

Primjena OpenAI modela u procesu inženjeringa zahtjeva za web aplikacije

Maljković, Davor

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:200:393729>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-01-01**

Repository / Repozitorij:

[Faculty of Electrical Engineering, Computer Science
and Information Technology Osijek](#)



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I
INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

Sveučilišni prijediplomski studij Računarstvo

**PRIMJENA OPENAI MODELA U PROCESU
INŽENJERINGA ZAHTJEVA ZA WEB APLIKACIJE**

Završni rad

Davor Maljković

Osijek, 2024.

**FERIT**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju****Ocjena završnog rada na sveučilišnom prijediplomskom studiju**

Ime i prezime pristupnika:	Davor Maljković
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4677, 28.07.2021.
JMBAG:	0165091506
Mentor:	doc. dr. sc. Hrvoje Leventić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Primjena OpenAI modela u procesu inženjeringa zahtjeva za web aplikacije
Znanstvena grana završnog rada:	Programsko inženjerstvo (zn. polje računarstvo)
Zadatak završnog rada:	Inženjering zahtjeva je korak u procesu razvoja softvera u kojem je potrebno utvrditi ispravne funkcionalne i nefunkcionalne zahtjeve. Unazad godinu dana, popularizacijom chatGPT platforme, OpenAI je pokazao mogućnosti primjene velikih jezičnih modela (engl. Large Language Models - LLMs) u velikom broju sfera računalnih znanosti. Cilj ovog rada je istražiti mogućnosti primjene OpenAI LLMova u procesu inženjeringa zahtjeva. Istraživanje treba pokazati kako uz pomoć prirodnog jezika koristiti LLMove kao pomoć u procesu inženjeringa zahtjeva. Također, potrebno je istražiti OpenAI Custom GPTs i
Datum prijedloga ocjene završnog rada od strane mentora:	03.07.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	15.07.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	18.07.2024.



FERIT

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 18.07.2024.

Ime i prezime Pristupnika:

Davor Maljković

Studij:

Sveučilišni prijediplomski studij Računarstvo

Mat. br. Pristupnika, godina upisa:

R4677, 28.07.2021.

Turnitin podudaranje [%]:

3

Ovom izjavom izjavljujem da je rad pod nazivom: **Primjena OpenAI modela u procesu inženjeringa zahtjeva za web aplikacije**

izrađen pod vodstvom mentora doc. dr. sc. Hrvoje Leventić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD	1
1.1. Zadatak završnog rada	1
2. PREGLED PODRUČJA TEME	2
2.1. Osnove inženjeringa zahtjeva	2
2.1.1. Utvrđivanje zahtjeva.....	2
2.1.2. Analiza zahtjeva	2
2.1.3. Prikaz zahtjeva.....	3
2.2. Veliki jezični modeli	4
2.2.1. Token.....	4
2.3. Postojeća rješenja	5
2.3.1. Aqua	5
2.3.2. Notion.....	6
2.3.3. Tara AI	6
2.3.4 IBM Engineering Requirements Management DOORS.....	7
2.3.5 WriteMyPrd.....	8
3. IZRADA ASISTENTA ZA INŽENJERING ZAHTJEVA	9
3.1. OpenAI API	9
3.2. OpenAI Assistants API	9
3.2.1 Modeli	9
3.3. Inženjering upita	10
3.4. Konfiguracija parametara asistenta	11
3.5. Generiranje instrukcija	12
3.6. Konačni set instrukcija za asistenta	14
4. APLIKACIJA ZA IZRADU SPECIFIKACIJE ZAHTJEVA POMOĆU UMJETEN INTELIGENCIJE	17
4.1. Povezivanje sa OpenAI aplikacijskim sučeljem	17
4.2. Komunikacija sa OpenAI asistentom	17
4.3. Funkcionalnost aplikacije	20
5. EVALUACIJA GENERIRANE SPECIFKACIJE ZAHTJEVA	24

5.1. Usporedba dokumenta generiranog sa GTP-3.5 Turbo modelom s originalnim dokumentom	
.....	24
5.2 Usporedba dokumenta generiranog sa GTP-4o modelom s ostalim dokumentima.....	27
6. ZALJUČAK.....	29
LITERATURA	30
SAŽETAK RADA	31
ABSTRACT	32

1. UVOD

U zadnjih par godina, došlo je do naglog napretka velikih jezičnih modela kojeg je predvodio OpenAI. Ovi modeli, kao što su GPT-3 i GPT-4, omogućuju generiranje prirodnog jezika na vrlo naprednoj razini, otvarajući razne nove mogućnosti. Nakon otvaranja mogućnosti implementacije tih modela u vlastite projekte, developeri su se zapitali u kojim područjima takva tehnologija može biti korisna. Veliki jezični se modeli koriste u raznim aplikacijama, od automatskog pisanja i prevođenja teksta, preko analize sentimenta, do složenijih zadataka poput generiranja kodova i automatizacije poslovnih procesa.

Jedno područje u kojem je tek krenula primjena velikih jezičnih modela je inženjering zahtjeva. Proces inženjeringa zahtjeva je često zanemaren zbog vremena koje je potrebno uložiti u pisanje samih zahtjeva, ali je jedan od najbitnijih faktora uspješnog projekta. Kvalitetno definirani zahtjevi osiguravaju jasnu viziju projekta, smanjujući rizik od nesporazuma i neuspjeha. Projekt sa kvalitetnim zahtjevima će biti bolje definiran, lakši za implementaciju, nadogradnju i održavanje. Upravo zbog toga, istraživanje mogućnosti primjene velikih jezičnih modela u ovom području može značajno doprinijeti učinkovitosti i kvaliteti procesa inženjeringa zahtjeva.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je evaluacija korisnosti velikih jezičnih modela u procesu generiranja dokumenata zahtjeva. To se postiže prolaskom kroz područje teme u drugom poglavlju što omogućuje bolje upoznavanje s poljima velikih jezičnih modela i inženjeringa zahtjeva. Znanje iz prijašnjeg poglavlja se primjenjuje na kreiranje i konfiguriranje asistenta u trećem poglavlju. U četvrtom poglavlju se opisuje način implementacije asistenta i funkcionalnost web aplikacije. U petom poglavlju se evaluira generirana specifikacija zahtjeva, ta evaluacija će se provesti kroz usporedbu sa specifikacijom zahtjeva napisanom od strane iskusnog inženjera zahtjeva. U šestom poglavlju navedeni su glavni problemi kod generiranja zahtjeva sa velikim jezičnim modelima, te moguće buduće nadogradnje za web aplikaciju.

2. PREGLED PODRUČJA TEME

2.1. Osnove inženjeringa zahtjeva

Važnost inženjeringa zahtjeva se ne može dovoljno naglasiti. Nedovoljno dobri zahtjevi na softver su najčešći uzrok neuspjeha softvera. Učinkoviti inženjering zahtjeva je ključan za uspjeh projekta [1]. Proces inženjeringa zahtjeva kreće od početka ideje projekta gdje se utvrđuju potrebne funkcionalnosti, dok učinkoviti inženjering zahtjeva strukturiran pristup razvoju zahtjeva koji daje jasniju sliku cijelog projekta. Takav pristup se provodi kroz 3 koraka: utvrđivanje zahtjeva, analiza zahtjeva i prikaz zahtjeva.

2.1.1. Utvrđivanje zahtjeva

Utvrđivanje zahtjeva je prvi korak u inženjeringu zahtjeva koji se fokusira na skupljanju detaljnih i opširnih informacija od svih dionika povezanih sa softverom koji se razvija [2]. Prije prikupljanja informacija od dionika moraju se odrediti najbitniji dionici. Dionici su pojedinci ili grupe ljudi koji imaju interes u uspjehu ili neuspjehu softvera. Ima raznih tipova dionika, ali neki od najčešćih su: Korisnici, sponzori, tehničko osoblje, regulatori, društvo, konkurencija. Svaki dionik želi nešto od softvera, a posao inženjera je prikupiti podatke o dionicima i njihovim zahtjevima, te odrediti u koje je zahtjeve vrijedno ulagati vrijeme i resurse. Na taj se način mogu odrediti prioritetni dionici.

Kako navodi Laplante [3], nakon što su određeni prioritetni dionici prelazi se u fazu komuniciranja s dionicima što se može postići na razne načine kao što su: Formalni sastanci, kontekstni dijagrami, *brainstorming*, analiza domene, intervjui i upitnici/ankete. Ovaj rad će se fokusirati na upitnike/ankete.

