursa prilikom lokalnog pokretanja. Solar pruža dobar balans između veličine i efikasnosti te točnosti i preciznosti prilikom generiranja odgovora zbog čega je odabran za implementaciju u agentu na kojem se temelji sustav Lumen.

### 2.2.1. Nedostatci velikih jezičnih modela

Unatoč svojim impresivnim mogućnostima, veliki jezični modeli nisu bez nedostataka. Jedan od temeljnih kod velikih jezičnih modela leži u njihovoj inherentnoj ovisnosti o podacima na kojima su trenirani. Budući da se proces treniranja velikih jezičnih modela temelji na golemim količinama podataka iz raznih izvora, modeli često reproduciraju pristranosti prisutne u tim podacima. To može dovesti do generiranja sadržaja koji je netočan, kulturno osjetljiv, neetičan ili neprimjeren u određenim kontekstima. Dodatno, zbog činjenice da modeli nemaju stvarno razumijevanje svijeta, već funkcioniraju na temelju statističkih obrazaca u jeziku, nisu sposobni razlikovati istinite informacije od lažnih, što može uzrokovati halucinacijama, odnosno situacijama u kojima model generira uvjerljiv, ali potpuno netočan ili izmišljen sadržaj. Ovaj problem dodatno naglašava tendencija pretjeranoj pouzdanosti modela. Veliki jezični modeli često generiraju odgovore s visokom razinom sigurnosti čak i kada ne posjeduju relevantne informacije ili kada su podaci iz kojih bi mogli izvući zaključak nejasni ili kontradiktorni. To može biti posebno opasno u domenama gdje je točnost presudna, poput medicine, prava ili određenih poslovnih konteksta.

Nadalje, dodatni problem predstavlja i visoka potrošnja računalnih resursa tijekom treniranja i pokretanja velikih jezičnih modela. Treniranje modela često zahtijeva tisuće vrlo snažnih grafičkih kartica (engl. *Graphics Processing Unit*, GPU) koje danonoćno rade kroz tjedne ili mjesece, što rezultira znatnim financijskim troškovima te ima velik negativan ekološki učinak. Ovo treniranje nije samo skupo već i teško dostupno manjim istraživačkim institucijama i organizacijama koje nemaju pristup resursima velikih tehnoloških tvrtki.

Veliki jezični modeli (i generativna umjetna inteligencija općenito) postavljaju i brojna pitanja vezana uz autorska prava i plagiranje sadržaja. Budući da se veliki jezični modeli treniraju na ogromnim količinama tekstualnih podataka prikupljenih s interneta, često je nejasno jesu li svi ti podaci legalno dostupni za takvu upotrebu. Prilikom treniranja mnogih modela, u podacima su uključeni sadržaje iz knjiga, članaka, akademskih radova, internetskih foruma i drugih izvora koji su zaštićeni autorskim pravima. Iako se podaci često prikupljaju automatski bez direktne ljudske intervencije, to ne oslobađa organizacije koje razvijaju velike jezične modele od pravne i moralne odgovornosti poštivanja autorskih prava. Osim pravnog aspekta, postoji i etički problem vezan uz plagijat. Veliki jezični modeli mogu, namjerno ili nenamjerno, reproducirati dijelove originalnog teksta gotovo doslovno, osobito ako je taj sadržaj bio često zastupljen u skupu podataka na kojima je model treniran. To može dovesti do situacija gdje se generirani tekst pogrešno prikazuje kao originalan sadržaj, bez pravilnog navođenja izvora.

Kako bi se prevladali neki od navedenih nedostataka, razvijaju se različite strategije za unapređenje velikih jezičnih modela. Jedna od najčešćih tehnika je prije spomenuta metoda finog podešavanja modela na specifičnim skupovima podataka. Time se omogućava prilagodba opće treniranog modela specifičnim domenama, što rezultira boljim performansama i smanjenjem učestalosti halucinacija. Često se primjenjuje i metoda učenja iz konteksta (engl. *in-context learning*). Kod ove metode, modelu se daje dodatan kontekst o zadatku unutar samog upita. Ovaj pristup često donosi bolje performanse modela i ublažava halucinacije bez potrebe za finim podešavanjem, koje zahtjeva dodatne računalne resurse i vrijeme. Osim toga često se koriste i dodatni sustavi za detekciju i ublažavanje halucinacija. Primjerice, za detekciju i filtraciju halucinacija može se koristiti dodatni veliki jezični model koji provjerava istinitost tvrdnji u generiranom tekstu i izvorima. Također, smanjenje mogućnosti halucinacija i poboljšanje relevantnosti i preciznosti kod odgovora modela može se postići i primjenom tehnika generiranja potpomognutim dohvaćanjem (engl. *Retrieval-Augmented Generation*, RAG). RAG metode omogućuju integraciju ažurnih i provjerenih informacija iz baza podataka, datoteka, internetskih izvora i sl. RAG također pridonosi transparentnosti i provjerljivosti odgovora. Budući da se informacije koje model koristi mogu pratiti do konkretnih izvora, korisnici imaju mogućnost samostalno provjeriti točnost izjava modela, čime se potiče odgovornija upotreba umjetne inteligencije. Ova sposobnost referenciranja izvora pogotovo je važna u znanstvenim, akademskim i profesionalnim kontekstima, gdje je jasnoća o podrijetlu informacija presudna.

## 2.3. AI agenti

U širem smislu, pojam AI agent odnosi se na računalni sustav koji posjeduje sposobnosti percepcije svoje okoline, donošenja odluka i izvođenja radnja u svrhu postizanja određenog cilja. Agent je entitet koji opaža svoje okruženje pomoću senzora te na temelju tih opažanja djeluje kroz efektore ili izvršne module [14]. Kod digitalnih agenata, senzori mogu biti ulazni tekst korisnika, stanje aplikacije, mrežne informacije itd., dok su efektori najčešće pozivi programskim sučeljima aplikacija (engl. *Application Programming Interface*, API), generiranje sadržaja i sl.

U kontekstu generativne umjetne inteligencije i velikih jezičnih modela, agenti su sustavi koji koriste LLM-ove kao središnji mehanizam rezoniranja i odlučivanja. Možemo reći da u ovim agentnim sustavima veliki jezični model predstavlja „mozak“ agenta. Veliki jezični model, na temelju korisnikovog upita, odlučuje o potrebnim koracima koje agentni sustav mora izvršiti kako bi ispunio korisnikov zahtjev. Ti koraci često uključuju pozivanje nekih od dostupnih alata (npr. alati za generiranje slika, slanje elektronske pošte i sl.). U tom slučaju, veliki jezični model generira ulazne podatke za alat te ih prosljeđuje alatu. Alat zatim obrađuje LLM-ov zahtjev te vraća odgovor koji sadrži zatražene informacije, informaciju o uspješnosti izvršenja zahtjeva, podatke o greškama (ako je do njih došlo) itd. Agent do konačnog rješenja na upit dolazi iterativno. U svakom koraku, na temelju informacija poput korisnikovog upita, svoje memorije (odnosno prošlih poruka iz razgovora s korisnikom), izlaznih podataka alata itd. agent donosi odluke o sljedećem koraku kojeg mora napraviti za približavanje konačnom odgovoru na korisnikov upit. Kada agent zaključi da je izvršio zadatak, proces se zaustavlja te se korisniku dostavlja konačan odgovor na upit.

Agenti temeljeni na velikim jezičnim modelima često se dijele na sljedeće kategorije [15]:

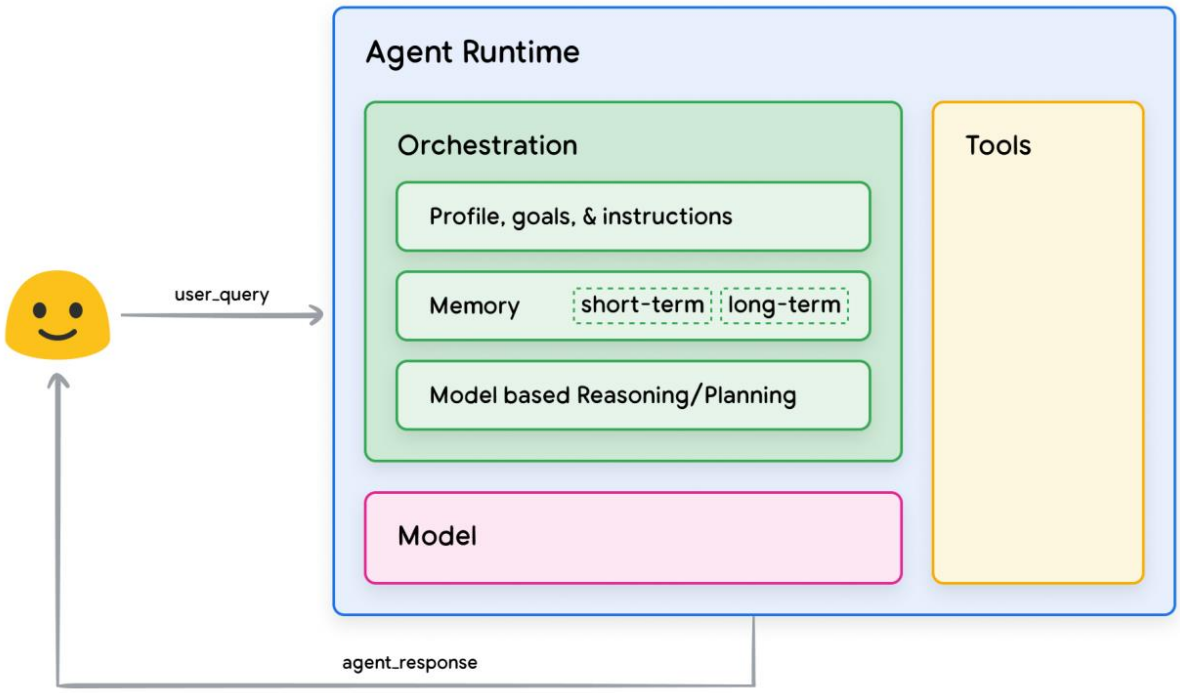
- Razgovorni agenti (engl. *conversational agents*): agenti koji komuniciraju s korisnicima u obliku dijaloga, imitirajući prirodan ljudski razgovor. Oslanjaju se na velike jezične modele. Tipični primjeri uključuju virtualne asistente i korisničke chatbotove. Za razliku od klasičnih reaktivnih sustava, mogu pamtiti kontekst razgovora i odgovarati suvislo čak i u složenijim interakcijama.
- Zadatkovno orijentirani agenti (engl. *task-oriented agents*): agenti specijalizirani za izvršavanje jasno definiranih zadataka kao što su rezervacije termina, obrada narudžbi ili automatizacija poslovnih procesa. Umjesto slobodnog dijaloga, ovi agenti slijede strukturirane tokove odluka kako bi učinkovito protumačili korisnikove zahtjeve i poduzeli odgovarajuće radnje. Njihov fokus je na funkcionalnosti i pouzdanosti, a manje na vođenju prirodnog razgovora.
- Kreativni agenti (engl. *creative agents*): koriste velike jezične modele za stvaranje novih sadržaja poput umjetničkih djela, glazbe, koda ili tekstova. Osmišljeni su tako da prepoznaju i primjenjuju ljudske estetske sklonosti i stilove, omogućujući korisnicima da brzo i automatizirano generiraju članke, pjesme, slike itd. Spadaju u novu generaciju alata koji proširuju kreativne kapacitete pojedinaca.

- Autonomni agenti (engl. *autonomous agents*): djeluju samostalno bez stalne kontrole korisnika, često u složenim i promjenjivim okruženjima. Obično imaju jasno definirane ciljeve i sposobnost samostalnog planiranja, evaluacije i izvršavanja radnji. Koriste se, primjerice, za automatizaciju opsežnih poslovnih procesa i obradu velike količine podataka.
- Suradnički agenti (engl. *collaborative agents*): dizajnirani su da rade u tandemu s ljudima ili drugim AI agentima na zajedničkim zadacima. Mogu podržavati timsku koordinaciju, dijeljenje informacija ili donošenje odluka u timovima. Tipična primjena uključuje agente za upravljanje projektima koji pomažu u usklađivanju rokova, resursa i komunikacije među članovima tima.
- Multimodalni agenti (engl. *multimodal agents*): agenti koji mogu procesirati i generirati sadržaje više vrsta, poput teksta, slike, zvuka ili videozapisa. Time korisnicima omogućuju veću razinu interaktivnosti i raznovrsnosti u usporedbi s agentima koji rade isključivo s tekstom. Na primjer, virtualni pomoćnik koji može prepoznati objekte na slici.
- Sustavi s više agenata (engl. *multi-agent systems*): agentni sustavi koji se sastoje od više specijaliziranih agenata. Ovi agenti međusobno surađuju kako bi zajednički riješili kompleksne zadatke. Takvi sustavi predstavljaju višu razinu organizacije umjetne inteligencije i često se koriste u dinamičnim i distribuiranim okruženjima.

AI agenti imaju mogućnost unaprijediti učinkovitost i kvalitetu rada pojedinaca i organizacija automatizacijom raznih digitalnih zadataka. Njihova sposobnost brze obrade velikih količina podataka, donošenja odluka u stvarnom vremenu i neprekidna dostupnost donosi ogromne prednosti pri optimizaciji poslovnih procesa, smanjenju troškova i povećanju produktivnosti. S druge strane, uporaba AI agenata sa sobom nosi i brojne izazove i rizike. Izlazne podatke agenata treba pažljivo promatrati zbog sklonosti AI sustava halucinacijama. Rizici su još veći nego kod klasičnih LLM sustava zbog dodataka alata koje agent, potaknut halucinacijom ili zlonamjernim korisničkim upitom, može koristiti na štetan način. Osim toga, moguća je i izrada namjerno zlonamjernih agenata, primjerice u svrhu širenja dezinformacija ili pronalaska sigurnosnih propusta u online sustavima. Stoga, iako AI agenti imaju ogroman potencijal kod unaprjeđenja efikasnosti, važno je AI agente razvijati na etičan način te sa sigurnošću na prvom mjestu.

### 2.3.1. Arhitektura AI agenata

AI agenti temeljeni na velikim jezičnim modelima sastoje se od tri glavne komponente: velikog jezičnog modela, alata i orkestracijskog sloja [16]. Za bolje razumijevanje AI agenata i njihovog načina rada, ključno je dobro razumjeti svaku od ovih komponenta te način na koji su povezane, što je prikazano na Slici 1.