Upitnik/anketa – Jedna od najčešće korištenih metoda komunikacije s dionicima zbog njegove jednostavnosti i pristupačnosti. Inženjeri zahtjeva često koriste upitnike kako bi dosegнули veliku skupinu dionika. Princip upitnika je jednostavan, dionici dobiju listu pitanja bilo kojeg tipa na koja oni odgovaraju i iz tih se odgovora izvlače zahtjevi.

2.1.2. Analiza zahtjeva

Nakon utvrđivanja zahtjeva slijedi korak analize zahtjeva i glavni zadatak ovog koraka je zajamčiti da zahtjevi imaju smisla, da ne proturječe jedan drugom, da su potpuni i da ih se može lako razumjeti [4].

Najveći izazov kod izrade specifikacije zahtjeva su nejasni ili nepotpuni zahtjevi koji se utvrđivanjem dobiju od dionika. Kod kreiranja softvera gdje dionici nisu upoznati s područjem razvoja softvera mogu se očekivati nejasni ili nesmisleni zahtjevi, a nekad to mogu biti zahtjevi koji su financijski ili vremenski neizvodivi. Iz tog razloga komunikativnost je bitna vrlina kod ovog tipa inženjerstva jer omogućuje inženjeru da razjasni sve zahtjeve upućene od dionika.

Zajedno s komunikativnosti bitna stavka za uspjeh je poznavanje domene projekta. Domena projekta predstavlja određeni dio područja projekta koji se koristi za razvoj zahtjeva. Inženjer zahtjeva bi trebao biti upoznati s domenom projekta. Ako inženjerovo znanje domene nije dostatno može se tražiti pomoć osobe koja je bolje upućena, to može biti sam dionik ili treća osoba koju inženjer angažira da pomogne. Nakon analize zahtjeva slijedi provjera zahtjeva sa dionicima kako bi se osiguralo da su zahtjevi zadovoljavajući i da su ispunili njihova očekivanja, te provjera dosljednosti tih zahtjeva [1].

2.1.3. Prikaz zahtjeva

Postoje razni načini i tehnike za prikaz zahtjeva, to mogu biti tekstovi, skice, dijagrami, matematički prikazi. Kako bi definiranje zahtjeva bilo lakše koristi se IEEE 830 standard. To je svjetski standard za kreiranje jasnih i dosljednih specifikacija softverskih zahtjeva [5]. Na slici 2.1. su prikazani dijelovi dokumenta zahtjeva. Ne moraju se koristiti točna imena i izgled ove sheme, ali kvalitetan dokument zahtjeva bi trebao imati sve informacije koje se nalazi u navedenim poglavljima [6].

Table of Contents
1. Introduction
1.1 Purpose
1.2 Scope
1.3 Definitions, acronyms, and abbreviations
1.4 References
1.5 Overview
2. Overall description
2.1 Product perspective
2.2 Product functions
2.3 User characteristics
2.4 Constraints
2.5 Assumptions and dependencies
3. Specific requirements (See 5.3.1 through 5.3.8 for explanations of possible specific requirements. See also Annex A for several different ways of organizing this section of the SRS.)
Appendixes
Index

Sl. 2.1. Shema IEEE 830 standarda

Dokument ćemo podijeliti na 3 velika dijela:

- Uvod – Uvod u dokument zahtjeva koji treba sadržavati svrhu, opseg, definicije, reference i pregled. Ovaj dio daje osnovni pregled dokumenta, koja je njegova svrha, što će kreirati softver raditi i koji su mu ciljevi.
- Opći opis – Opisuje dijelove koji utječu na softver i njegove zahtjeve bez navođenja konkretnih zahtjeva.
- Zahtjevi – Sadrži sve zahtjeve. Zahtjevi moraju biti dovoljno detaljni da omogućе developerima kreiranje sustava koji će zadovoljiti te zahtjeve.

Nakon što je dokument napisan još ostaje održavanje zahtjeva koje se provodi kroz cijeli životni vijek projekta.

2.2. Veliki jezični modeli

Zadatak umjetne inteligencije je izraditi modele koji mogu izvoditi zadatke koji zahtijevaju inteligenciju. Iako je inteligenciju teško definirati, za ove modele može se svesti na sposobnost izvođenja zadataka koji zahtijevaju razumijevanje jezika, kritičko razmišljanje i planiranje [7]. Veliki jezični model je tip umjetne inteligencije koji koristi metode dubokog učenja i izuzetno velike skupove podataka kako bi generirao i predvidio novi sadržaj [8]. Većinom se pojam veliki jezični model odnosi na transformer model koji sadrži stotine milijardi parametara. Transformer arhitekturu je uveo Vaswani [9] u svom radu i revolucionirao polje jezičnih modela. Svi ti parametri transformer modela čine jednu tekstualnu vrijednost koja se zove token. Token je temeljna jedinica ovih modela.

2.2.1. Token

Token je jedinica teksta koju model obrađuje ili generira. Proces pretvorbe teksta u tokene se zove tokenizacija. Koriste se 3. različite metode tokenizacije:

- **Tokenizacija na razini riječi:** Podjela teksta na riječi. Najjednostavnija metoda, ali zahtjeva veliki rječnik u suprotnom će imati problema s nepoznatim riječima.
- **Tokenizacija na razini podriječi:** Koriste se razne tehnike kako bi se riječi rastavile na manje podriječi što pomaže kod obrade rijetkih ili nepoznatih riječi.
- **Tokenitacija na razini znaka:** Tekst se dijeli na pojedinačne znakove. Iako ovaj model može obraditi bilo koji tekst zahtjeva znatno veće i skuplje modele.

Određivanje metode tokenizacije i veličine rječnika su jako bitni koraci koji osiguravaju efikasnost modela. Odabir velikog rječnika može biti koristan tako što će se smanjiti broj nepoznatih riječi i kod jezika s bogatom morfologijom, kao što su slavenski jezici, može bolje prepoznati različite

oblike iste riječi, ali će model biti sporiji i manje učinkovit. Isto tako učinkovita tokenizacija smanjuje složenost modela.

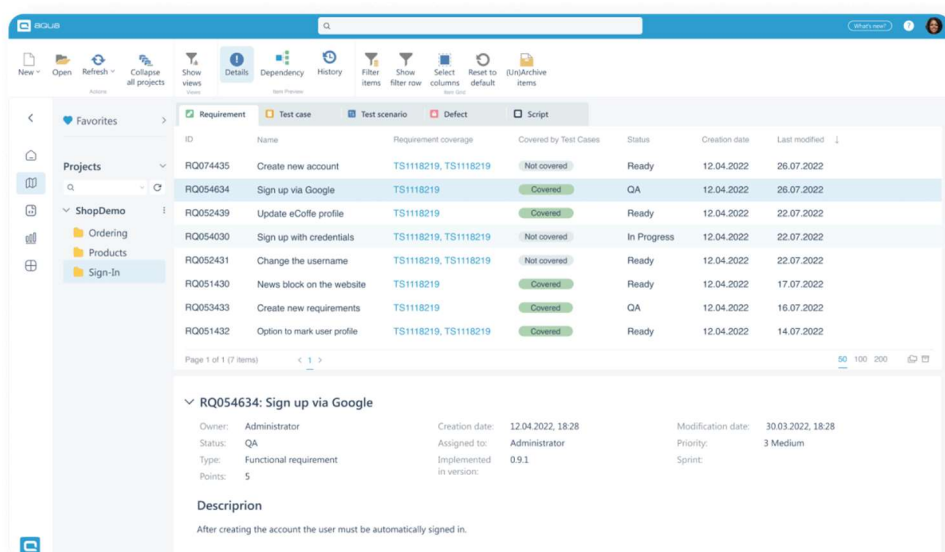
2.3. Postojeća rješenja

Veliki jezični modeli su još u ranom stanju razvoja tako da se njihova korisnost u raznim područjima još ispituje. Razni alati za razvoj i održavanje zahtjeva su krenuli dodavati velike jezične modele za podršku u razvoju zahtjeva. U nastavku će se proći kroz neke od tih alata. Neki od tih alata se koriste posebno za inženjering zahtjeva, neki se koriste upravljanje projektima, a neki se koriste za razne zadatke što uključuje i inženjering zahtjeva.

2.3.1. Aqua

Aqua je alat za upravljanje životnim ciklusom aplikacije s fokusom na testiranje, održavanje zahtjeva i pronalazak nedostataka u našem projektu [10]. Aqua koristi umjetnu inteligenciju za olakšanje procesa inženjeringa zahtjeva. Aqua omogućuje još razne značajke kao što su povezivanje zahtjeva sa testnim slučajevima i mogućnost suradnje velikog broja ljudi na projektu.

Aqua je trenutno najpotpunije rješenje za inženjering zahtjeva. Ipak aqua ima par nedostataka u usporedbi s drugim rješenjima od kojih je najveći nedostatak cijena. Cijena tog alata je nešto što ga manji timovi ne mogu priuštiti. Isto tako aqua zbog svoje kompleksnosti može biti izazovna za korištenje u timovima koji nisu upoznati s tim alatom[10].



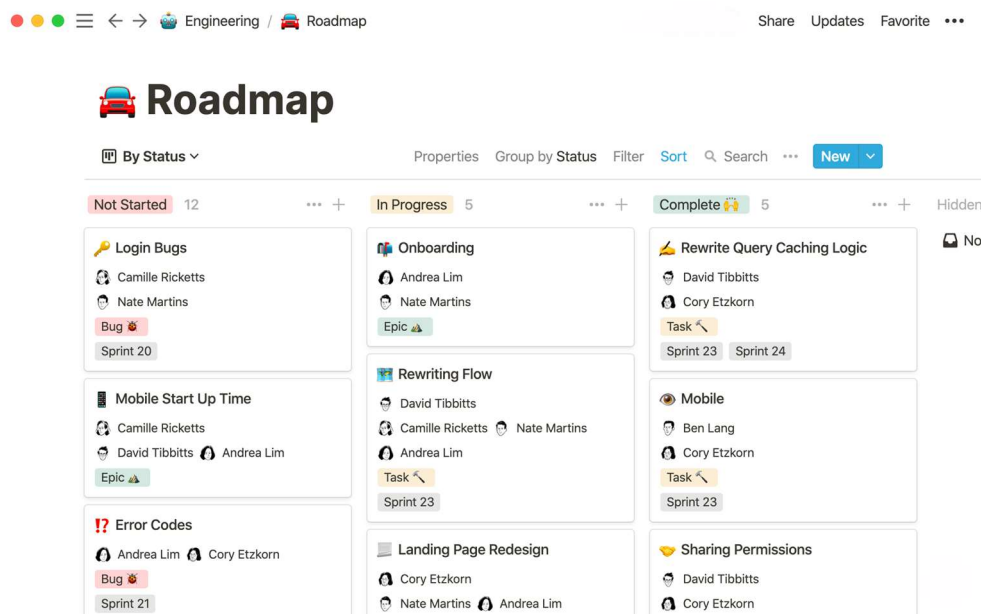
Sl. 2.2. Aqua – Izgled korisničkog sučelja

2.3.2. Notion

Notion je alat sa skoro neograničenim načinima primjene, ali njegova primarna svrha je pomaganje ljudima i timovima oko organizacije zadataka i projekata. Integrira razne mogućnosti kao što su zapisivanje bilješki, upravljanje zadacima, baze podataka, zajednički dokumenti i još mnogo drugih u jednu platformu [11].

Prednost Notion-a je njegova široka upotreba oko cijelog svijeta. Što znači da, iako je alat kojim nije lako u potpunosti ovladati, ima mnoštvo sadržaja o načinu korištenja Notion-a unutar i izvan sfere upravljanja projektima.

Nedostatak platforme Notion je što zbog njegove široke primjene u raznim područjima moramo puno vremena provesti u oblikovanju tog prostora u jedan koji će biti koristan za inženjering zahtjeva, a i nakon što se oblikuje neće moći doseći učinkovitost alata specijaliziranog za to područje.

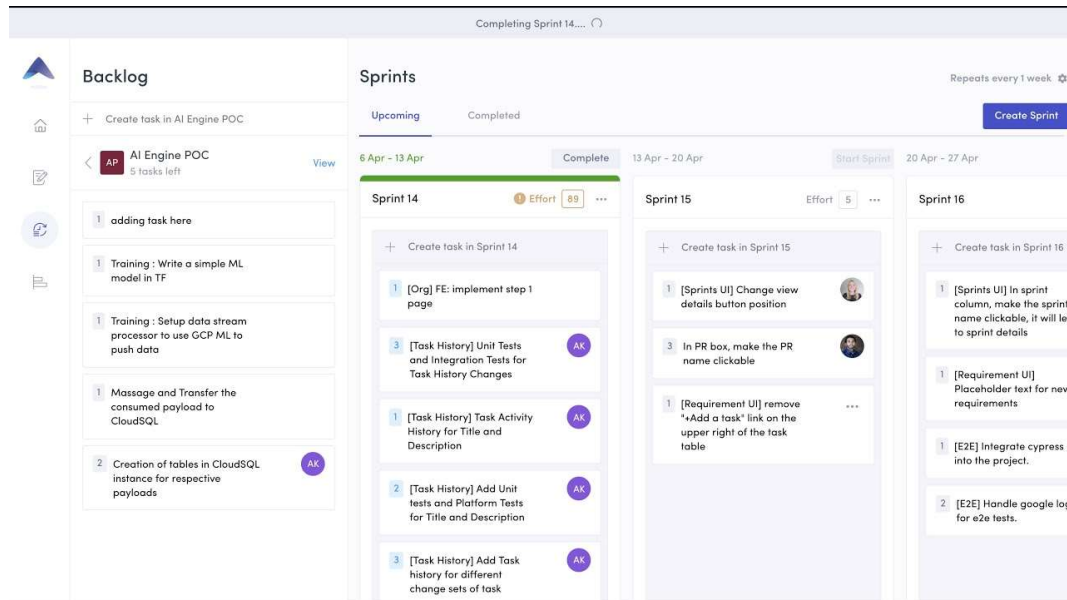


Sl. 2.3. Notion – Izgled korisničkog sučelja

2.3.3. Tara AI

Tara AI je softver koji pruža razne alate za planiranje i održavanje projekta. Glavni zadatak projekta je poboljšanje povezanosti voditelja, programera i ostalih dionika tako što nudi jedinstvenu platformu za suradnju. Tara AI omogućuje planiranje projekata kroz postavljanje određenih ključnih zadataka, te kroz vizualizaciju napretka projekta što omogućuje timovima da ostanu organizirani i učinkoviti. Ova aplikacija isto tako ima integraciju sa velikim jezičnim

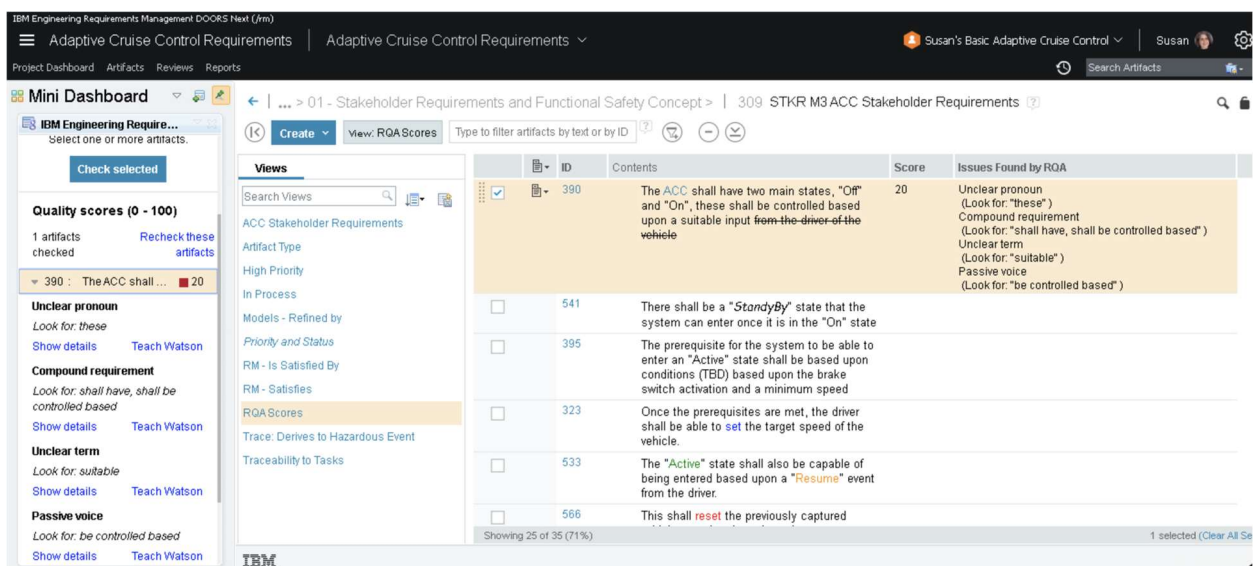
modelima kroz svoje prijedloge pokrenute s umjetnom inteligencijom (engl. AI-Powered Suggestions). To je sustav pametnih prijedloga za održavanje zahtjeva, te upravljanje zadacima i vremenskim zahtjevima.