Slika 1. Opća arhitektura agenta i komponente. Izvor: [16]

Kako je prikazano na Slici 1, jedna od glavnih komponenta agenta je veliki jezični model. Veliki jezični model predstavlja mozak AI agenta. Veliki jezični modeli detaljnije su pojašnjeni u poglavlju 2.2. U kontekstu agentnih sustava, veliki jezični modeli služe za obradu korisnikovih uputa, generiranja potrebnih koraka za odgovaranje na korisnikov upit, pozivanje potrebnih alata te generiranje konačnog odgovara na upit. Modeli se najčešće dodatno usmjeravaju korištenjem tehnika inženjeringa upita opisanih u poglavlju 2.1.1. poput ReAct obrasca, lanca misli, stabla misli i sistemskih upita. Za ispravan rad agenta, bitno je i odabrati prikladan veliki jezični model. Veliki jezični modeli se često razvijaju s određenim preddefiniranim ciljevima i predviđenim načinima korištenja. Primjerice, model koji je odličan u zadacima poput obrade prirodnog jezika i vođenja uvjerljivog razgovora s korisnicima ne mora nužno biti dobar i u zadacima koji zahtijevaju precizno praćenje uputa, rezoniranje i pozivanje odgovarajućih alata. Iz ovog razloga, za velike jezične modele često je dostupna i varijanta modela fino podešena za praćenje uputa (engl. *instruct models*) koja je najčešće prikladnija za integraciju u agentne sustave. Važno je napomenuti i da ovu komponentu agentnih sustava ne mora nužno sačinjavati samo jedan model. Često se koristi i kombinacija više modela koji su specijalizirani za pojedine zadatke te surađuju međusobno kako bi ostvarili zadane ciljeve. Primjerice, kod tehnike stabla misli, može se koristiti jedan agent za generiranje različitih grana razmišljanja te drugi agenti koji evaluiraju svaku pojedinu granu. Sustav može sadržavati i zaseban model koji na temelju evaluacija svake grane bira najbolju putanju te generira konačan odgovor.

Druga komponenta agentnih sustava su alati dostupni agentu. Iako veliki jezični modeli posjeduju iznimne sposobnosti obrade prirodnog jezika i rezoniranja, oni su po svojoj prirodi zatvoreni sustavi bez mogućnosti izravne komunikacije s vanjskim sustavima. Zbog ovog nedostatka, veliki jezični modeli sami po sebi nemaju pristup ažuriranim podacima i

mogućnosti interakcije s vanjskim svijetom, izuzev komunikacije s korisnikom putem korisničkih upita i odgovora modela. Kako bi se ovaj problem prevladao, velikim jezičnim modelima daje se pristup raznim alatima. Ovi alati često se dijele u tri kategorije: ekstenzije, funkcije i spremišta podataka. Ekstenzije najčešće podrazumijevaju alate koji agentu dopuštaju interakciju s vanjskim sustavima putem API poziva. Kako bi koristio ove alate, agent generira ulazne podatke koji se šalju vanjskim sustavima pomoću API poziva. Agentu se zatim prosljeđuje odgovor poziva, koji najčešće uključuje informacije koje je agent zatražio i/ili informacije o uspješnosti API poziva. Ovakvi alati agentu primjerice dopuštaju slanje elektroničke pošte, dohvaćanje informacija o vremenskim prognozama i sl. Funkcije su slična vrsta alata, ali se razlikuju po tome što ne uključuju direktno pozivanje API-ja od strane agenta, već agent poziva lokalne funkcije definirane u programskom kodu sustava. Spremišta podataka omogućuju agentu pristup raznim strukturiranim i nestrukturiranim podacima, poput PDF datoteka, Excel tablica, HTML sadržaja itd. Ovi sadržaji se najčešće vektoriziraju i pohranjuju u vektorske baze radi efikasnijeg dohvaćanja relevantnog sadržaja primjenom RAG tehnika. Time se modelu pruža pristup ažuriranim i relevantnim informacijama, što može uvelike poboljšati točnost generiranih odgovora.

Sljedeću važnu komponentu agentnih sustava predstavlja orkestracijski sloj. Ovaj dio sustava zadužen je za nadzor svih koraka koje agent poduzima u postizanju cilja. Ti koraci uključuju sve od primitka korisnikovog upita, planiranja, izvršavanja radnji, evaluacije rezultata, odlučivanja o sljedećim koracima itd. Orkestracijski sloj djeluje kao kontrolna petlja koja se neprekidno izvršava dok agent ne zaključi da je zadatak završen. U toj petlji koristi se kombinacija informacija iz memorije agenta, stanja razgovora s korisnikom i rezultata iz prethodnih koraka. Na temelju svih dostupnih informacija, agent koristi veliki jezični model za rezoniranje (npr. putem ReAct obrasca) te odlučuje koje će alate koristiti, koje će parametre poslati alatima, i interpretira rezultate kako bi odlučio o sljedećim koracima. Upravo ova sposobnost iterativnog planiranja i korištenja alata razdvaja agentne sustave od samih velikih jezičnih modela.

# 3. Razvoj i implementacija sustava Lumen

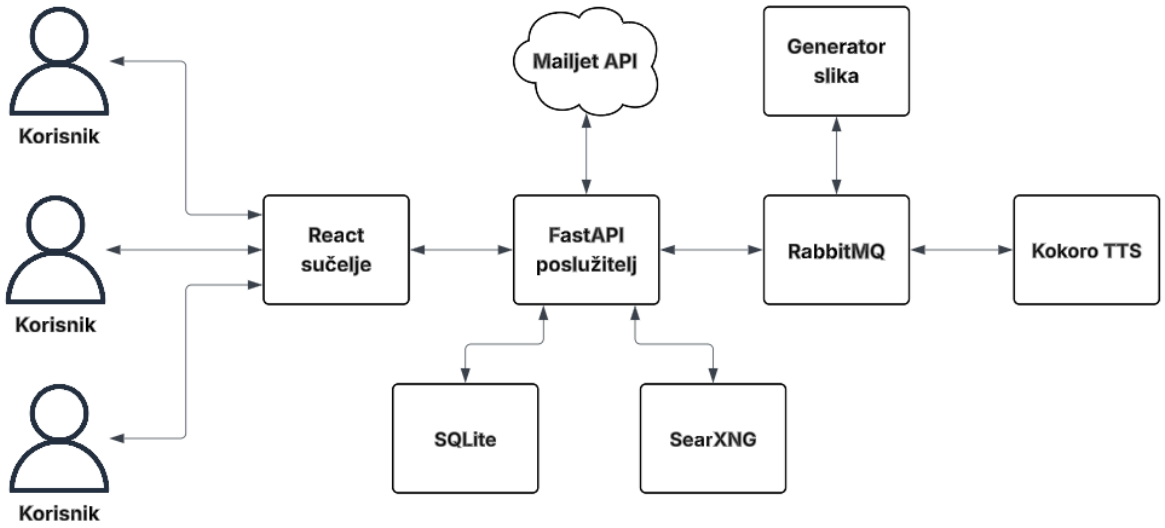
U ovom poglavlju detaljno je opisan cijeli proces razvoja, implementacije i objavljivanja agentnog sustava Lumen. Na kraju poglavlja objašnjene su i potencijalne mogućnosti daljnjeg razvoja sustava te njegove tehničke specifikacije. Sustav Lumen predstavlja agentni sustav temeljen na velikim jezičnim modelima i tehnologijama otvorenog koda koji korisnicima pruža funkcionalnosti poput obrade prirodnog jezika putem STT i TTS funkcionalnosti, slanja elektroničke pošte, generiranja slika, pretraživanja web-a i izvršavanja kalkulacija koristeći programski jezik Python. Sustav sadrži i moderno i responzivno korisničko sučelje izrađeno primjenom Reacta.

## 3.1. Razvoj poslužiteljske strane sustava

Razvoj poslužiteljske strane sustava Lumen sačinjavali su koraci poput odabira arhitekture sustava, odabira LLM-a i njegove integracije u sustav, osmišljanja arhitekture podataka te implementacije baze podataka, odabira te integracije ostalih alata potrebnih za ostvarivanje svih funkcionalnosti sustava itd.

### 3.1.1. Pregled arhitekture sustava

Sustav se sastoji od više povezanih servisa. Korisnici pristupaju sustavu putem React sučelja. Sučelje komunicira s poslužiteljskom stranom sustava putem API poziva FastAPI poslužitelju. Ovaj servis sadrži agenta i LLM na kojemu se temelji. Servis je povezan i s bazom podataka SQLite koja pohranjuje podatke o korisnicima te njihovim porukama i razgovora s agentom. Agent ima pristup raznim alatima koje proizvoljno koristi na temelju upita korisnika. Ti alati obrađuju upite agenta te mu vraćaju informacije potrebne za donošenje daljnjih koraka za ispunjenje korisnikovog upita. Agent šalje zahtjeve alatima putem RabbitMQ posrednika za poruke, osim kod alata za slanje elektroničke pošte, koji funkcionira pomoću Mailjet API-ja. Ostali alati uključuju generator slika i SearXNG alat za internetske pretrage. Nakon što agent utvrdi da je korisnikov upit ispunjen, generira završni odgovor koji se, također preko RabbitMQ-a, šalje Kokoro TTS servisu za generiranje glasovnog odgovora. Tekstualni, glasovni i po potrebi slikovni odgovori šalju se React sučelju te se poruke pohranjuju u bazu podataka. Na Slici 2. prikazana je navedena arhitektura sustava Lumen u obliku dijagrama. Svi navedeni servisi detaljnije su opisani u narednim poglavljima.




Slika 2. Dijagram arhitekture sustava Lumen.

### 3.1.2. Baza podataka

Za bazu podataka sustava, odabran je SQLite. SQLite je relacijska baza podataka otvorenog koda poznata po svojoj jednostavnosti, brzini i kompaktnosti [17]. Sustav u bazu pohranjuje informacije o korisnicima i njihovim razgovorima s agentom. Baza se sastoji od četiri tablice koje su opisane u nastavku.

**Users** – tablica koja pohranjuje podatke o korisnicima.

Stupci tablice Users:

- id – primarni ključ. Cjelobrojni tip podataka. Jedinstveno identificira svakog korisnika.
- email – tekstualni tip podataka. Pohranjuje e-mail adresu korisnika.
- password_hash – tekstualni tip podataka. Pohranjuje hash lozinke korisnika.
- is_temporary_password – boolean vrijednost. Označava je li korisnik promijenio inicijalnu lozinku.
- created_at – timestamp. Vrijeme kada je korisnički račun kreiran.
- is_admin – boolean vrijednost. Označava ima li korisnik administratorske ovlasti.

**Chats** – tablica koja pohranjuje informacije o razgovorima korisnika.

Stupci tablice Chats:

- id – primarni ključ. Cjelobrojni tip podataka. Jedinstveno identificira svaki razgovor.
- user_id – vanjski ključ koji referencira users(id). Označava korisnika kojem razgovor pripada.
- title – tekstualni tip podataka. Naslov razgovora.

- created_at – timestamp. Vrijeme kada je razgovor kreiran.

**Messages** – tablica koja pohranjuje pojedinačne poruke unutar razgovora.

Stupci tablice Messages:

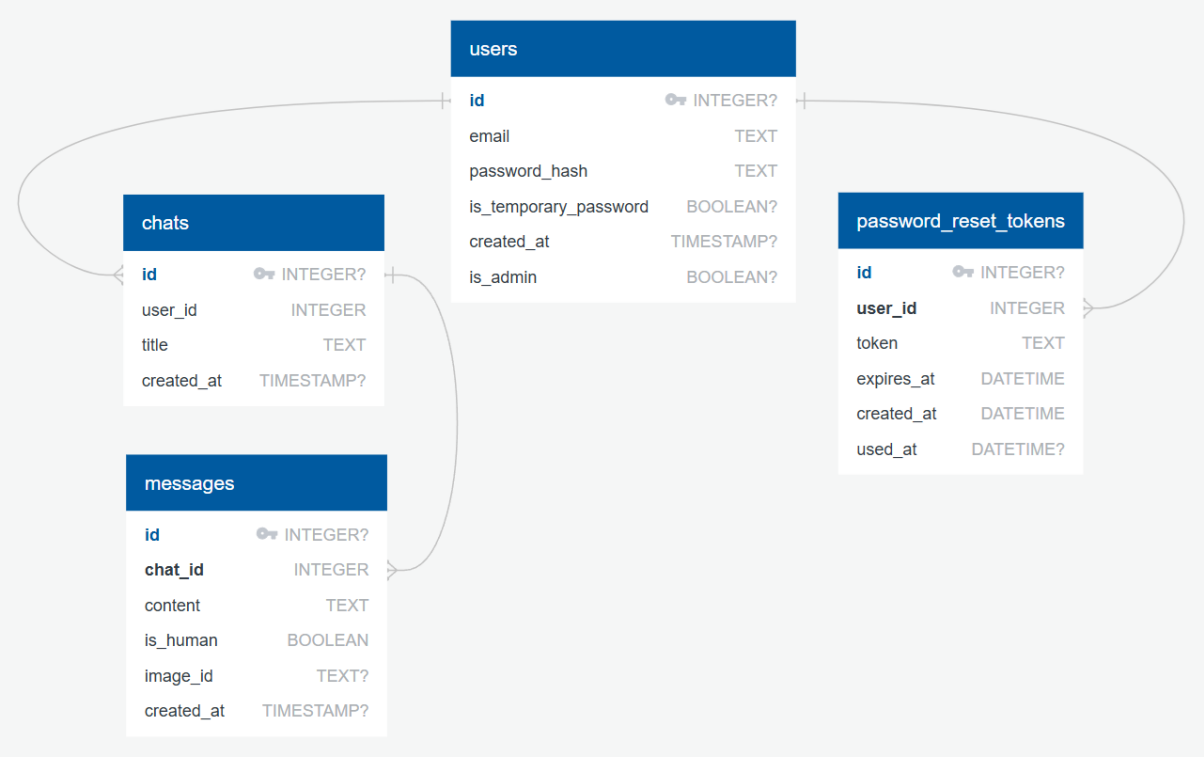
- id – primarni ključ. Cjelobrojni tip podataka. Jedinstveno identificira svaku poruku.
- chat_id – strani ključ koji referencira chats(id). Označava kojem razgovoru pripada poruka.
- content – tekstualni tip podataka. Sadržaj poruke.
- is_human – boolean vrijednost. Označava je li poruku napisao korisnik ili agent.
- image_id – tekstualni tip podataka (opcionalan). Može sadržavati ID slike povezane s porukom.
- created_at – timestamp. Vrijeme kada je poruka poslana.

**Password_reset_tokens** – tablica koja pohranjuje informacije o tokenima za resetiranje lozinke.

Stupci tablice Password_reset_tokens:

- id – primarni ključ. Cjelobrojni tip podataka. Jedinstveno identificira svaki token.
- user_id – vanjski ključ koji referencira users(id). Označava korisnika za kojeg je token izdan.
- token – tekstualni tip podataka. Token koji se koristi za resetiranje lozinke.
- expires_at – datetime. Vrijeme isteka tokena.
- created_at – datetime. Vrijeme kada je token generiran.
- used_at – datetime (opcionalan). Vrijeme kada je token iskorišten.

Slika 3. prikazuje Dijagram entiteta i veza baze podataka na kojem su vidljive navedene tablice.




Slika 3. Dijagram entiteta i veza baze podataka.