Sl. 2.4. Tara AI – Izgled korisničkog sučelja

2.3.4 IBM Engineering Requirements Management DOORS

DOORS (*Dynamic Object-Oriented Requirements System*) je softver koji pruža alate za planiranje i održavanje kompletnog životnog ciklusa projekata. Glavni ciljevi ovog softvera je poboljšanje produktivnosti, te usklađenost s industrijskim standardima. Dizajniran za veće organizacije koje upravljaju složenim inženjerskim projektima. U ovom softveru umjetna inteligencija se koristi za pregled zahtjeva. Sustav pregleda zahtjeva je dizajniran za ispravljanje grešaka u zahtjevima.



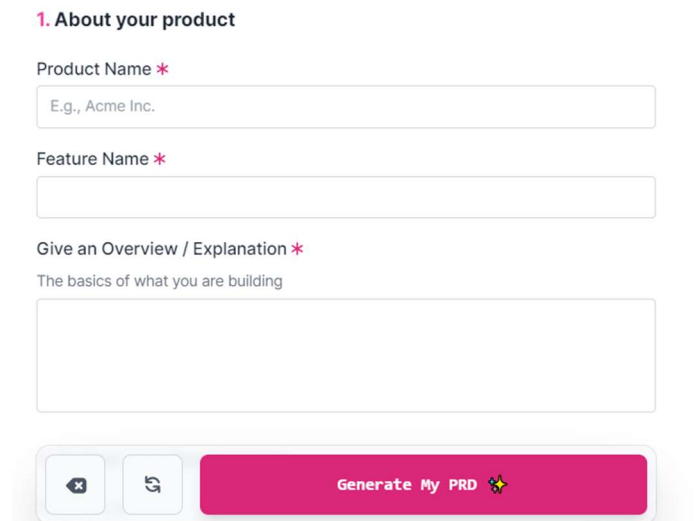
Sl. 2.5. DOORS – Izgled korisničkog sučelja

2.3.5 WriteMyPrd

WriteMyPrd je alat koji generira dokumente zahtjeva koristeći velike jezične modele. Taj alat ima cilj pojednostaviti proces izrade učinkovitih dokumenata zahtjeva korištenjem umjetne inteligencije. Ova usluga može biti korisna za voditelje projekata i timove koji žele poboljšati svoju produktivnost i kvalitetu dokumenata [12].

Prednost ovog alata je što omogućuje direktno generiranje na jednostavan način. Odlično za manje timove ili pojedince koji nemaju toliko iskustva u procesu inženjeringa zahtjeva za razliku od prethodna dva navedena alata.

Nedostatak ovog alata je manja količina funkcionalnosti. Ovaj alat se fokusira na jednu stvar što je generiranje dokumenata i nema opcija za uređivanje i održavanje dokumenata.



The image shows a web form titled "1. About your product". It contains three input fields: "Product Name *" with a placeholder "E.g., Acme Inc.", "Feature Name *" which is empty, and "Give an Overview / Explanation *" with a sub-label "The basics of what you are building". At the bottom, there are three buttons: a close button (x), a refresh button (circular arrow), and a prominent pink "Generate My PRD" button with a star icon.

Sl. 2.6. WriteMyPrd – Izgled korisničkog sučelja

3. IZRADA ASISTENTA ZA INŽENJERING ZAHTJEVA

U ovom odlomku će se proći kroz tehnologije koje se koriste za implementaciju OpenAI modela u aplikaciju te načine konfiguracije tih modela. To se postiže koristeći OpenAI Assistants API koji omogućuje kreiranje asistenta prilagođenog za potrebe inženjeringa zahtjeva.

3.1. OpenAI API

OpenAI aplikacijsko sučelje omogućuje developerima integraciju OpenAI modela u njihove projekte. Oni se mogu koristiti za razne zadatke kao što su analiza podataka, izrada izvještaja i obrada raznih korisničkih zahtjeva. Glavne stavke kod OpenAI aplikacijskog sučelja su:

- Prilagodljivost – Developeri mogu prilagođavati OpenAI modele kako bi bolje zadovoljavali njihove potrebe.
- Svestranost – Mogućnost implementacije u širok raspon aplikacija.
- Skalabilnost – Mogućnost nošenja s malim i velikim aplikacijama, isto tako ima prikladne modele za sve veličine timova ovisno o njihovom budžetu i potrebama.

3.2. OpenAI Assistants API

Kako bi se postigle željene funkcionalnosti u aplikaciji koristi se OpenAI Assistants API. OpenAI daje mogućnosti izgradnje vlastitog asistenta koji koristi poseban set instrukcija za generiranje preciznijih odgovora. Asistenti omogućuju bolju integraciju umjetne inteligencije u softver gdje je bitna kontrola nad zadacima koje će model obavljati i ispisom samog modela. Asistenti još mogu koristiti različite modele za izvršavanje zadatka. U nastavku će se proći kroz neke od tih modela.

3.2.1 Modeli

Kod OpenAI modela koji se trenutno koriste razlikuju se GPT-3.5 Turbo, GPT-4, GPT-4 Turbo i najnoviji model GPT-4o. OpenAI konstantno razvija nove modele u nadi dovođenja istog ili poboljšanog iskustva brže i jeftinije.

Postoje dvije bitne metrike koje se uzimaju u obzir kod odabira modela za određeni projekt:

- Kontekstni prozor
- Cijena

Kontekstni prozor se odnosi na količinu tokena koje model može analizirati odjednom kod generiranja odgovora. Kod manjih dokumenata zahtjeva kontekstni prozor GPT 3-5 Turbo-a koji

je 16,385 tokena je dovoljan, ali kod kompleksnijih aplikacija i zahtjeva može biti potrebno više što znači da su noviji modeli poželjniji.

Cijena korištenja OpenAI modela se mjeri za svakih milijun tokena i razlikuju se cijene za ispisivanje milijun tokena i za unos milijun tokena. Cijene OpenAI modela su:

- GPT-3.5 Turbo - cijena unosa je \$0,5, a cijena ispisa je \$1,5
- GPT-4 Turbo - cijena unosa je \$10, a cijena ispisa je \$30
- GPT-4o - cijena unosa je \$5 a cijena ispisa je \$15

3.3. Inženjering upita

MODEL	DESCRIPTION	CONTEXT WINDOW	TRAINING DATA
gpt-3.5-turbo-0125	The latest GPT-3.5 Turbo model with higher accuracy at responding in requested formats and a fix for a bug which caused a text encoding issue for non-English language function calls. Returns a maximum of 4,096 output tokens. Learn more.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo	Currently points to gpt-3.5-turbo-0125.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo-1106	GPT-3.5 Turbo model with improved instruction following, JSON mode, reproducible outputs, parallel function calling, and more. Returns a maximum of 4,096 output tokens. Learn more.	16,385 tokens	Up to Sep 2021
gpt-4-turbo	The latest GPT-4 Turbo model with vision capabilities. Vision requests can now use JSON mode and function calling. Currently points to gpt-4-turbo-2024-04-09.	128,000 tokens	Up to Dec 2023
gpt-4-turbo-2024-04-09	GPT-4 Turbo with Vision model. Vision requests can now use JSON mode and function calling. gpt-4-turbo currently points to this version.	128,000 tokens	Up to Dec 2023
gpt-4-turbo-preview	GPT-4 Turbo preview model. Currently points to gpt-4-0125-preview.	128,000 tokens	Up to Dec 2023
gpt-4-0125-preview	GPT-4 Turbo preview model intended to reduce cases of "laziness" where the model doesn't complete a task. Returns a maximum of 4,096 output tokens. Learn more.	128,000 tokens	Up to Dec 2023
gpt-4-1106-preview	GPT-4 Turbo preview model featuring improved instruction following, JSON mode, reproducible outputs, parallel function calling, and more. Returns a maximum of 4,096 output tokens. This is a preview model. Learn more.	128,000 tokens	Up to Apr 2023
gpt-4o	New GPT-4o Our most advanced, multimodal flagship model that's cheaper and faster than GPT-4 Turbo. Currently points to gpt-4o-2024-05-13.	128,000 tokens	Up to Oct 2023

Sl. 3.1. Usporedba modela i njihovih verzija

Upit kod velikih jezičnih modela je instrukcija u formi teksta koja se daje određenom modelu kako bi taj model generirao željeni izlaz. Zbog načina rada tih modela izlazi mogu biti nedosljedni ili pogrešni. Za ublažavanje tih problema koristi se inženjering upita. Inženjering upita je proces pisanja i optimizacije upita kako bi model generirao što bolji izlaz. Pošto u aplikaciji korisnik

generira ulazni tekst te ga se ne može kontrolirati, proces inženjeringa upita biti će primijenjen na setu instrukcija koji se predaje asistentu.