### 3.1.3. FastAPI poslužitelj

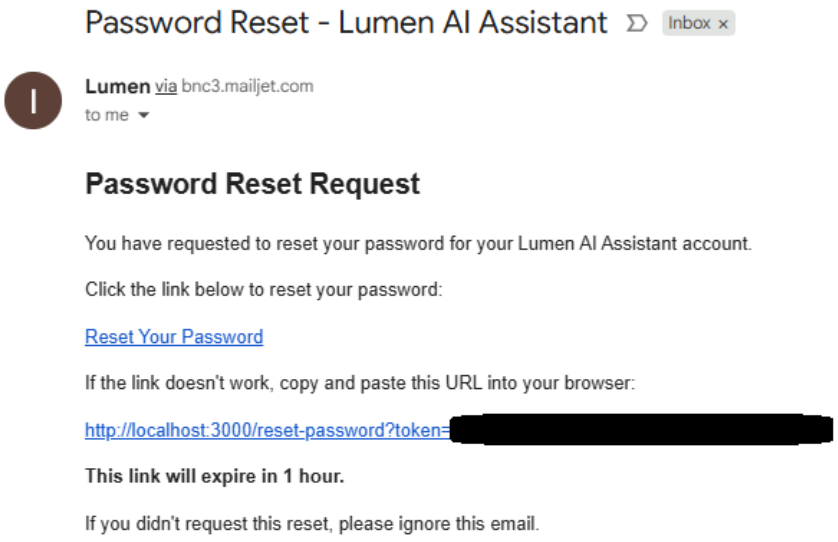
FastAPI je razvojni okvir u programskom jeziku Python za izradu API-ja. Odabran je kao temelj poslužiteljske strane sustava Lumen zbog svoje brzine i jednostavnosti. Ovaj dio sustava sadrži sve rute potrebne React sučelju za ispravan rad , uključujući:

- Rute vezane uz autentifikaciju korisnika i dohvaćanje informacija o korisnicima.
- Rute vezane uz oporavak lozinke korisnika.
- Rute za stvaranje, dohvaćanje i brisanje podataka o razgovorima i porukama korisnika iz baze podataka.
- Rute za komunikaciju s agentom.
- Rute namijenjene administratoru, vezane za dohvaćanje, stvaranje i brisanje korisnika.

Na sve rute postavljene su odgovarajuće sigurnosne mjere kako bi korisnici mogli pristupiti samo podacima koji im pripadaju te kako bi se korisnicima zabranio pristup rutama namijenjenim administratorima sustava, prateći načelo najmanje privilegije.

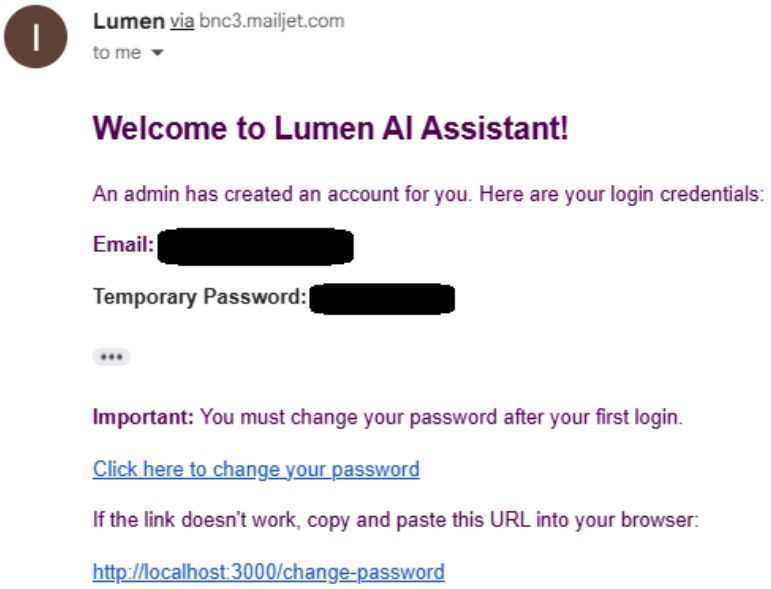
Autentifikacija korisnika implementirana je koristeći JSON web tokene (engl. *JSON Web Token*, JWT) zajedno s podacima o korisnicima spremljenim u bazi podataka. Kada korisnik pošalje zahtjev za prijavu, provjerava se ispravnost poslanih podatka za prijavu usporedbom s podacima iz base podataka. Ako je utvrđeno da su podaci ispravni, generira se JWT koji se vraća korisniku kao odgovor te služi kao način autentifikacije korisnika pri pristupanju ostalim rutama.

Sustav također podržava funkcionalnost oporavka lozinke. Korisnik šalje zahtjev za oporavak lozinke na temelju svoje adrese elektroničke pošte. U slučaju da navedena adresa postoji u bazi korisnika, generira se token za oporavak lozinke koji se sprema u bazu te šalje na adresu u obliku poveznice na stranicu namijenjenu za oporavke lozinke. Na ovoj stranici, nakon što poslužitelj provjeri valjanost tokena, korisnik može odabrati novu lozinku. Na Slici 4. prikazan je primjer elektronske pošte za oporavak lozinke koju korisnik dobiva prilikom slanja zahtjeva za oporavak lozinke. U produkcijskom okruženju, poveznica ne vodi na *localhost* adresu, već na domenu na kojoj je sustav objavljen.




Slika 4. Primjer elektronske pošte za oporavak lozinke.

Dodavanje novih korisnika nije implementirano na klasičan način putem stranice za registraciju zbog hardverskih ograničenja. Računalo na kojem je objavljen sustav, o kojem će se više pričati u kasnijim poglavljima, nema dovoljno računalne snage za istovremeno posluživanje većeg broja korisnika zbog visokog vremena čekanja na odgovore na korisničke zahtjeve. Umjesto toga, korisnici mogu od administratora sustava zatražiti stvaranje korisničkog računa na temelju adrese elektroničke pošte. Administrator tada može, pomoću svog sučelja, stvoriti novog korisnika te mu na adresu poslati inicijalne podatke za prijavu. Prilikom prve prijave, od korisnika se traži promjena inicijalne lozinke. Na Slici 5. prikazan je primjer elektronske pošte koju dobivaju novi korisnici sustava. U produkciji, poveznica ne vodi na *localhost* adresu, već na domenu na kojoj je sustav objavljen.




Slika 5. Primjer elektroničke pošte koju dobivaju novi korisnici sustava.

### 3.1.4. AI Agent i veliki jezični model

Glavnu komponentu sustava Lumen predstavlja AI agent temeljen na velikim jezičnim modelima. Za razvoj agenta, korišten je Python razvojni okvir otvorenog koda LangChain. LangChain služi za razvoj aplikacija temeljenih na LLM-ovim te olakšava proces pokretanja LLM-ova te njihovog povezivanja s vanjskim podacima i alatima [18]. Neke od podržanih funkcionalnosti unutar LangChain-a su:

- Integracija s vanjskim podacima (poput baza podataka, dokumenata, internetskih izvora itd.).
- Korištenje vlastitih i preddefiniranih alata.
- Memorija, odnosno sposobnost agenta da pamti prošle dijelove razgovora.
- Agentna arhitektura, gdje LLM može sam odlučiti koje alate i na koji način koristiti.

Za ovaj projekt odabran je Solar 10.7B Instruct v1.0 model (u nastavku Solar). Ovaj je LLM odabran zbog relativno male veličine modela te iznimno dobrim performansama. Zbog hardverskih ograničenja i velikog broja servisa koji se moraju lokalno pokrenuti za ispravan rad sustava Lumen, glavni čimbenik pri odabiru modela bila je njegova veličina. Iako postoje vrlo uspješni modeli i s malo manjim brojem parametara poput Mistral 7B i Llama 3 8B, odabran je Solar jer predstavlja dobar balans između veličine i snage modela te pobjeđuje modele sličnih veličina po brojnim metrikama [13]. Zbog puno šireg izbora modela koji podržavaju engleski, naspram ostalih jezika, odabrani model podržava samo engleski jezik te je i cijeli sustav Lumen zamišljen kao web aplikacija na engleskom jeziku. Odabrana je Instruct verzija Solar modela, koji predstavlja originalni Solar 10.7B model fino podešen za bolje razumijevanje i praćenje uputa korisnika. Ova karakteristika je vrlo važna kod izrade agenta, jer agenti moraju precizno i jasno pratiti način pozivanja alata temeljen na ReAct obrascu.

ReAct obrazac sastoji se od niza uputa agentu koje pobliže objašnjavaju način na koji agent mora generirati odgovore i pozivati dostupne alate. ReAct obrazac detaljnije je objašnjen u poglavlju 2.1.1. Slijedi konkretan primjer korištenja ReAct obrasca unutar sustava Lumen za odgovaranje na korisnikov upit:

```
Korisnikov upit: What is the weather like in Rijeka?

Thought: To answer this question, I will search for current weather conditions in Rijeka.
Action: searx_search
Action Input: "current weather conditions Rijeka"
Observation: Rijeka, Croatia Weather Forecast, with current conditions ...

Thought: The weather in Rijeka is currently sunny, with a temperature of 82°F (31°C). The
forecast indicates mostly clear skies for the next few days.
Final Answer: The weather in Rijeka right now is sunny and around 82°F (31°C), with mainly
clear skies predicted in the following days.
```

Za brži rad modela, kalkulacije potrebne za njegov rad izvršavaju se na grafičkoj kartici računala pomoću CUDA platforme. CUDA je računalna platforma i programski model koji je razvila tvrtka NVIDIA, a omogućuje programerima korištenje GPU-ova za opće računalne zadatke [19]. CUDA omogućuje da se GPU, koji se inače koristi za prikaz slike i videa, koristi i za izvođenje složenih računalnih operacija, poput znanstvenih izračuna, strojnog učenja, video obrade itd. Ovaj pristup često daje znatno brže performanse od izvođenja istih operacija pomoću centralnog procesora računala (engl. *Central Processing Unit*, CPU). CPU u pravilu sadrži manje broj vrlo snažnih jezgri (tipično 4 do 16), dok GPU sadrži tisuće manjih jezgri te više aritmetičko-logičkih jedinica (engl. *Arithmetic Logic Unit*, ALU) koje mogu istovremeno raditi operacije. Rad LLM-ova zahtjeva velik broj kalkulacija koje uključuju množenje velikih matrica i tenzora, što je zadatak koji se može višestruku ubrzati izvođenjem paralelno na tisućama jezgri koje pruža GPU.

Kako bi korisnici imali bolji uvid u radnje koje vrši agent i stadij u kojemu se nalazi njihov zahtjev, poslužitelj nakon svake iteracije agenta sučelju šalje podatke o agentovim akcijama. Npr. kada agent koristi alat za generiranje slika, dok korisnik čeka na odgovor umjesto generičke „Please wait…“ poruke pokazati će mu se poruka „Generating image…“. Ovo je postignuto pomoću poslužiteljski poslanih događaja (engl. *Server-Sent Events*, SSE) gdje poslužitelj kontinuirano šalje poruke sučelju o trenutnim koracima agenta koji se u React aplikaciji prikazuju korisniku u stvarnom vremenu.

Također, agent posjeduje funkcionalnost memorije, odnosno svjestan je prethodnih poruka između sebe i korisnika. Ovo je postignuto dohvaćanjem informacijama o porukama poslanim unutar razgovora iz baze podataka te njihovim dodavanjem korisnikovim upitima.

Agent ima pristup osnovnim informacijama o svojem imenu (Lumen), ulozi te stvaratelju putem sistemske poruke:

```
„You are a helpful AI assistant named Lumen. You where created by Bojan Radulović.“
```

### 3.1.5. Alati

Agent za uspješno izvršavanje korisničkih upita na raspolaganju ima nekoliko alata koje može proizvoljno koristiti. Za neke alate postoji ugrađena podrška unutar LangChain razvojnog

okvira, dok su neki alata ručno definirani. Kod definiranja alata navode se ulazne varijable i njihovi tipovi te opis funkcionalnosti alata. Ove informacije prosljeđuju se agentu kako bi imao uvid u pravilan način korištenja alata. Agent na temelju korisnikovog upita sam odlučuje treba li koristiti alate te, ako da, koje alate treba upotrijebiti za dobivanje najboljeg mogućeg odgovora na korisnikov upit. Dostupni alati su:

- SearXNG – alat za pretraživanje interneta. Omogućuje agentu pristup najnovijim informacijama s web-a. Agent koristi ovaj alat kada korisnički upit zahtjeva informacije koje nisu dostupne lokalno, kao što su aktualne vijesti, cijene, vremenska prognoza itd.
- Python interpreter – služi za izvođenje složenih numeričkih operacija pomoću programskog jezika Python. Omogućuje agentu izvršavanje Python koda u svrhu rješavanja matematičkih problema.
- Generator slika – omogućuje generiranje slika na temelju tekstualnog opisa. Agent koristi ovaj alat kada korisnik zatraži vizualizaciju neke ideje, ilustraciju, konceptualni prikaz itd.
- Mailjet API – služi za slanje elektroničke pošte. Omogućuje agentu slanje elektroničke pošte.

Navedeni alati detaljnije su opisani u nastavku.

#### 3.1.5.1. SearXNG

Jedan od alata dostupnih agentu je alat za pretraživanje interneta. Ovaj alat temeljen je na SearXNG-u [20]. SearXNG je meta-pretraživač otvorenog koda koji istovremeno šalje zahtjeve na nekoliko tražilica te objedinjuje rezultate na jednom mjestu.

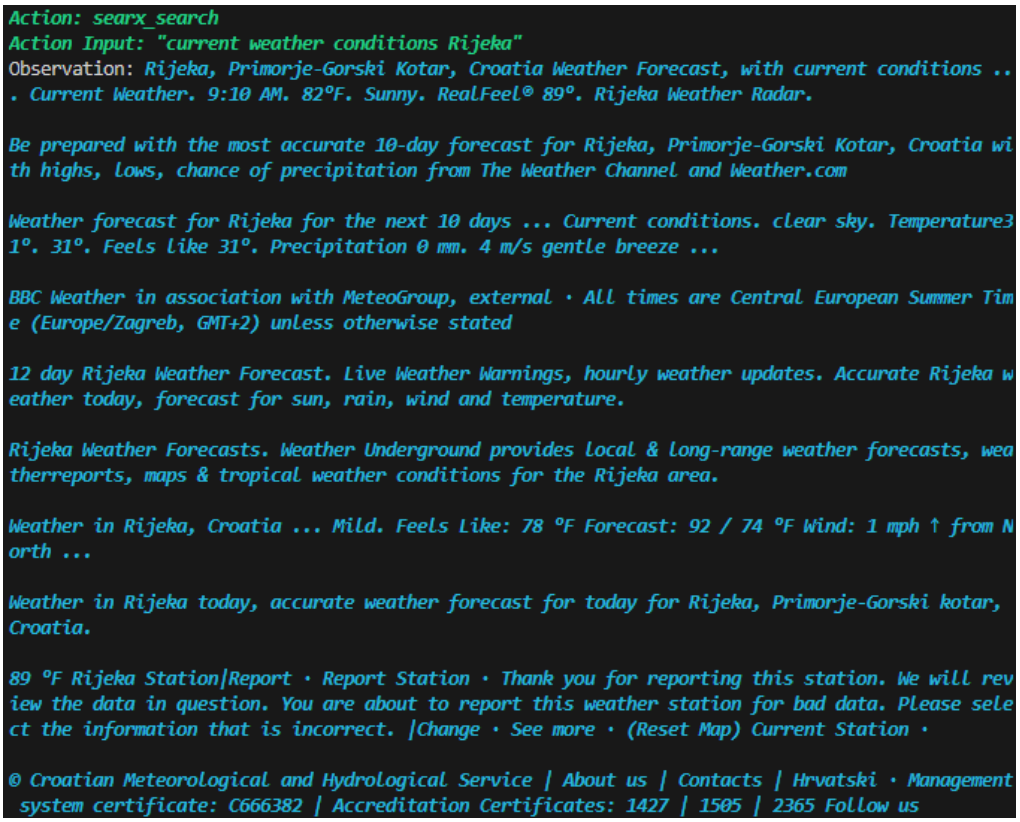
Glavne značajke ovog pretraživača, zbog kojih je odabran prilikom razvoja sustava Lumen su:

- Otvoreni kod – potpuni programski kôd SearXNG-a javno je dostupan na GitHub platformi.
- Mogućnost lokalnog pokretanja – SearXNG se može lako pokrenuti lokalno pomoću Docker slike na vlastitom računalu ili serveru.
- Laka integracija s LangChain-om – LangChain sadrži ugrađenu funkciju za lakše stvaranje alata koja podržava SearXNG. Ovo uklanja potrebu pisanja programskog koda za izradu vlastitog alata.
- Anonimnost i privatnost – ne prikuplja osobne podatke.
- Višestruki izvori – SearXNG agregira rezultate s velikog broja izvora i tražilica.
- Aktivna zajednica – SearXNG vodi aktivna zajednica GitHub entuzijasta koji svakodnevno rade na njegovom unaprjeđenju i održavanju.

SearXNG se kod inicijalizacije sustava Lumen pokreće u obliku Docker kontejnera. Prilikom inicijalizacije agenta, kod definiranja alata, korištena je LangChain-ova ugrađena funkcija

load_tools koja podržava SearXNG. Kod ovog koraka definira se i koji port koristiti za komunikaciju s SearXNG-om.

Sam alat koji povezuje agenta i SearXNG kao ulazni argument prihvaća upit koji agent želi pretražiti. Taj upit se šalje SearXNG Docker kontejneru koji pretražuje web te vraća agregirane rezultate pretraživanja. Ti se rezultati dostavljaju agentu u obliku opservacije. Slika 6. prikazuje primjer rezultata pretraživanja pomoću SearXNG. U ovom primjeru agent je zatražio pretragu informacija o trenutnom meteorološkom stanju u gradu Rijeci te kao odgovor dobio agregirane rezultate pretraživanja.




Slika 6. Primjer rezultata pretraživanja pomoću SearXNG-a.

### 3.1.5.2. Python interpreter

Agent također ima pristup Python REPL (Read-Eval-Print Loop) alatu. Python REPL je interaktivno okruženje u kojem je moguće upisati kôd, izvršiti ga te odmah dobiti rezultate izvršavanja [21]. Ovaj alat dopušta agentu izvršavanje Python koda u stvarnom vremenu. Glavna ideja iza implementacije ovog alata u sustav Lumen bila je omogućiti agentu korištenje Python interpretera za izvršavanje raznih kalkulacija i analizu danih podataka. Primjerice, agent može koristiti ovaj alata kako bi prebrodio činjenicu da su LLM-ovi zloglasno loši kod rješavanja matematičkih problema. Temeljna svrha LLM-ova obrada je prirodnog jezika i generiranje tekstualnih izlaza te stoga LLM-ovi nemaju pravo temeljno znanje matematičkih koncepata i načina na kojih matematičari dolaze do rješenja [22]. Pomoću ovog alata, agent može prebroditi ovaj i slične probleme. Python REPL alat je također ugrađen u LangChain što čini njegovu integraciju s agentom vrlo laganom i jednostavnom.

Slijedi primjer agentovog korištenja ovog alata za matematički izračun:

```
Korisnikov upit: What is 321413^2

Thought: To find the square of a number, we can multiply it by itself. However, for large
numbers, manual calculation may be difficult. We could use the python_repl tool to
calculate this value for us.
Action: python_repl
Action Input: `print(321413**2)`
Observation: 103306316569

Thought: The result of 321413^2 is 103306316569.
Final Answer: 103306316569
```

Slijedi primjer agentovog korištenja ovog alata za generiranje nasumičnog broja:

```
Korisnikov upit: Generate a random number between 1 and 10000

Thought: To generate a random number between 1 and 10000, we can use Python.
Action: python_repl
Action Input: print(random.randint(1, 10000))
Observation: NameError("name 'random' is not defined")

Thought: We need to import the random module first.
Action: python_repl
Action Input: import random; print(random.randint(1, 10000))
Observation: 2233

Thought: Now that the number has been generated, I can share it with the user.
Final Answer: The randomly generated number between 1 and 10000 is 2233.
```

Korištenje ovog alata sa sobom donosi i mnoga sigurnosna pitanja. Python REPL može izvršiti arbitrarni kôd na poslužitelju te ga treba koristiti s oprezom. Zlonamjerni korisnik ili halucinacija agenta mogla bi rezultirati izvršavanjem štetnog programskog koda. U produkciji, na ovaj alat je važno primijeniti sigurnosne mjere poput izolacije izvršavanja koda u zasebno virtualno okruženje, filtriranja ulaznih podataka u alat, onemogućavanja korištenja određenih Python paketa i modula itd. Osim toga, mogu se razmotriti i druge opcija alata, npr. izrada alata specifično namijenjenog za matematički kalkulacije. Ovo rješenje je sigurnije, ali smanjuje raspon funkcionalnosti agenta.

#### 3.1.5.3. Generator slika

Još jedan alat dostupan agentu je alat za generiranje slika. Temelj ovog alata je AI model za generiranje slika. Korišten je osnovni model Stable Diffusion XL Base 1.0 (dalje u tekstu SDXL) sa SDXL Lightning 2step unet (dalje u tekstu SDXL Lightning) težinama.

SDXL je napredni model za generiranje slika pomoću AI kojeg je razvila tvrtka Stability AI. Riječ je o difuzijskom modelu koji, na temelju tekstualnog opisa, može generirati visoko kvalitetne slike [23]. Tehniku difuzije, koju model koristi za generiranje slika, započinje s nasumičnim šumom, odnosno nasumičnim pikselima, te postupno rekonstruira sliku prema uputama zadanim u tekstualnom opisu slike.

SDXL Lightning je brža i efikasnija varijanta osnovnog SDXL modela. Ovaj model dobiva na brzini tako da proces rekonstrukcije slike obavlja u samo 2 do 4 koraka, umjesto desetaka koraka kod klasične difuzije. Ovaj pristup daje znatno bolje performanse uz relativno mali

gubitak na kvaliteti generiranih slika [24]. Na Slici 7. prikazana je usporedba slika generiranih pomoću SDXL i SDXL Lightning modela. Iz slika je vidljiv gotovo zanemariv gubitak na kvaliteti slika generiranih pomoću SDXL Lightning modela.




Slika 7. Usporedba slika generiranih pomoću SDXL i SDXL Lightning modela. Izvor: [25]

Za ovaj alat ne postoji ugrađena varijanta unutar LangChain-a pa je bilo potrebno definirati vlastiti alat. Ulazni podatak alata je opis slike koju agent želi generirati. Za alat je napisan i opis njegove funkcionalnosti kako bi ga agent znao pravilno koristiti. Nakon što alat zaprimi upit od agenta, šalje ga putem RabbitMQ-a servisu za generiranje slika. Ovaj servis generira sliku pomoću SDXL Lightning modela te ju šalje nazad agentu. Za generirane slike stvara se ID koji se pohranjuje u bazu zajedno s odgovarajućom porukom. Generirane slike spremaju se na poslužitelju pod nazivom jednakim njihovom ID-u radi lakšeg dohvaćanja. Kao i LLM, model za generiranje slika konfiguriran je da radi na GPU-u pomoću CUDA-e. Slika 8. prikazuje primjer slike generirane pomoću sustava Lumen i alata za generiranje slika na temelju upita „Generate an image of a forest in winter“. Slike koje agent generira su rezolucije 1024 x 1024 piksela. Vidljivo je da je slika visokokvalitetna i realistična, pogotovo uzevši u obzir da je generirana u samo dva koraka rekonstrukcije.



Slika 8. Primjer slike generirane pomoću sustava Lumen.

#### 3.1.5.4. Mailjet API

Posljednji alat dostupan agentu je alat za slanje elektroničke pošte. Za ovu funkcionalnost ne postoji ugrađeni alat, već je ručno izrađen alat koji radi pomoću Mailjet API-ja. Mailjet je online platforma koja dopušta automatizirano slanje elektroničke pošte. Za uspješno korištenje ove usluge, kao i za objavljivanje sustava opisano u kasnijim poglavljima, bilo je potrebno zakupit domenu. Zakupljena je domena bojan-radulovic.xyz putem Namecheap platforme. Domena je povezana s Mailjet-om dodavanjem posebnih DNS zapisa za domenu unutar

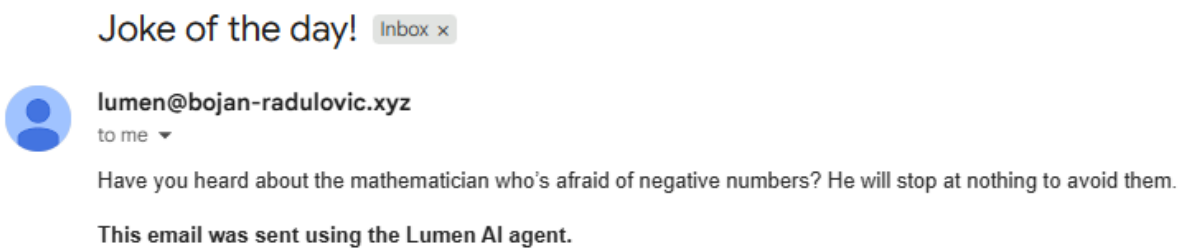
Namecheap web aplikacije. Nakon ovih koraka, pomoću Mailjet-a nam je omogućeno slanje elektronske pošte s bilo koje adrese koja pripada bojan-radulovic.xyz domeni. Za poštu koju šalje sustav Lumen odabrana adresa elektroničke pošte je lumen@bojan-radulovic.xyz. Sustav je povezan s Mailjet-om pomoću API ključa. Mailjet pruža besplatni plan koji dopušta slanje 200 elektroničkih pošta dnevno, što je sasvim dovoljno za potrebe implementacije i testiranja sustava Lumen.

Alat za slanje elektroničke pošte od agenta očekuje adresu primatelja te naslov i sadržaj pošte. Na temelju ovih podataka, alat konstruira elektroničku poštu te ju šalje primatelju pomoću Mailjet API-ja. Agent kao opservaciju dobiva informaciju o tome je li elektronička pošta uspješno poslana. Na Slici 9. prikazan je isječak koda koji prikazuje proces slanja elektroničke pošte. Na temelju informacija koje agent prosljeđuje alatu, ispunjavaju se potrebni podaci za slanje pošte. Elektronička pošta se šalje, a agenta se obavještava o uspješnosti slanja.

```
data = {
    'Messages': [
        {
            "From": {
                "Email": "lumen@bojan-radulovic.xyz",
                "Name": "Lumen"
            },
            "To": [
                {
                    "Email": to_email,
                    "Name": "Recipient"
                }
            ],
            "Subject": subject,
            "HTMLPart": html_content + "<p><strong>This email was sent using the Lumen AI agent.</strong></p>"
        }
    ]
}
try:
    result = mailjet.send.create(data=data)
    if result.status_code == 200:
        return f"Email sent! Status Code: {result.status_code}."
    else:
        return f"Failed to send email. Status Code: {result.status_code}."
except Exception as e:
    return f"Error: {e}."
```

Slika 9. Isječak koda koji prikazuje proces slanja elektroničke pošte.

Slika 10. prikazuje primjer elektroničke pošte poslane kroz sustav Lumen od strane agenta. Korisnički upit koji je rezultirao slanjem ove elektroničke pošte bio je „Send a joke to bojanradulovic25@gmail.com“.




Slika 10. Primjer elektroničke pošte poslane kroz sustav Lumen.

Prilikom implementacije ovog alata, kroz zanimljivu opasku, dodatno je naglašena važnost alata otvorenog koda, lokalnog pokretanja i smanjenja ovisnosti o alatima treće strane. Naime, u ranijim verzijama sustava Lumen, umjesto Mailjet platfore korištena je slična platforma Sendgrid. Sendgrid je dugi niz godina pružao besplatan plan koji je bio adekvatan za potrebe sustava Lumen. Međutim, počevši od 27. svibnja 2025. godine, Sendgrid ukida besplatni plan. Zbog ovoga, bilo je potrebno pronaći alternativnu platformu te je odabrana platforma Mailjet. Ova situacija pokazuje primjer jednog od glavnih nedostataka korištenja rješenja treće strane, a to je ovisnost o njihovim politikama, oblicima naplate, uvjetima korištenja itd. koji su skloni promjenama. Idealno rješenje bilo bi pokretanje i održavanje vlastitog poslužitelja za slanje elektroničke pošte (engl. *Simple Mail Transfer Protocol server*, SMTP server), ali ovo zahtjeva značajnu količinu truda i resursa te je izvan opsega ovog diplomskog rada.

### 3.1.6. TTS i STT funkcionalnosti

Jedna od važnih značajki sustava Lumen mogućnost je komunikacije s agentom prirodnim jezikom. Kako bi se komunikacija učinila još lakšom te sustav pristupačnijim, implementirane su TTS i STT funkcionalnosti.

TTS funkcionalnost implementirana je pomoću Kokoro TTS platforme. Kokoro je softver otvorenog koda razvijen za generiranje govora temeljenog na tekstu. Za integraciju ovog modela u sustav Lumen, korištena je modificirana verzija GitHub repozitorija otvorenog koda PierrunoYT/Kokoro-TTS-Local koji pruža izbor od 54 različitih glasova kroz 8 jezika, lakše lokalno pokretanje modela, jednostavno web sučelje korišteno pri testiranju TTS funkcionalnosti itd. [26]. Kod generiranja glasa unutar sustava Lumen, odabran je engleski glas naziva „bf_isabella“. U budućnosti, kao dodatna funkcionalnost sustava, mogla bi se implementirati i mogućnost da korisnici biraju glas koji žele unutar korisničkog sučelja sustava.

TTS dio sustava pokrenut je u obliku zasebnog servisa koji komunicira s FastAPI poslužiteljem putem RabbitMQ-a. Kada agent završi zadatak i generira završni odgovor, tekst završnog odgovara se preko RabbitMQ-a šalje TTS servisu. TTS servis na temelju završnog odgovora generira glas te ga šalje nazad FastAPI poslužitelju. FastAPI poslužitelj na posljetku šalje tekstualni i glasovni odgovor na korisnikov upit React klijentu.

STT funkcionalnost implementirana je pomoću Web Speech API-ja. Web Speech API je web standard koji aplikacijama u pregledniku omogućuje prepoznavanje govora. Kada korisnik sustava Lumen odabere funkcionalnost prepoznavanja glasa, nakon što dopusti korištenje mikrofona, sustav započinje slušati korisnikov govor. Ako govor nije prepoznat, prepoznavanje glasa se obustavlja. Nakon što korisnik završi govoriti ili ručno prekine proces prepoznavanja glasa, prepoznati glas se u tekstualnom obliku šalje FastAPI poslužitelju u obliku korisnikovog upita, na isti način kao i kada korisnik ručno utipka upit.

### 3.1.7. RabbitMQ

Komunikacija među različitim servisima unutar sustava Lumen postignuta je koristeći RabbitMQ posrednika za poruke. RabbitMQ je softver otvorenog koda za razmjenu poruka.