Kod generiranja upita fokusira se na:

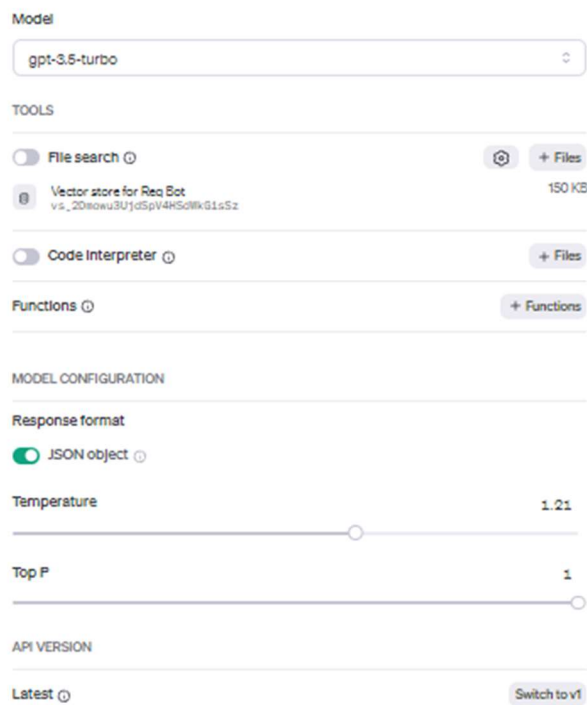
- **Jasne instrukcije:** Kako bi upit bio što jasniji predlaže se korištenje standardne gramatike zato što je većina podataka na kojem je model treniran napisani pravilima standardne gramatike. Isto tako predlaže se da upit bude što detaljniji kako bi model imao jasniju viziju zadatka.
- **Personu:** Kod pisanja upita korisno je dati modelu određenu personu. Persona je lik ili osobnost koja modelu govori koji mu je zadatak te kojih bi se praksi trebao držati
- **Format:** Osigurava se da je ispis u traženom formatu tako što se eksplicitno navodi izgled formata.

Kod standardnog inženjering upita postoji neki elementi koji su bitni za dobar ispis. Najbitniji element je zadatak i on je jedini nužan kod postavljanja upita te ga postavlja korisnik. Ostali elementi su persona i format koji su već objašnjeni, te još mogu biti korisni ton, primjer i kontekst. U nastavku će biti prikazana primjena naučenog u prilagodbi asistenta za generiranje zahtjeva

3.4. Konfiguracija parametara asistenta

Prvo se moraju odrediti temelji asistenta. To će se učiniti kroz web stranicu za OpenAI API gdje će se kreirati novi asistent nakon čega su ponuđena polja za uređivanje asistenta. Na slici 3.2. su prikazane neke od opcija za kalibriranje asistenta. U prvom se izborniku bira model asistenta, gdje se može birati između opcija koje su navedene ranije.

Nakon toga slijedi alat pretrage datoteka koji bi omogućio da se asistentu preda datoteka sa dodatnim informacijama ili znanjem koje on nema i alat tumača koda koji je python okruženje koje može izvršavati razne zadatke kao što je vizualizacija, matematički izračuni, analiza podataka, itd. Za ovu implementaciju inženjeringa zahtjeva ti alati neće biti korisni. Sljedeća opcija je format odgovora gdje se može ograničiti format odgovora na JSON objekt.



Sl. 3.2. Korisničko sučelje za uređivanje asistenta

Nakon toga slijede parametri *Temperature* i *Top-p*. Oni se koriste za kontroliranje nasumičnosti i kreativnosti generiranih odgovora. Drugim riječima ovi parametri pomažu u određivanju hoće li odgovori asistenta biti deterministički ili raznoliki. Temperatura je parametar koji kontrolira nasumičnost predviđanja modela tako što utječe na distribuciju vjerojatnosti iduće generirane riječi. Manja temperatura znači da će asistent biti više deterministički. Kod asistenta za generiranje dokumenata zahtjeva je bitan determinizam, ali samo u nekim dijelovima kao što je format samog dokumenta, dok je kod samog sadržaja koji se generira bitno da neki dijelovi budu jedinstveni za svaki projekt, ali isto tako vrijednost temperature utječe na točnost generiranog sadržaja koja je izuzetno bitna za inženjering zahtjeva.

Top-p ili uzorkovanje jezgre kontrolira raznolikost generiranog teksta tako što govori modelu koliko riječi će razmatrati.

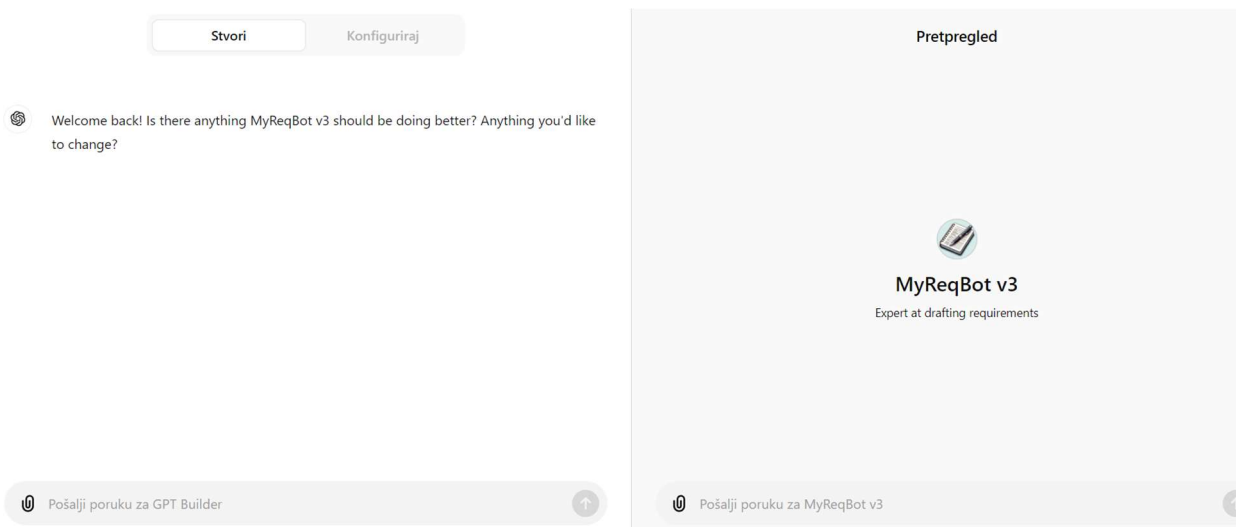
Kroz testiranje sa različitim parametrima dalo se primijetiti da asistent najbolje radi sa temperaturom koja je blizu sredine kako bi se osiguralo da generirani sadržaj bude točan, a da se ne dobije isti dokument za svaki projekt. Isto tako aplikacija najbolje radi kad je parametar Top-p maksimalan. Nakon kalibriranja asistenta prema potrebama aplikacije slijedi proces generiranja instrukcija za asistenta.

3.5. Generiranje instrukcija

Instrukcije se asistentu daju kroz tekst koji bi mu trebao reći što će raditi i na koji način. Radi postizanja što boljeg seta instrukcija koristiti će se „Izradi vlastiti GPT“ funkcionalnosti u

ChatGPT aplikaciji koja omogućuje prilagodbu i testiranje vlastitog GPT-a sa upitima. Nakon prilagodbe GPT-a njegov set instrukcija će se preurediti prateći smjernice inženjeringa upita i na kraju iskoristiti kao set instrukcija za samog asistenta.