On omogućava servisima da međusobno komuniciraju putem poruka na skalabilan i pouzdan način. Unutar sustava Lumen, RabbitMQ je pokrenut u obliku Docker kontejnera na kojeg se servisi spajaju. Kod sustava Lumen, RabbitMQ je korišten kako bi FastAPI poslužitelj mogao komunicirati sa servisom za generiranje slika i TTS servisom. Prilikom generiranja slika, FastAPI servis šalje zahtjev servisu za generiranje slike na temelju opisa te, nakon obrade zahtjeva, kao odgovor dobiva generiranu sliku. Sličan proces odvija se i kod TTS servisa, gdje FastAPI šalje zahtjev za generiranje glasa na temelju teksta te kao odgovor dobiva generirani glas.

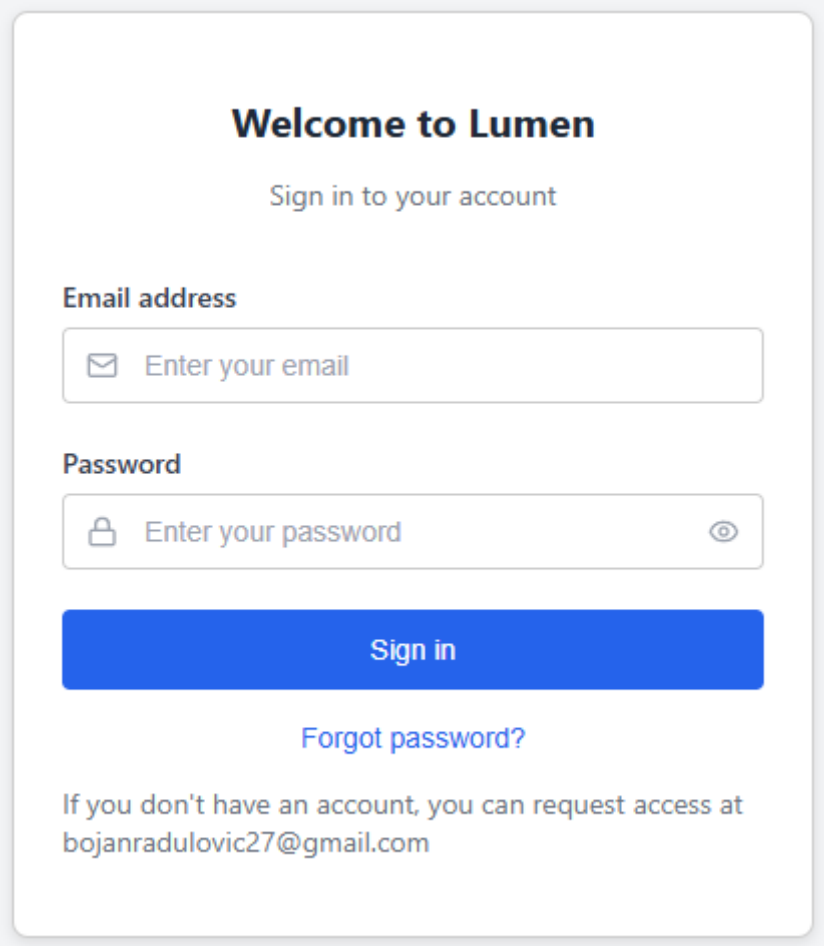
Ovaj pristup olakšava proces skaliranja sustava. Na primjer ako se primijeti velik broj korisničkih upita koji uključuju generiranje slika, kako bi se ubrzao proces obrade takvih zahtjeva, moguće je pokretanje više instanci servisa za generiranje slika što bi dopustilo paralelno izvođenje više zahtjeva za generiranje slika. Ovo bi ubrzalo rad sustava, ali zahtjeva dodatne resurse. Osim toga, u slučaju kvara ili pada nekog od servisa, ostatak sustava može normalno nastaviti raditi bez te funkcionalnosti. Na primjer, u slučaju pada alata za generiranje slika, kod korisničkih upita koji zahtijevaju generiranje slika agent bi prepoznao na temelju opservacija da alat za generiranjem slika ne radi pravilno te bi o tome obavijestio korisnike. Ostali upiti koji ne zahtijevaju generiranje slika obrađivali bi se normalno.

## 3.2. Implementacija korisničkog sučelja

Drugi dio sustava Lumen predstavlja korisničko sučelje izrađeno pomoću Reacta. React je JavaScript biblioteka za izradu korisničkih sučelja. Razvija ju i održava tvrtka Meta te predstavlja jedan od najpopularnijih alata za razvoj korisničkih sučelja. Odabrana je za potrebe sustava Lumen zbog svoje jednostavnosti, fleksibilnosti i ogromnog ekosustava. Sučelje je u potpunosti responzivno te prilagođeno i za stolna računala i za mobilne uređaje.

### 3.2.1. Sučelje za prijavu

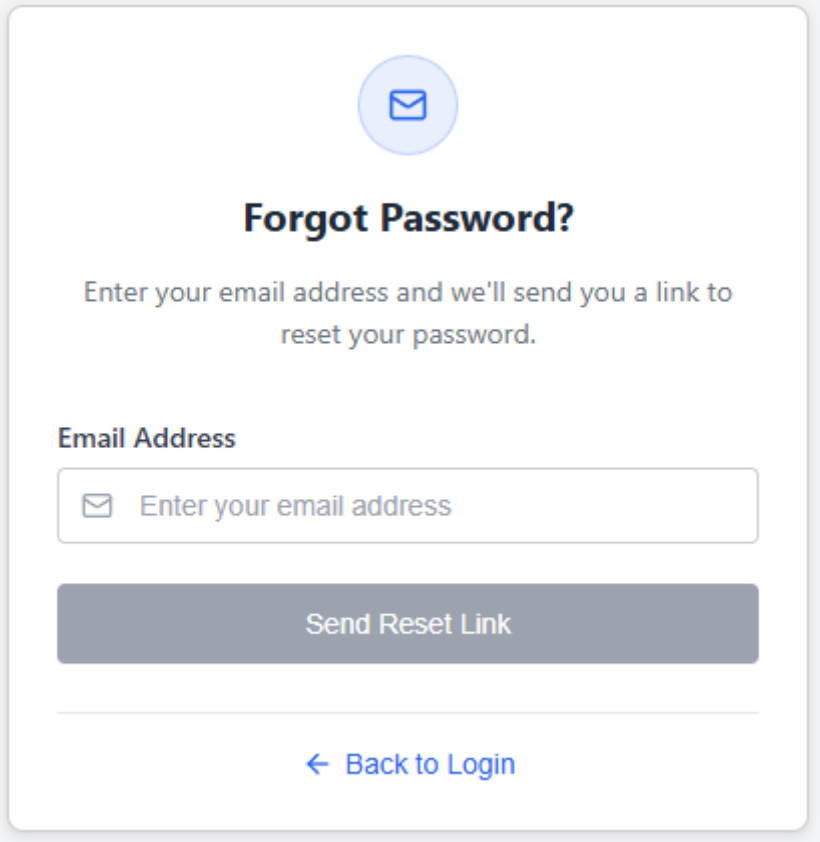
Prva stranica sučelja koja se pokazuje korisnicima prilikom pristupanja sustavu je stranica za prijavu. Na Slici 11. prikazana je stranica za prijavu korisnika. Ovdje se korisnici mogu prijaviti u sustav nakon što upišu svoju adresu elektroničke pošte i lozinku. Polje za upis lozinke ne prikazuje upisane znakove, osim ako korisnik ne klikne gumb u obliku oka. Ova stranica također sadrži gumb koji vodi na stranicu za oporavak lozinke.




Slika 11. Stranica za prijavu korisnika.

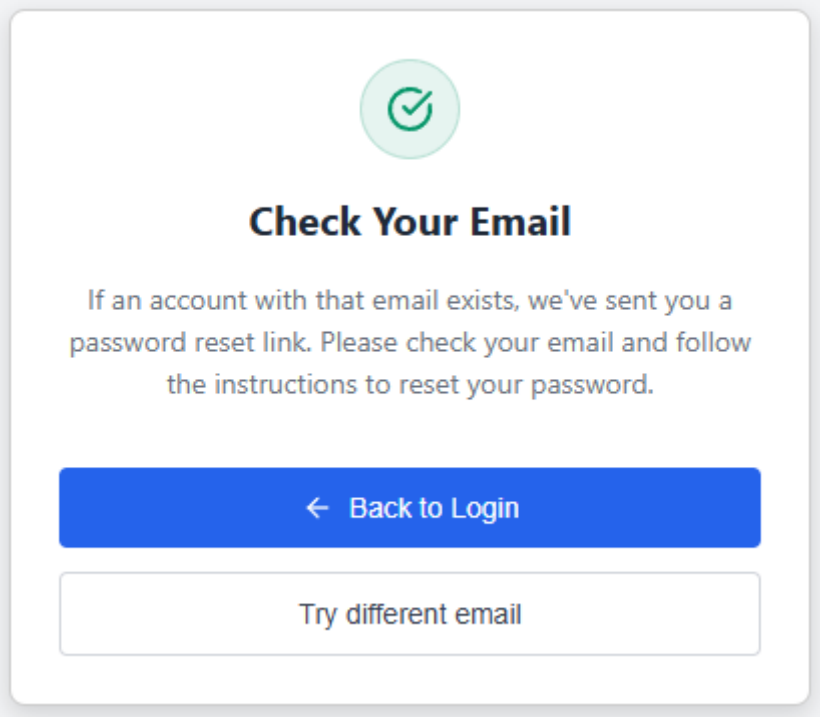
### 3.2.2. Oporavak lozinke

U slučaju da korisnik zaboravi lozinku svojeg Lumen korisničkog računa, može zatražiti oporavak lozinke. Slika 12. prikazuje stranicu za oporavak lozinke. Na ovoj stranici korisnik može zatražiti oporavak zaboravljene lozinke upisom svoje adrese elektroničke pošte. Nakon uspješno zatraženog oporavka lozinke, korisniku se prikazuje stranica koja potvrđuje uspješnost zatraživanja oporavka lozinke.




Slika 12. Stranica za oporavak lozinke.

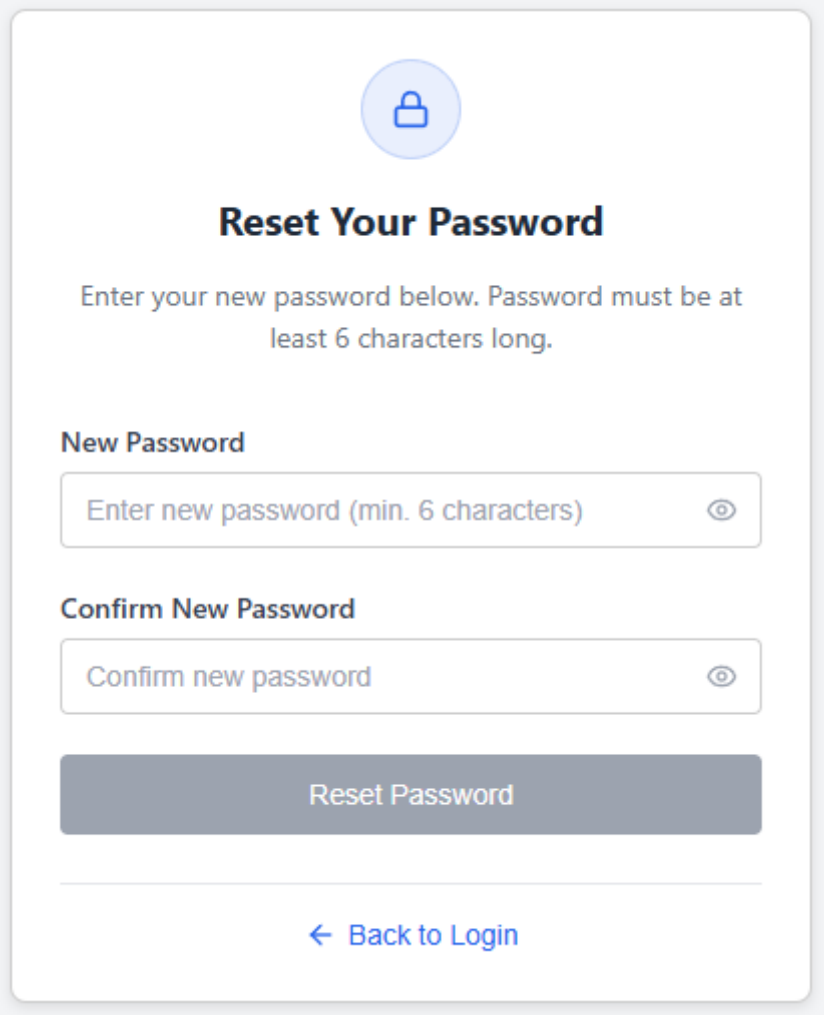
Na Slici 13. prikazana je stranica koja potvrđuje slanje zahtjeva za oporavak lozinke. Kako korisnici ne bi mogli znati postoji li u sustavu Lumen korisnički račun za određenu adresu elektroničke pošte, ova stranica se korisnicima prikazuje čak i kada korisnički račun s navedenom adresom ne postoji.




Slika 13. Stranica koja potvrđuje slanje zahtjeva za oporavak lozinke.

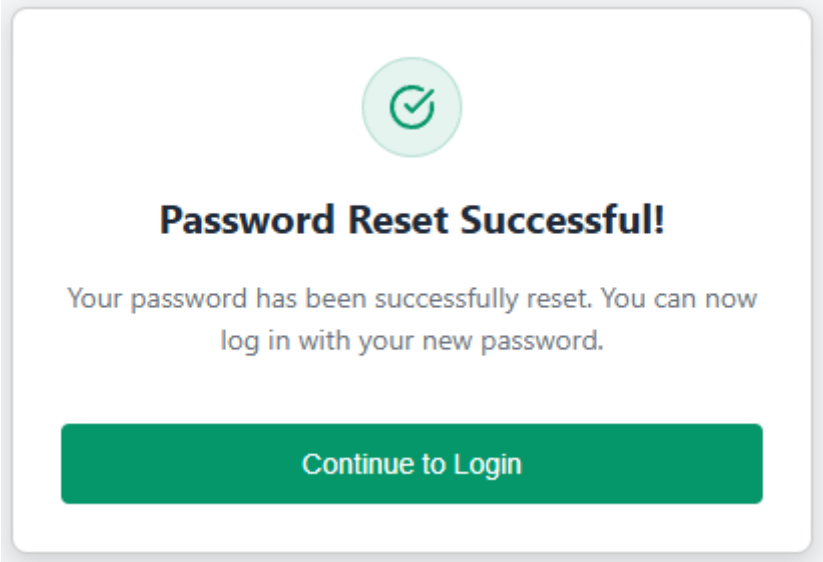
Nakon uspješno zatraženog oporavka lozinke, na korisnikovu adresu elektroničke pošte šalje se pošta s uputama za oporavak lozinke. Primjer takve pošte vidljiv je na Slici 4. Klikom na poveznicu iz pošte, korisniku se prikazuje stranica za promjenu lozinke.

Slika 14. prikazuje stranicu za promjenu lozinke. Nakon što korisnik odabere novu lozinku, ona se mijenja u bazi podataka sustava te se korisniku prikazuje poruka o uspješnoj promjeni lozinke.




Slika 14. Stranica za promjenu lozinke.

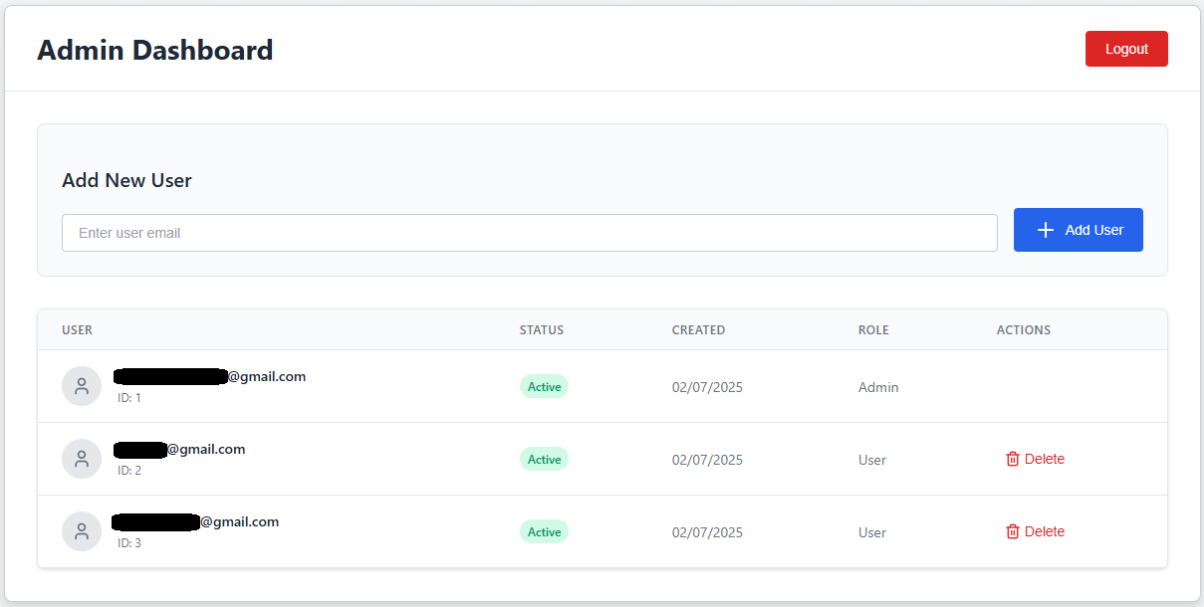
Slika 15. prikazuje poruku o uspješnoj promjeni lozinke koja se javlja korisniku nakon što promjeni lozinku. Klikom na gumb, korisnik se vraća na stranicu za prijavu gdje se može prijaviti sa svojom novom lozinkom.




Slika 15. Poruka o uspješnoj promjeni lozinke.

### 3.2.3. Administratorsko sučelje

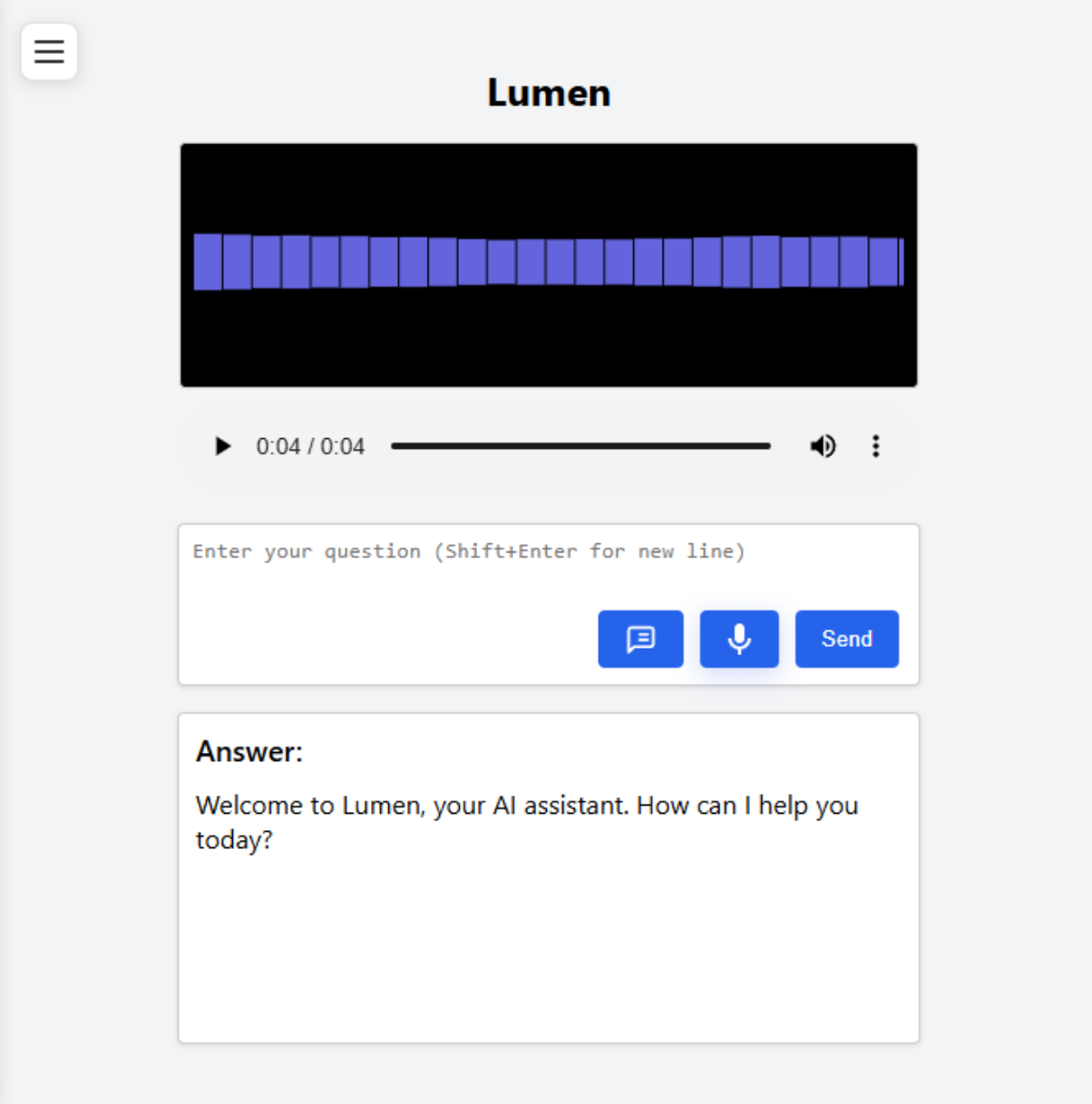
Prilikom prijave korisnika, u slučaju da korisnik ima ulogu administratora, umjesto prikazivanja glavne stranice sučelja prikazuje se stranica za administratore. Na Slici 16. prikazano je administratorsko sučelje. Na ovoj stranici, administratori sustava mogu pregledati podatke o korisnicima, uključujući njihov status, datum stvaranja korisničkog računa i rolu. Status predstavlja je li korisnik prihvatio poziv za pridruživanje sustavu te promijenio svoju inicijalnu lozinku. Rola se odnosi na ulogu korisnika, donosno je li riječ o običnom korisniku sustava ili administratoru. Administratori na ovoj stranici također mogu ukloniti korisnički račun te dodati nove korisnike. Dodavanjem novog korisnika, na adresu elektroničke pošte korisnika dostavlja se pošta vidljiva na Slici 5. Pri vrhu stranice nalazi se i gumb koji služi za odjavu administratora iz sustava.




Slika 16. Administratorsko sučelje.

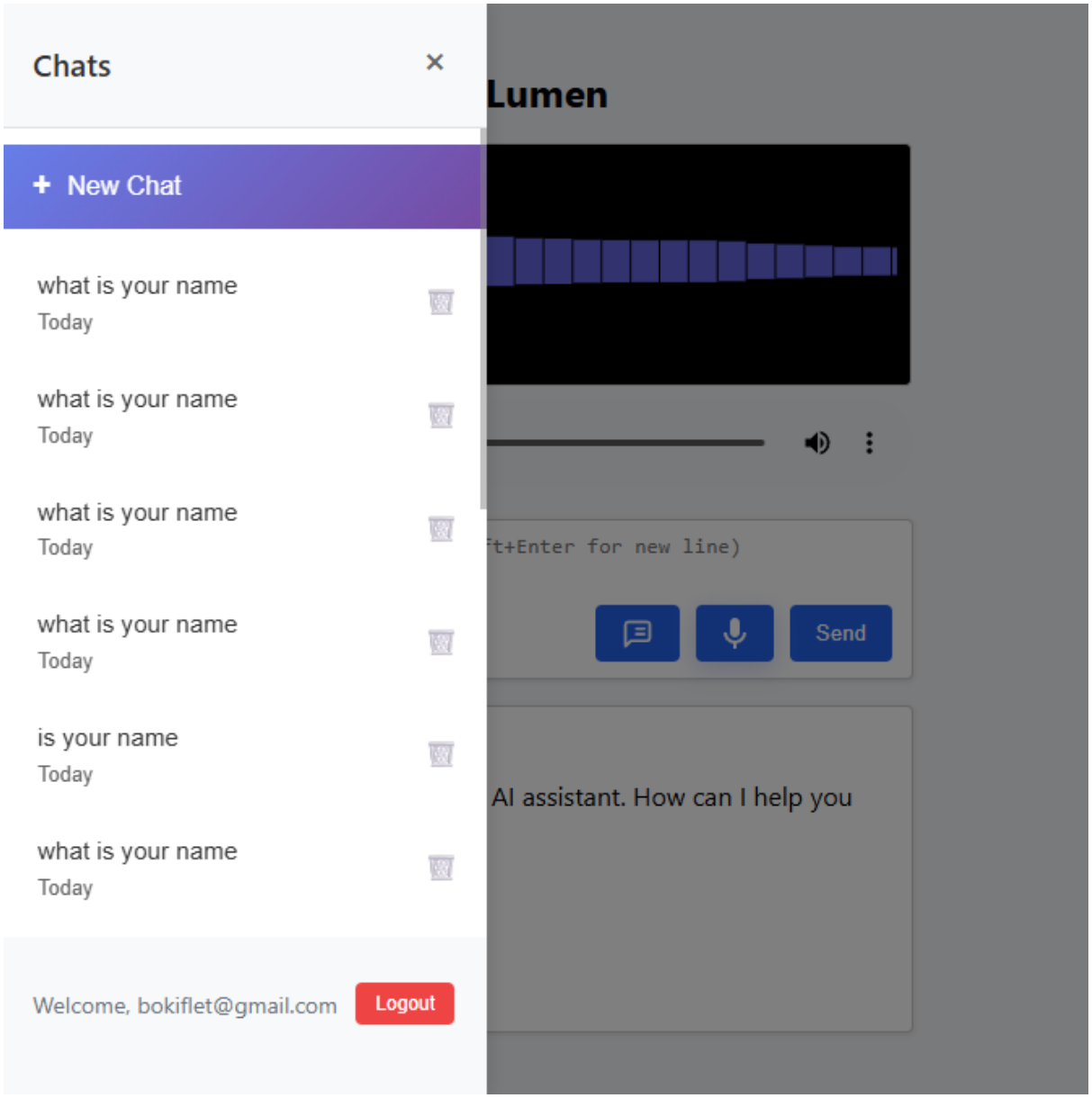
### 3.2.4. Sučelje za razgovor s agentom

Temeljna stranica korisničkog sučelja sustava Lumen, koja se pojavljuje korisnicima nakon prijave u sustav, je stranica za razgovor s Lumen AI agentom. Na Slici 17. vidljiva je stranica za razgovor s agentom. Ovdje korisnici glasovno ili tekstualno postavljaju upite agentu. Nakon obrade korisnikovog upita, korisniku se pri dnu stranice prikazuje tekstualni odgovor te započinje i glasovno čitanje odgovora. U gornjem desnom kutu stranice nalazi se gumb koji otvara burger meni s popisom korisnikovih razgovora s agentom te gumbom za odjavu. Na vrhu stranice nalazi se audio vizualizator za glasovne odgovore agenta. Audio vizualizator se sastoji od nekoliko desetaka stupaca koji se povećavaju ili smanjuju ovisno o glasnoći audio zapisa. Kada nije pokrenuta reprodukcija niti jednog audio zapisa, stupci vizualizatora se nasumično smanjuju i povećavaju. Ispod vizualizatora nalaze se opcije za audio reprodukciju. Ovdje korisnik može pokrenuti ili zaustaviti reprodukciju audio zapisa, odabrati određen trenutak audio zapisa, pojačati ili smanjiti jačinu zvuka reprodukcije audio zapisa, preuzeti audio zapis te promijeniti brzinu reprodukcije zapisa. Ispod toga nalazi se forma za upis korisnikovog upita. Ovdje se nalaze i gumbi za promjenu prikaza stranice u alternativni, više tradicionalni, oblik sučelja za razgovor, pokretanje glasovnog postavljanja pitanja i slanje upita. Naposljetku, na dnu stranice, nalazi se područje za prikazivanje odgovora agenta na upite.



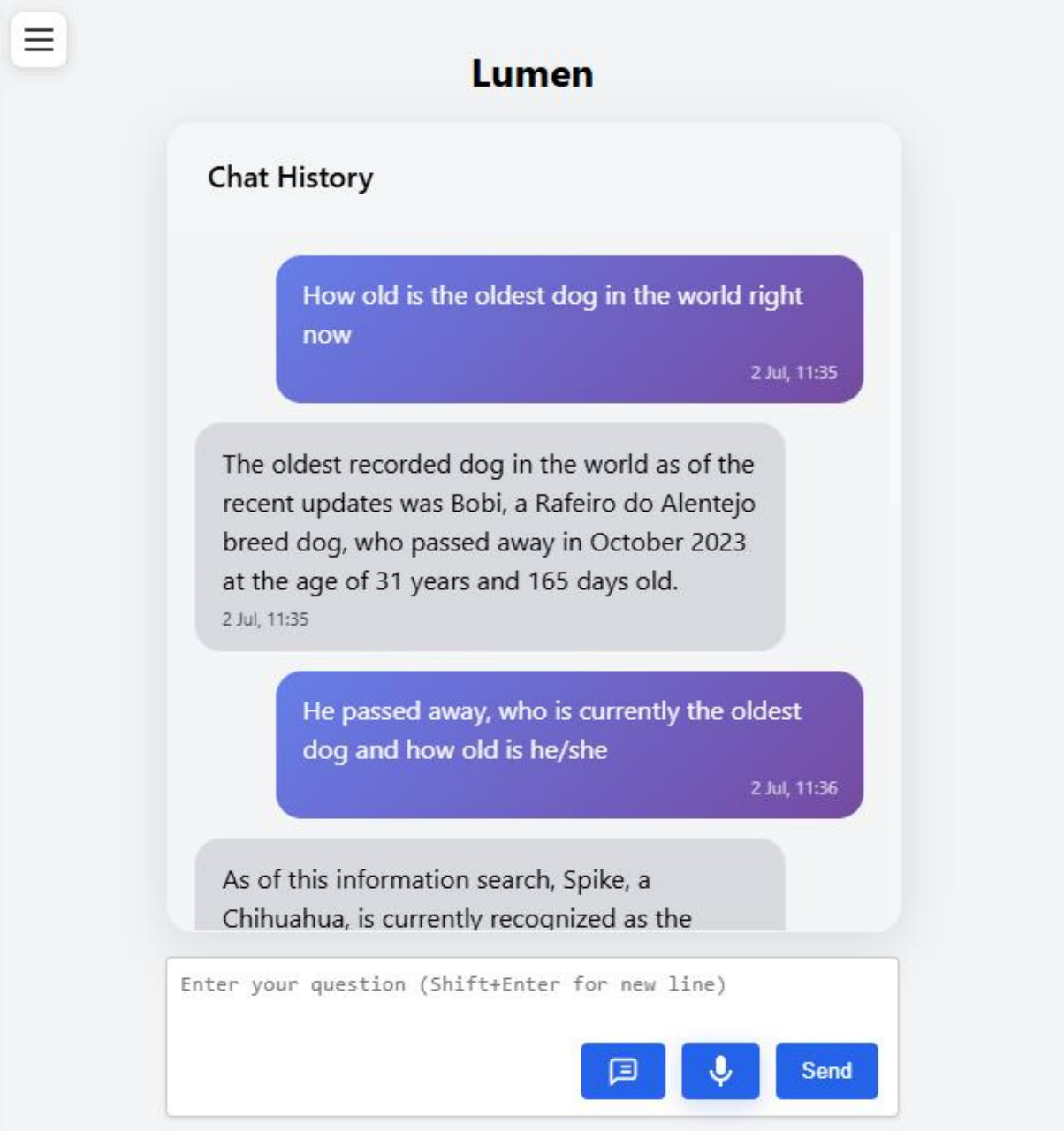

Slika 17. Stranica za razgovor s Lumen agentom.