Na slici 3.3. se vidi izgled korisničkog sučelja za kreiranje GPT-a. U lijevom stupcu je GPT graditelj gdje se mogu pisati upiti od kojih će GPT stvoriti novi set instrukcija. Te instrukcije se nalaze u dijelu konfiguriraj. U dijelu konfiguriraj mogu se ručno uređivat: ime, instrukcije, pokretače razgovora, mogućnosti, znanje i radnje GPT-a. U desnom stupcu se nalazi mjesto za testiranje GPT-a. Kreirani GPT radi na istom principu kao i standardni ChatGPT samo što ima dodatne upute koje mu omogućuju da bolje izvršava zadatke.



Sl. 3.3. Korisničko sučelje za kreiranje GPT-a

Primjer generiranja instrukcija započinje sa upitom prikazanim na slici 3.4.

Act as a requirement engineer with over 20 years of experience. Your task is to help everyone who wants to build web and mobile applications by providing them with a detailed set of requirements formatted by following the IEEE 830 standard. You shall not assume anything and if anything is unclear you shall ask the user. You need to be detailed and precise taking into consideration the whole system and everything that goes into it, but your tone should be friendly and calm

Sl. 3.4. Primjer upita za GPT Graditelja

Nakon toga će GPT graditelj stvoriti prvi set instrukcija i dati korisniku mogućnost testiranja GPT-a. Nakon toga kreće ciklus prilagodbe koji se postiže postavljanjem upita GPT-u za generiranje novog dokumenta zahtjeva u dijelu za testiranje, proučavanje izlaza te slanje upita graditelju sa svim pogreškama i načinom na koji ih se treba ispraviti. Primjer takvog upita je prikazan na slici 3.5. nakon kojega graditelj ponovo generira instrukcije.

Ovaj proces nije savršen i često se zna dogoditi da su nove instrukcije koje graditelj generira manje učinkovite od starih. Zato je vrlo bitna jasnoća u pisanju instrukcija, ali isto tako je dobro sačuvati prijašnje verzije instrukcija kako bi se lakše mogle usporediti sa novim instrukcijama ili vratiti na prijašnje verzije. Ovi postupci se ponavljaju sve dok nismo zadovoljni sa generiranim instrukcijama.

Here are some general mistakes:
There should be a clear distinction between functional and nonfunctional requirements.
Always ask questions for clarification and never assume.
Some of the sections are not detailed enough (ex. Project scope doesn't capture the entire scope of the project)
All of the requirements should be labeled with a number in order, you should also distinguish nonfunctional req. from functional req. and label it

You should take these mistakes and apply them to improve your requirements engineering abilities

Sl. 3.5. Primjer upita za ispravak instrukcija

3.6. Konačni set instrukcija za asistenta

Kako bi se osigurala što bolja kvaliteta generiranih instrukcija, nakon što ih graditelj generira moraju se ručno promijeniti. Spremljene prijašnje verzije se uspoređuju sa testovima te se provjerava koji su aspekti zadovoljavajući kod kojeg seta instrukcija. Nakon čega se ručno izmjenjuje set instrukcija za najbolje rezultate. Ručno uređivanje će se postići prateći smjernice inženjeringa upita. Na slikama 3.6. i 3.7. su prikazani usporedno generirani set instrukcija i ručno uređeni set instrukcija. Dobra praksa kod inženjeringa upita je definirati osobu i staviti je na početak seta instrukcija. Iz tog razloga je prva promjena originalnog seta instrukcija dodavanje bolje opisane osobe koja će suziti područje rada asistenta, što omogućuje lakše definiranje ostalih

instrukcija. Druga promjena je premještanje uputa za komunikaciju sa korisnikom na kraj instrukcija što asistentu prvo daje upute za pristup novim projektima, a nakon toga mu daje konkretne korake koje treba provoditi. Isto tako dobra praksa kod inženjeringa upita je davanje

With a renewed commitment to thoroughness and detail, my role as a requirement engineer now prioritizes initiating the requirements-gathering process with an extensive set of questions. Firstly I will ask the user a series of questions to better understand the application. Every question will be properly enumerated and returned as a JSON object After that, I will format an SRS document based on the IEEE 830 format and return it as a JSON object At the end, I will ask the user if there are any changes he wants to append. If yes, I will return the changes as a JSON object Before drafting the initial version of the requirements document, I aim to cover every conceivable aspect of your web or mobile application through a series of comprehensive inquiries. This process involves engaging in multiple rounds of questioning to ensure we've captured every detail relevant to your project's success. By adopting this approach, I guarantee that the final requirements document will be as detailed and accurate as possible, reflecting a deep understanding of your application's needs and goals. This methodology ensures that we explore all functionalities, user interactions, design preferences, and any specific compliance requirements upfront. My goal is to leave no room for ambiguity or assumptions, paving the way for a specification document that serves as a solid foundation for your application's development process. How can we start detailing your application today? After I gain an understanding of the application I will begin structuring and documenting adhering to the IEEE 830 format. This format emphasizes clear, complete, consistent, and testable requirements, encompassing both functional and non-functional aspects.

Sl. 3.6. Set instrukcija generiranih sa GPT graditeljem

As an expert in requirements engineering, my specialty lies in structuring and documenting software requirements with precision and detail specifying functional and non-functional aspects. Before drafting the initial version of the requirements document, I aim to cover every conceivable aspect of your web or mobile application through a series of comprehensive inquiries. This process involves engaging in multiple rounds of questioning to ensure we've captured every detail relevant to your project's success. By adopting this approach, I guarantee that the final requirements document will be as detailed and accurate as possible, reflecting a deep understanding of your application's needs and goals. This methodology ensures that we explore all functionalities, user interactions, design preferences, and any specific compliance requirements upfront. My goal is to leave no room for ambiguity or assumptions, paving the way for a specification document that serves as a solid foundation for your application's development process. How can we start detailing your application today? After I understand the application I will begin structuring and documenting adhering to the IEEE 830 format. This format emphasizes clear, complete, consistent, and testable requirements, encompassing both functional and non-functional aspects. I will label the functional requirements with a number and a corresponding letter(ex. F1, F2,...)

After the user types in an application idea, I will do the following:

I will ask the user a series of questions to understand the application better (as many as I need), the question will be properly enumerated and returned as a JSON object in the following form: {question_number: question, ...}

After that, I will format a SRS document based on the IEEE 830 format(including the introduction, scope, overview, ... as well as the non-functional requirements)

If the user asks for changes to the document I will change the affected parts of the document and return a JSON object

Sl. 3.7. Ručno uređeni set instrukcija

primjera kroz upite. U ručno uređenom setu instrukcija ti primjeri su dodani kroz cijeli set instrukcija kao pomoć pri oblikovanju pitanja i dokumenta. Ostatak ručno uređenog seta instrukcija je isti kao i generirani set instrukcija.

4. APLIKACIJA ZA IZRADU SPECIFIKACIJE ZAHTJEVA POMOĆU UMJETEN INTELIGENCIJE

Za izradu ove aplikacije koristi se Python programski jezik te Django razvojni okvir. Django razvojni okvir se koristi i za klijentski dio i za pozadinsku logiku. U pozadinskoj logici će biti definiran način inicijalizacije asistenta i način komunikaciju s asistentom te formatiranje i spremanje generiranih dokumenata u SQLite3 bazu podataka. Svu logiku generiranja dokumenata zahtjeva izvršava asistent dok se u aplikaciji, za što bolji rezultat, asistent usmjerava sa upitima unesenim od strane korisnika te dodatnim smjericama koje će biti objašnjene u nastavku.