Na Slici 18. prikazan je izgled burger menija s podacima o razgovorima korisnika. Na ovom meniju se nalazi popis svih razgovora koje je korisnik imao s agentom. Za svaki razgovor, moguće je vidjeti njegov naslov i datum stvaranja. Svaki razgovor je također moguće obrisati. Klikom na bilo koji razgovor, u sučelju sustava otvara se taj razgovor. Pri vrhu menija u gornjem desnom kutu nalazi se gumb za zatvaranje menija. Klikom na „New Chat“ gumb korisnik može započeti novi razgovor s agentom. Na dnu menija prikazuje se trenutno prijavljeni korisnik te gumb za odjavu.



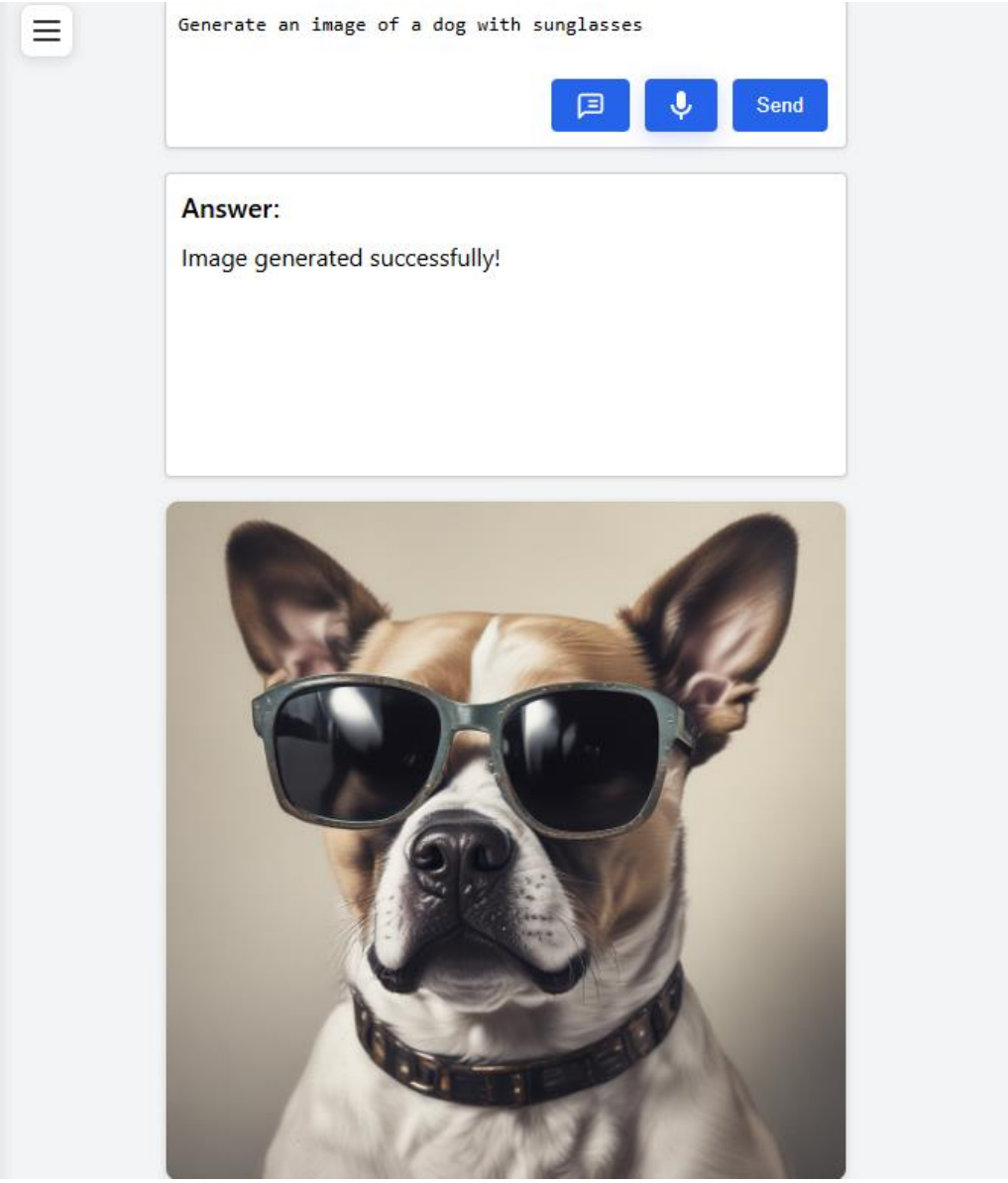

Slika 18. Stranica za razgovor s otvorenim burger menijem.

Slika 19. prikazuje alternativni oblik stranice za razgovor s agentom. Ova varijacija stranice dopušta prikaz svih poruka u razgovor i njihovog vremena slanja, zajedno s bilo kojim generiranim slikama. Kada je stranica u ovom obliku audio vizualizator je sakriven.



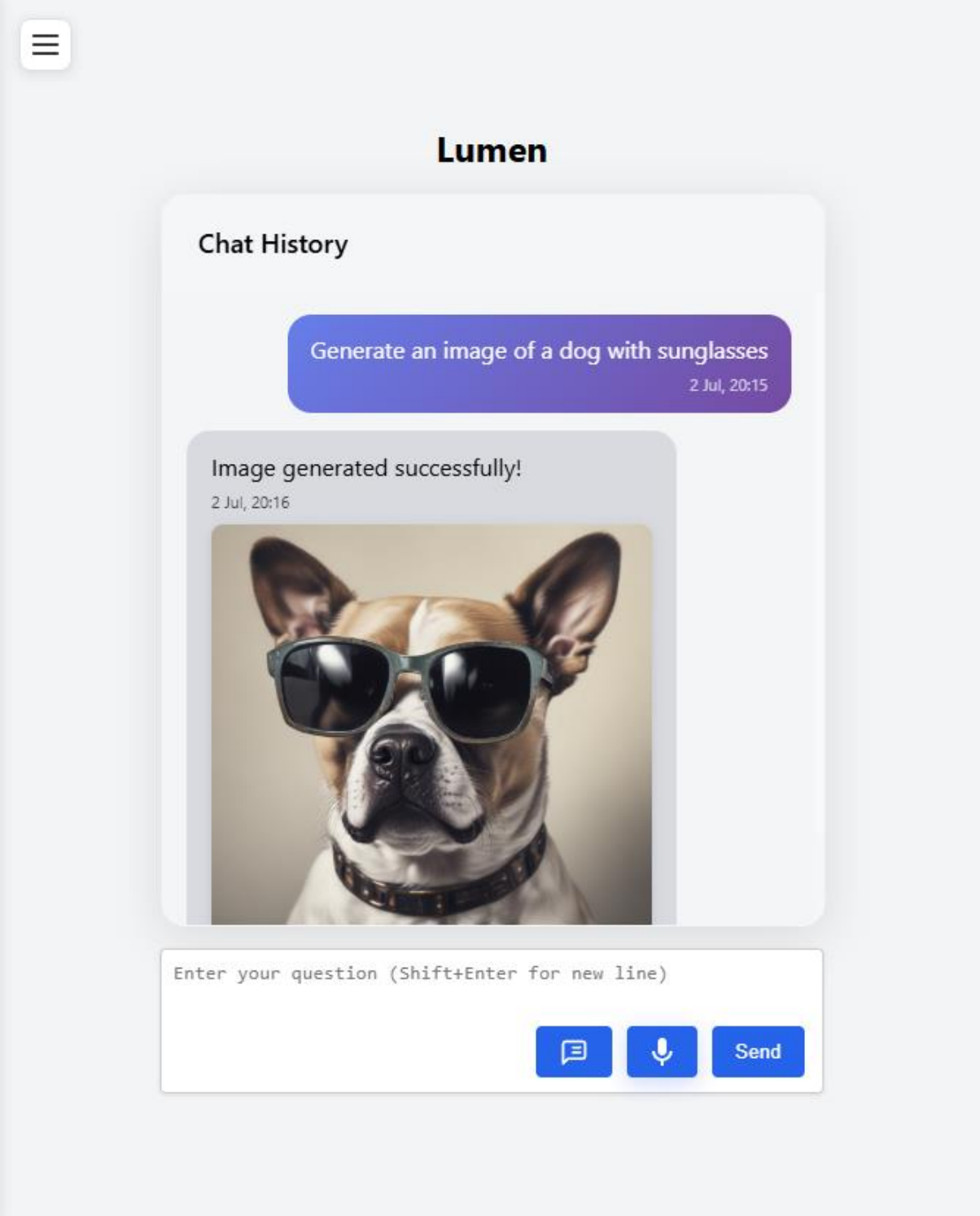

Slika 19. Alternativni prikaz stranice za razgovor s agentom.

Slika 20. prikazuje način na koji se prikazuju generirane slike. Generirane slike prikazuju se na dnu stranice za razgovor s agentom, ispod područja za tekstualne odgovore.




Slika 20. Prikazivanje generiranih slika.

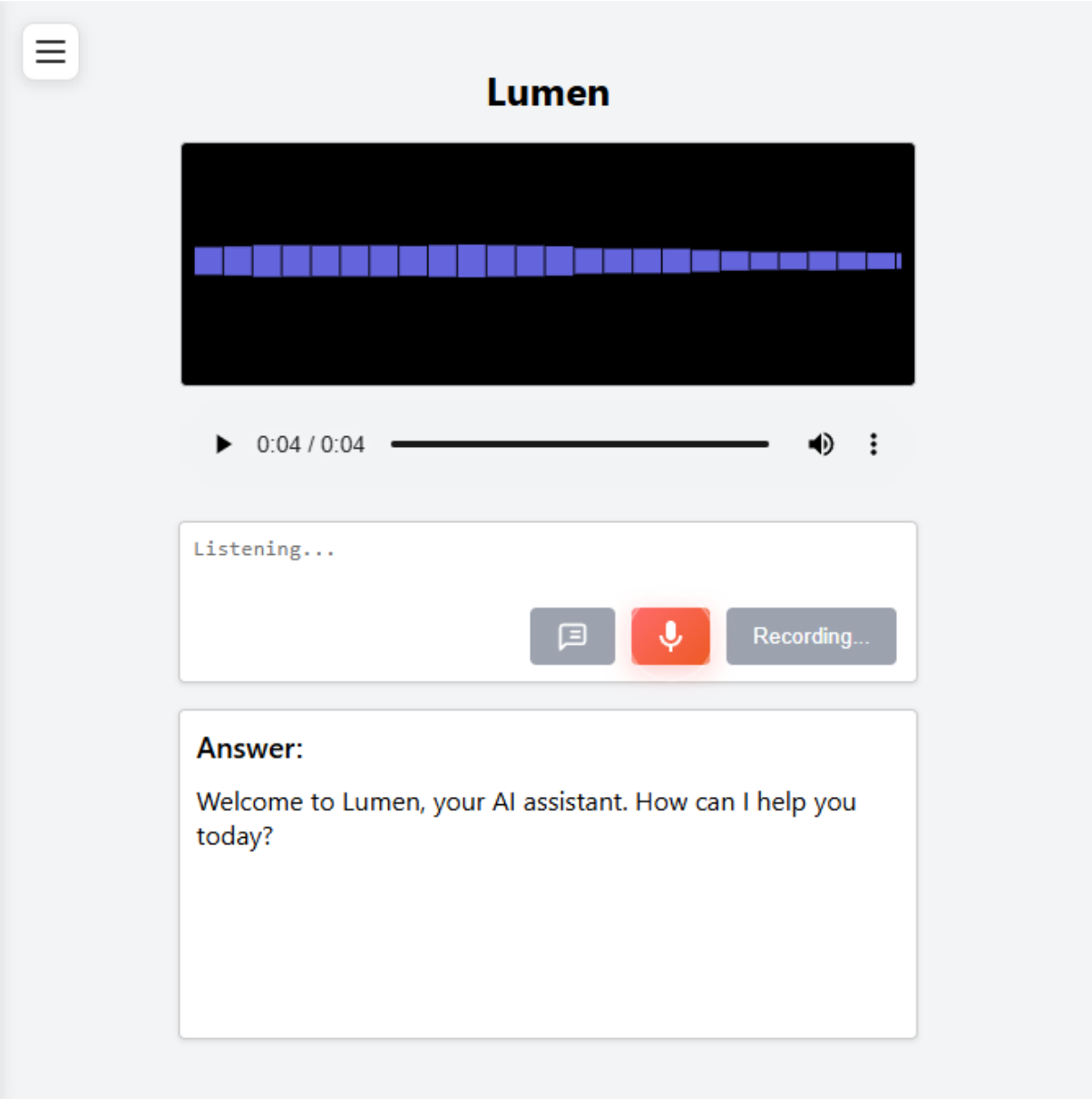
Kod alternativnog prikaza stranice za razgovor s agentom, generirane slike se prikazuju unutar samih poruka agenta, kao što je prikazano na Slici 21.




Slika 21. Prikazivanje generiranih slika kod alternativnog prikaza stranice.

Kada korisnik klikne na gumb za početak snimanja glasa, sustav počinje snimanje pomoću korisnikovog mikrofona. Prilikom snimanja, ostali gumbi na stranici su onemogućeni. Gumb za snimanje poprima crvenu boju koja predstavlja početak snimanja, vidljivo na Slici 22. Kada snimanje završi, ili tako da korisnik prestane govoriti ili ponovnim klikom na gumb za snimanje, tekst prepoznat iz korisnikovog govora šalje se kao upit poslužiteljskoj strani

sustava. Prilikom prvog korištenja ove funkcionalnosti, od korisnika se traži dopuštenje za korištenje mikrofona.




Slika 22. Sučelje prilikom snimanja glasa.

## 3.3. Objavljivanje sustava

Kako bi sustav Lumen bilo javno dostupna, a kako bi se nastavila ideja što manjeg oslanjanje na rješenja treće strane, sustav je objavljena na vlastitom Apache serveru. Apache je besplatan softver otvorenog koda za izradu web poslužitelja. Apache omogućuje poslužiteljima isporučivanje web sadržaja putem HTTP i HTTPS internetskih protokola. Apache je odabran zbog velike online zajednice i ekosustava te svoje lakoće korištenja. Osim toga, Apache pruža veliku razinu kontrole, modularnosti i fleksibilnosti prilikom konfiguracije.