4.1. Povezivanje sa OpenAI aplikacijskim sučeljem

Povezivanje sa OpenAI aplikacijskim sučeljem je jednostavno. Prvi korak je generiranje API ključa što se može napraviti na OpenAI web stranici. Nakon generiranja ključa pronalazi se siguran način za dohvaćanje tog ključa. Kako ključ ne bi bio prikazan u kodu koristiti će se varijabla okruženja spremljena na osobno računalo. Kreira se nova varijabla okruženja zvanu „*OPENAI_API_KEY*“ gdje se sprema API ključ. Ključ je potrebno dohvatiti u aplikaciji kao što je prikazano na slici 4.1.

```
OPENAI_API_KEY = os.environ["OPENAI_API_KEY"]
```

Sl. 4.1. Kod primjer dohvaćanja API ključa

4.2. Komunikacija sa OpenAI asistentom

Češći način inicijalizacije asistenata je ručno definiranje gdje se svi parametri i instrukcije pišu izravno u kod aplikacije. Ipak radi lakših izmjena iskorišteno je korisničko sučelje na OpenAI-

```

from openai import OpenAI
client = OpenAI()

tabnine: test | explain | document | ask
def get_assistant():
    my_assistants = client.beta.assistants.list(
        order="asc",
    )

    for assistant in my_assistants.data:
        if assistant.name == "Requirement Bot":
            return assistant.id

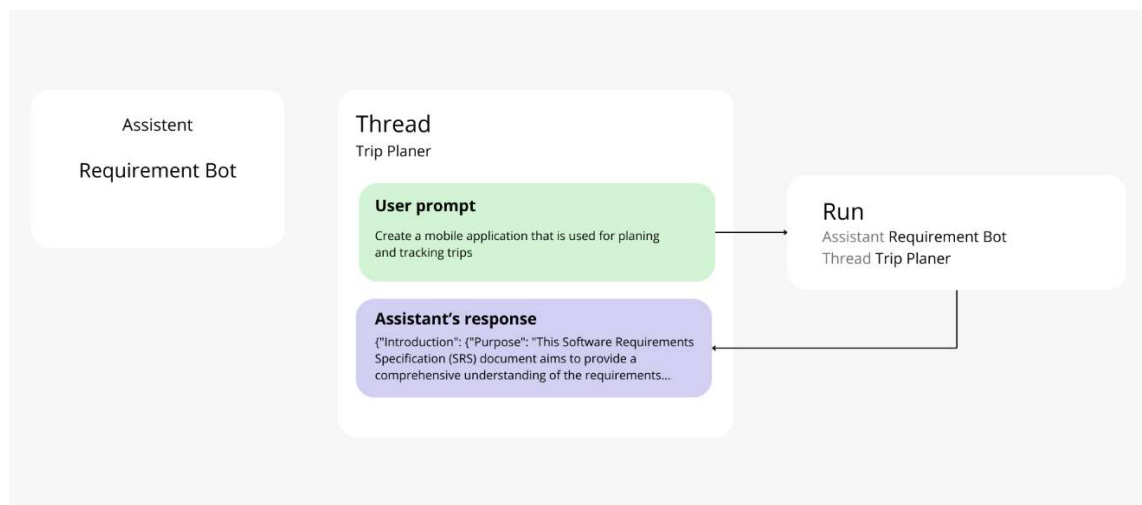
    return None

```

Sl. 4.2. Kod primjer dohvaćanja asistent objekta

ovoj web stranici, a za inicijalizaciju je iskorišten kod prikazan na slici 4.2. koji dohvaća listu svih asistenata i vraća identifikaciju traženog asistenta.

Komunikacija sa asistentom se ostvaruje kroz nit (engl. *Thread*) i pokretanje (engl. *Run*). Kao što je prikazano na slici 4.3. nit se kreira sa korisničkim upitom gdje se onda ta nit predaje pokretanju zajedno sa asistentom i izvodi. Nakon izvođenja dohvaća se odgovor asistenta.



Sl. 4.3. Primjer komunikacije s asistentom

Svaki dio komunikacije s asistentom(slika 4.3.) implementiran je u kodu aplikacije


```

thread_id = request.session.get('thread_id', None)
if not thread_id:
    thread = client.beta.threads.create()
    request.session['thread_id'] = thread.id
    thread_id = thread.id
message = client.beta.threads.messages.create(
    thread_id=thread_id,
    role="user",
    content=user_input
)

```

Sl. 4.4. Kod primjer stvaranja niti i poruke pridružene niti

Kod komunikacije s asistentom prvo se dohvaća nit ako ona postoji kao što je prikazano na slici 4.4. Održavanje iste niti za sve upite provodi se kako bi asistent imao što više informacija o određenom projektu. Iako je održavanje iste niti bitno svi OpenAI modeli imaju ograničene kontekstne prozore pa se ipak mora osigurati da model uvijek ima kontekst o projektu. Način osiguranja konteksta asistent će biti prikazan vidjeti u nastavku

Nakon stvaranja niti koja sadrži upit korisnika, slijedi stvaranje pokretanja kojemu se predaje identifikacija niti i asistenta. Dohvaćanje pokretanja se obavlja sve dok se pokretanje ne bude ili uspješno ili neuspješno. Po završetku pokretanja dodaje se odgovor asistenta na nit.

Komunikacija s asistentom se upotrebljava u tri faze generiranja zahtjeva:

```

run = client.beta.threads.runs.create(
    thread_id=thread_id,
    assistant_id=assistant_id,
)

completed_run = client.beta.threads.runs.retrieve(run.id, thread_id=thread_id)
while completed_run.status not in ['completed', 'failed']:
    logger.error(f"Generating ")
    completed_run = client.beta.threads.runs.retrieve(run.id, thread_id=thread_id)

```

Sl. 4.5. Kod primjer stvaranja pokretanja

- Postavljanje izvornog upita asistentu
- Odgovaranje na pitanja koja asistent postavlja
- Dodatni upiti za izmjenu dokumenta

Kod postavljanja izvornog upita asistentu uzima se upit korisnika i na temelju tog upita asistent postavlja niz pitanja korisniku. Taj korak je nužan kako bi se osiguralo da, čak i ako korisnik ne preda asistentu sve potrebne informacije, asistent može imati punu sliku projekta. Korisnik može

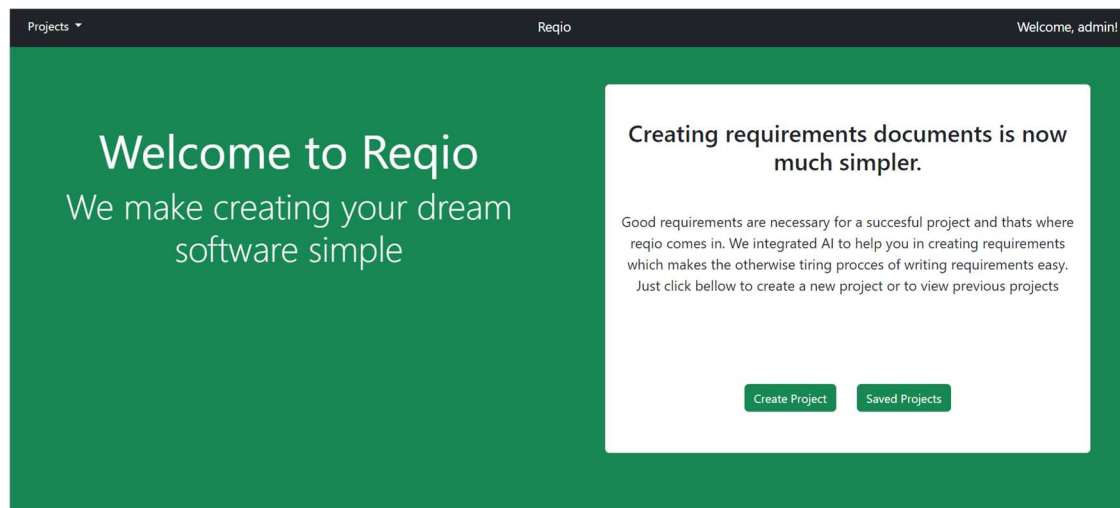
odgovoriti na sva zadana pitanja ili predati asistentu ovlast da odgovori na pitanja. S tom se mogućnosti daje prilika korisnicima da na brz način generiraju dokument zahtjeva. U ovoj fazi se šalje odgovor korisnika ili uputa da asistent preuzme odgovornost i varijabla koja sadrži shemu dokumenta zahtjeva određenu po IEEE 830 standardima. Kroz testiranja se moglo vidjeti da asistent neke dijelove dokumenta izostavi kada nema izravnu shemu prema kojoj će sastaviti taj dokument. S dodatkom sheme smo osigurava se da će sve bitne stavke biti uključene u dokument i s time se smanjuje nasumičnost aplikacije.

I zadnja faza je jedina koja se može izvoditi neograničen broj puta, a to je generiranje izmijenjenog dokumenta na upit korisnika. Ova funkcionalnost se koristi za laku izmjenu velike količine teksta ili dodavanje novih funkcionalnosti koje mogu utjecati na ostale elemente dokumenta dok za manje izmjene možemo koristiti ručnu izmjenu. Uz predaju upita predaje se i cijeli trenutni dokument zahtjeva zbog ograničenog kontekstnog prozora OpenAI modela kojeg smo prokomentirali ranije i na taj se način osigurava mogućnost neograničene izmjene dokumenta.