Prvi korak objavljivanja sustava bio je zakup internetske domene. Zakupljena je domena bojan-radulovic.xyz putem Namecheap platforme.

Drugi korak bio je usmjeravanje domene na računalo na kojem se nalazi sustav Lumen. Kako bi se ovo postiglo, prvo je bilo potrebno, u mrežnim postavkama računala, lokalnu IP adresu postaviti kao statičnu. Zatim je bilo potrebno, u postavkama internetskog rutera na koji je spojeno računalo, postaviti pravila prosljeđivanja portova. Postavljena su pravila da se zahtjevi koji pristižu ruteru na portove 80 i 443 proslijede na statičnu lokalnu IP adresu računala na portove 80 i 443. Zadnji problem koji je trebalo riješiti je to da javna IP adresa rutera nije statična. Srećom, Namecheap nudi opciju dinamičnog DNS-a. S Namecheap stranice, na računalo na kojem se nalazi sustav Lumen, preuzet je softver koji periodički provjerava javnu IP adresu te po potrebi ažurira DNS zapise kako bi bojan-radulovic.xyz domena uvijek pokazivala na IP adresu na kojoj je dostupno računalo.

Zatim je na računalu instaliran Apache web server. U postavkama servera dodana su pravila obrnutog proxy-ja koja zahtjeve na bojan-radulovic.xyz/lumen preusmjeravaju na port 3000 računala na kojem je pokrenut React klijent. Također, dodan je i obrnuti proxy za bojan-radulovic.xyz/lumen-backend koji pokazuje na port 8000 računala na kojemu je pokrenut FastAPI poslužitelj. Nakon ovog podešavanja, korisnici se na sustav mogu povezati putem poveznice bojan-radulovic.xyz/lumen, a kada React klijent šalje zahtjeve poslužitelju, šalje ih na bojan-radulovic.xyz/lumen-backend.

Zadnji korak objavljivanja sustava Lumen bio je dobivanje SSL certifikata. SSL certifikat omogućuje korištenje HTTPS protokola te pruža dodatnu sigurnost korisnicima. Ovo ne samo da je poželjno, već je i nužno za određene funkcionalnosti sustava. Na primjer, kako bi sustav mogao zatražiti korištenje mikrofona, što je nužno za pravilan rad STT funkcionalnosti, potrebna je sigurna veza putem HTTPS protokola. SSL certifikat dobiven je pomoću Certbot-a i Let's Encrypt autoriteta. Let's Encrypt je besplatni certifikacijski autoritet koji izdaje SSL i TLS certifikate. Njegov glavni cilj je povećati dostupnost certifikata za web servere te napraviti proces dobivanja certifikata što lakšim i jednostavnijim moguće. Certbot je besplatni softverski alat koji automatizira proces dobivanja i obnavljanja certifikata. Ovaj alat automatski preuzima, instalira i obnavlja certifikate. Koristeći ovo softversko rješenje, dobiven je i instaliran SSL certifikat te omogućeno korištenje HTTPS protokola.

Ovakav pristup objavljivanju sustava Lumen demonstrira kako je moguće osigurati punu funkcionalnost sustava koristeći vlastitu infrastrukturu, bez oslanjanja na servise treće strane. Ovime se zadržava veća kontrola nad sustavom te potencijalno smanjuju dugoročni troškovi zakupa web servera.

## 3.4. Mogućnost unaprjeđenja sustava

Iako su ovim diplomskim radom pokazane brojne prednosti koje pruža lokalno pokretanje softverskih rješenja otvorenog koda, postoje i određene mane ovog pristupe. Također, kod samog sustava Lumen isto postoji prostor za daljnji razvoj.

Jedno od područja potencijalnog unaprjeđenja sustava leži u pokretanju vlastitog poslužitelja. Otvaranje vlastitog računala internetu, u obliku web poslužitelja, sa sobom nosi rizike vezane uz razne metode kibernetičkih napada. Ovaj problem bi se mogao umanjiti virtualizacijom na računalu te postavljanjem rigoroznijih sigurnosnih pravila. Osim toga, u slučaju gubitka napajanja računala ili prekida internetske veze, sustav postaje nedostupna. Kako bi se ovo izbjeglo, može se koristiti uređaj za neprekidno napajanje, redundantna internetska veza i/ili repliciranje poslužitelja na više računala na više lokacija. Nadalje, pošto postoji samo jedan poslužitelj koji se nalazi na samo jednoj lokaciji, za obradu zahtjeva iz udaljenijih dijelova svijeta bilo bi potrebno više vremena zbog povećane latencije.

Drugo veliko područje unaprjeđenja sustava su hardverska ograničenja. Korištenje većih i jačih AI modela poboljšalo bi rad sustava Lumen, ali bi zahtijevalo puno veće hardverske resurse. Osim toga, veći LLM-ovi često podržavaju više jezika pa bi korištenjem takvih modela sustav Lumen potencijalno mogao biti višejezičan, umjesto da podržava samo engleski jezik.

Također, zbog hardverskih i vremenskih ograničenja, za određene dijelove sustava Lumen korištena su rješenja treće strane. Primjerice, za STT funkcionalnost bi se također mogli koristiti AI modeli otvorenog koda, ali to zahtjeva dodatne hardverske resurse. Mailjet API bi se također mogao zamijeniti korištenjem vlastitog SMTP poslužitelja, ali postavljanje i održavanje takvog poslužitelja zahtjeva značajan trud, tehničko znanje i hardver na kojem bi poslužitelj boravio.

Moguće je u sustav Lumen implementirati i dodatne funkcionalnosti te agentu dati pristup većem broju alata. Na primjer, korisnicima bi se moglo omogućiti objavljivanje datoteka (npr. PDF ili Word datoteka) na temelju kojih bi agent mogao pružati odgovore koristeći RAG metode. Ova funkcionalnost nije implementirana jer povećava kontekst razgovora te zato što s povećanim brojem alata agentu postaje sve teže točno odabrati kojeg koristiti. Oba problema bi se mogla riješiti korištenjem većeg modela s većim kontekstom.

## 3.5. Specifikacije sustava

Cijeli sustav Lumen pokrenut je na jednom računalu. Računalo je opremljeno procesorom Intel Core i5 12400F te matičnom pločom Gigabyte Technology Co. Ltd. H610M H V3 DDR4. Ugrađeno je 64 GB DDR4 radne memorije s radnim taktom od 3200 MHz. Računalo sadrži i grafičku kartica NVIDIA GeForce RTX 4070 SUPER s 12 GB video memorije. Na softverskoj razini, korišten je operacijski sustav Windows 11, dok je sustav Lumen razvijen i pokrenut u programskom jeziku Python verzije 3.9.13. Za komunikaciju između komponenti korišten je RabbitMQ verzije 4.0, a kao poslužitelj je korišten Apache verzije 2.4.63. Dijelovi sustava kontejnerizirani su pomoću Docker verzije 28.0.1.

# 4. Zaključak

U ovom diplomskom radu uspješno je pokazana vrijednost korištenja lokalno pokrenutih rješenja otvorenog koda na primjeru izrade agentnog sustava temeljenog na umjetnoj inteligenciji. Razvijen je sustav Lumen čija je temeljna komponenta AI agent implementiran pomoću velikih jezičnih modela otvorenog koda. Objašnjene su funkcionalnosti agenta te alati koji su mu dostupni, uključujući alate za generiranje slika, pretraživanje interneta, pokretanje Python kôda i slanje elektroničke pošte. Opisana je i implementacija poslužiteljske strane sustava koristeći tehnologije poput FastAPI-ja i RabbitMQ-a te sučelja izrađenog pomoću Reacta.

Prikazan je i proces objavljivanja sustava u obliku web aplikacije na vlastitom Apache poslužitelju. Objašnjen je proces zakupa internetske domene, konfiguracije DNS zapisa, dobivanja SSL certifikata, konfiguracije Apache poslužitelja itd.

Kroz cijeli rad predstavljene su pozitivne i negativne strane lokalnog pokretanja i korištenja alata otvorenog koda naspram korištenja servisa treće strane. Lokalno pokretanje softverskih rješenja otvorenog koda može imati ogromne benefite u obliku povećane privatnosti, fleksibilnosti te kontrole nad sustavom. Korištenje lokalne infrastrukture izbjegava ovisnost o rješenjima treće strane, ali ujedno zahtjeva i veću razinu tehničkog znanja i hardverskih resursa.

Na kraju, rad pokazuje da je izgradnja složenih agentnih AI sustava moguća i bez oslanjanja na komercijalne platforme, što otvara vrata razvoju otvorenih i neovisnih softverskih rješenja u području umjetne inteligencije. Time se potiče prilagodljivost i transparentnost u razvoju softverskih rješenja, posebice u okruženjima gdje je privatnost i sigurnost ključna. U konačnici, rad doprinosi širem razumijevanju potencijala otvorenog koda i lokalno pokrenutih AI sustava.

# Literatura


[1] K. Zhu, „Ranked: Most Popular AI Tools by Monthly Site Visits", *Visual Capitalist*. https://www.visualcapitalist.com/ranked-most-popular-ai-tools-by-monthly-site-visits/ (pristupljeno 28.6.2025.).

[2] F. Duarte, „Number of ChatGPT Users (June 2025)", *Exploding Topics*. https://explodingtopics.com/blog/chatgpt-users (pristupljeno 28.6.2025.).

[3] L. Banh i G. Strobel, „Generative artificial intelligence", *Electronic Markets*, vol. 33, br. 63, 2023.

[4] R. Bommasani i dr., „On the Opportunities and Risks of Foundation Models", *arXiv preprint arXiv:2108.07258*, 2022.

[5] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi i G. Neubig, „Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing", *arXiv preprint arXiv:2107.13586*, 2021.

[6] H. Naveed i dr., „A Comprehensive Overview of Large Language Models", *arXiv preprint arXiv:2307.06435*, 2024.

[7] K. Babić, S. Martinčić-Ipšić i A. Meštrović, „Survey of Neural Text Representation Models", *Information*, vol. 11, br. 11, art. 511, 2020.

[8] „Research", *OpenAI*. https://openai.com/research/ (pristupljeno 9.7.2025.).

[9] J. Devlin, M.-W. Chang, K. Lee i K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[10] H. Touvron i dr., „LLaMA: Open and Efficient Foundation Language Models", *arXiv preprint arXiv:2302.13971*, 2023.

[11] E. Almazrouei i dr., „The Falcon Series of Open Language Models", *arXiv preprint arXiv:2311.16867*, 2023.

[12] A. Q. Jiang i dr., „Mistral 7B", *arXiv preprint arXiv:2310.06825*, 2023.

[13] D. Kim i dr., „SOLAR 10.7B: Scaling Large Language Models with Simple yet Effective Depth Up-Scaling", *arXiv preprint arXiv:2312.15166*, 2023.

[14] S. J. Russell i P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, London, 2016.

[15] E. H., „LLM Agents Explained: Types, Use Cases, and Future Trends", *PromptLayer Blog*. https://blog.promptlayer.com/types-of-llm-agent/ (pristupljeno 6.7.2025.).

[16] J. Wiesinger, P. Marlow i V. Vuskovic, „Google AI Agents Whitepaper", *Google*. https://archive.org/details/google-ai-agents-whitepaper (pristupljeno 9.7.2025.).

[17] M. Allen, „SQLite Baza podataka: Kako stvoriti, otvoriti, sigurnosno kopirati i ispustiti datoteke", *GURU99*. https://www.guru99.com/hr/sqlite-database.html (pristupljeno 28.6.2025.).

[18] D. Bergmann i C. Stryker, „What is LangChain?", *IBM*. https://www.ibm.com/think/topics/langchain (pristupljeno 28.6.2025.).

[19] F. Oh, „What Is CUDA?", *NVIDIA*. https://blogs.nvidia.com/blog/what-is-cuda-2/ (pristupljeno 29.6.2025.).

[20] „searxng/searxng", *GitHub*. https://github.com/searxng/searxng (pristupljeno 29.6.2025.).

[21] J. Horcasitas, „What Is REPL?", *DigitalOcean Community*, 31.7.2021. https://www.digitalocean.com/community/tutorials/what-is-repl (pristupljeno 29.6.2025.).

[22] V. Singh, „Why LLMs Are Bad at Math — and How They Can Be Better", *Reach Capital*, 16.7.2024. https://www.reachcapital.com/2024/07/16/why-llms-are-bad-at-math-and-how-they-can-be-better/ (pristupljeno 29.6.2025.).

[23] D. Podell i dr., „SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis", *arXiv preprint arXiv:2307.01952*, 2023.

[24] S. Lin, A. Wang i X. Yang, „SDXL-Lightning: Progressive Adversarial Diffusion Distillation", *arXiv preprint arXiv:2402.13929*, 2024.

[25] „Hyper-SD: Trajectory Segmented Consistency Model for Efficient Image Synthesis", *Hyper-SD Project*. https://hyper-sd.github.io/ (pristupljeno 29.6.2025.).

[26] PierrunoYT, „Kokoro-TTS-Local: A local implementation of the Kokoro Text-to-Speech model", *GitHub*. https://github.com/PierrunoYT/Kokoro-TTS-Local (pristupljeno 29.6.2025.).

# Popis slika

Slika 1. Opća arhitektura agenta i komponente. Izvor: [16] ..... 11
Slika 2. Dijagram arhitekture sustava Lumen. ..... 14
Slika 3. Dijagram entiteta i veza baze podataka. ..... 16
Slika 4. Primjer elektronske pošte za oporavak lozinke. ..... 17
Slika 5. Primjer elektroničke pošte koju dobivaju novi korisnici sustava. ..... 18
Slika 6. Primjer rezultata pretraživanja pomoću SearXNG-a. ..... 21
Slika 7. Usporedba slika generiranih pomoću SDXL i SDXL Lightning modela. Izvor: [25] ..... 23
Slika 8. Primjer slike generirane pomoću sustava Lumen. ..... 24
Slika 9. Isječak koda koji prikazuje proces slanja elektroničke pošte. ..... 25
Slika 10. Primjer elektroničke pošte poslane kroz sustav Lumen. ..... 25
Slika 11. Stranica za prijavu korisnika. ..... 28
Slika 12. Stranica za oporavak lozinke. ..... 29
Slika 13. Stranica koja potvrđuje slanje zahtjeva za oporavak lozinke. ..... 30
Slika 14. Stranica za promjenu lozinke. ..... 31
Slika 15. Poruka o uspješnoj promjeni lozinke. ..... 32
Slika 16. Administratorsko sučelje. ..... 33
Slika 17. Stranica za razgovor s Lumen agentom. ..... 34
Slika 18. Stranica za razgovor s otvorenim burger menijem. ..... 35
Slika 19. Alternativni prikaz stranice za razgovor s agentom. ..... 36
Slika 20. Prikazivanje generiranih slika. ..... 37
Slika 21. Prikazivanje generiranih slika kod alternativnog prikaza stranice. ..... 38
Slika 22. Sučelje prilikom snimanja glasa. ..... 39

# Popis priloga

Prilog 1: Cjelovit programski kôd