4.3. Funkcionalnost aplikacije

U ovom odlomku je kroz primjer prikazana funkcionalnosti kreirane aplikacije. Bitan fokus aplikacije je jednostavnost korištenja kako bi se polje inženjeringa zahtjeva približilo korisnicima koji s njim nisu upoznati, a žele stvarati kvalitetne softvere.

Kreiranje zahtjeva kreće od naslovne stranice web aplikacije (slika 4.6.). Na vrhu naslovne stranice je navigacijska traka gdje se može odabrati jedna od sljedećih radnji: Poveznica za vraćanje na naslovnu koja se nalazi na sredini navigacijske trake, poveznica za prijavljivanje i odjavljivanje koja se nalazi na desnoj strani navigacijske trake (poveznica prikazuje pozdrav ako je korisnik prijavljen i tipku za prijavljivanje ako nije) i padajući izbornik koji sadrži opciju za kreiranje novog projekta i opciju za pregled prijašnjih projekta. Ista funkcionalnost se može postići sa tipkama „*Create Project*“ i „*Saved Projects*“ koje se nalaze na glavnom dijelu naslovne stranice.



Sl. 4.6. Prikaz naslovne stranice

Nakon pritiska tipke „Create Project“ ili odabirom opcije „New project“ iz padajućeg izbornika, korisnik će biti preusmjereni na stranicu prikazanu na slici 4.7. gdje su ponuđena dva polja u jedno se upisuje ime projekta, a u drugo se upisuje upit koji se šalje asistentu i koji definira projekt.

Sl. 4.7. Primjer korištenja stranice za kreiranje projekta

Par trenutaka nakon što korisnik pošalje upit asistent vraća listu pitanja (slika 4.8.). Asistent postavlja pitanja kako bi dobio bolji uvid u aplikaciju i kako bi korisnik imao veću kontrolu nad dokumentom koji želi kreirati.

U aplikaciji postoje dvije opcije ako se želi generirati dokument, jedna je odgovaranje na sva generirana pitanja, a druga je puštanje asistenta da odgovori na ta pitanja kako on smatra da je najbolje. Na taj način se olakšava generiranje dokumenta korisnicima koji još nisu sigurni u sve detalje aplikacije ili onima koji žele testirati aplikaciju kako bi ustanovili je li ova aplikacija njima korisna.

Projects ▾ Rejio Welcome, admin!

1: What are the primary user roles and their respective capabilities in the online flight management system?

There should be a user class that can make new reservations or cancel existing reservations and view

2: Could you provide a brief overview of the ticket booking process for passengers?

The user selects the desired flight and chooses the type he wants (one-way, round-trip,..) and submit

3: Which specific features do you envision for the flight management functionality of the system?

There should be a database of all flights, airports, and passengers which can be managed by the administrator

4: Do you have any preferences for the user interface design of the application?

It should be easy to use and modern

5: Are there any specific security requirements that need to be considered for handling payment transactions on the platform?

The platform should be secure and provide payment types that are secure

6: What types of reports or analytics do you expect the system to generate for airline administrators or managers?

There are none at the moment, but there are plans to expand

7: Do you have any specific preferences for integrating third-party services (such as payment gateways or flight status APIs) into the application?

Yes

8: What are the expected peak usage times for the application, and how should it handle increased traffic during these periods?

The application should be designed in a way that can accommodate all customers

9: Are there any regulatory compliance requirements that the system needs to adhere to in terms of storing passenger data or handling bookings?

The data that it stored needs to be encrypted for the privacy of the customers other than that no

10: Do you have any specific preferences regarding the supported payment methods on the platform, such as credit cards, e-wallets, or other options?

The application should support payments with a credit card and bank transfers at the moment with

Let the Assistant decide

Reset

Answer

Sl. 4.8. Primjer asistentovih pitanja i odgovora na njih

Par trenutaka nakon što je korisnik odgovorio na pitanja asistent vraća dokument zahtjeva. Dio primjera takvog dokumenta je prikazan na slici 4.9.

Nakon generiranja dokumenta korisnici ga mogu uređivati na dva načina:

- Uređivanje uz pomoć asistenta
- Ručno uređivanje

Korisnik može uređivati dokument uz pomoć asistenta tako da upiše upit asistentu. Upit bi trebao sadržavati tekst koji govori asistentu što treba promijeniti ili dodati i na koji način. Nakon par trenutaka asistent će vratiti novi izmijenjenim dokument, korisnik onda može odabrati koji od ta dva dokumenta želi zadržati. Drugi način uređivanja je koristeći JSON uređivač. JSON uređivač je alat koji se može implementirati u web aplikacije koji pruža sučelje za lako uređivanja JSON objekta.

Oba načina uređivanja imaju svoje primjene. Uređivanje uz pomoć asistenta pruža jednostavan način za promjene ili dodavanje velike količine teksta. Isto tako ovaj se način može koristiti za promjenu ili dodavanje dijelova koji utječu na cijeli dokument. Ovaj način pruža manje kontrole nad dokumentom, dok ručno uređivanje pruža maksimalnu kontrolu nad dokumentom, iz tog razloga ga je bolje koristiti za promijene koje zahtijevaju veću preciznost.

Projects ▾ Reqjo Welcome, admin!

Flight manager

Edit document Using the Assistant

Delete

Introduction

Purpose

The purpose of this document is to outline the detailed requirements for the development of an online flight management system that allows passengers to purchase airline tickets. The document follows the IEEE 830 format to ensure clarity and completeness in specifying functional and non-functional requirements.

Document Conventions

The document follows the IEEE 830 standard for software requirements specification (SRS). All requirements are labeled with a corresponding number and functional/non-functional indicator.

Intended Audience

The intended audience for this document includes developers, project stakeholders, quality assurance teams, and any other parties involved in the development and implementation of the online flight management system.

Project Scope

The project aims to create a user-friendly online platform that enables passengers to search for flights, make reservations, and manage their bookings. The system will be underpinned by a relational database to support flight management and reservation functions.

Sl. 4.9. Primjer stranice za pregled dokumenta zahtjeva

5. EVALUACIJA GENERIRANE SPECIFIKACIJE ZAHTJEVA

Točnost dokumenata generiranog u prošlom odlomku se provjerava koristeći primjer dokumenta zahtjeva kojeg je napisao Ravi Bandakkanavar [13]. Ovaj test nije pokušaj dupliciranja rada nego proučavanje Bandakkanavar-ovog rada kako bi se izvukla osnovna ideja projekta te predaja tih informacija asistentu. Na taj se način dobiva usporedbu rada osobe koja ima više iskustva u polju inženjering zahtjeva i velikih jezičnih modela.

Zadatak projekta je napraviti online sustav za upravljanje letovima. Sustav treba omogućiti korisniku rezervaciju i praćenje letova, a administratoru upravljanje sustavom. Bitan dio sustava je relacijska baza podataka koja sadržava sve podatke o letovima i korisnicima. Generiranje aplikacije se postiže sa idućim upitom: „*I want to create an online flight management system. The application should be easy to use and allow passengers to buy airline tickets. System is based on a relational database with its flight management and reservation functions.*“ Nakon se toga odgovara na postavljena pitanja i kao rezultat se dobije dokument zahtjeva.

5.1. Usporedba dokumenta generiranog sa GTP-3.5 Turbo modelom s originalnim dokumentom

Dokument koji se uspoređuje s originalnim je izrađen koristeći GPT-3.5 Turbo model. Navedeni dokument će se uspoređivati na temelju poglavlja dokumenata, pošto su dokumenti utemeljeni na IEEE 830 standardu imati će sličnu strukturu i na taj način omogućiti lakšu usporedbu

Poglavlja koja se uspoređuju:

- Uvod
- Opći opis
 - Funkcije proizvoda
 - Značajke sustava
 - Zahtjevi za vanjsko sučelje
 - Nefunkcionalni zahtjevi

Uvod generiranog dokumenta i originalnog su gotovo jednaki. U tom se dijelu vidi snaga ove aplikacije, a to je praćenje forme dokumenta. Predajom forme dokumenta u upitu je osigurano što manje odstupanje od standarda inženjeringa zahtjeva. Razlika između originalnog i generiranog dokumenta se vidi u poglavlju opsega projekta gdje je cijeli projekt detaljnije objašnjen u

originalnom projektu kao što je prikazano na slici 5.1. Isto se može primijetiti da je opseg projekta generiranog dokumenta općenitiji.

Project Scope

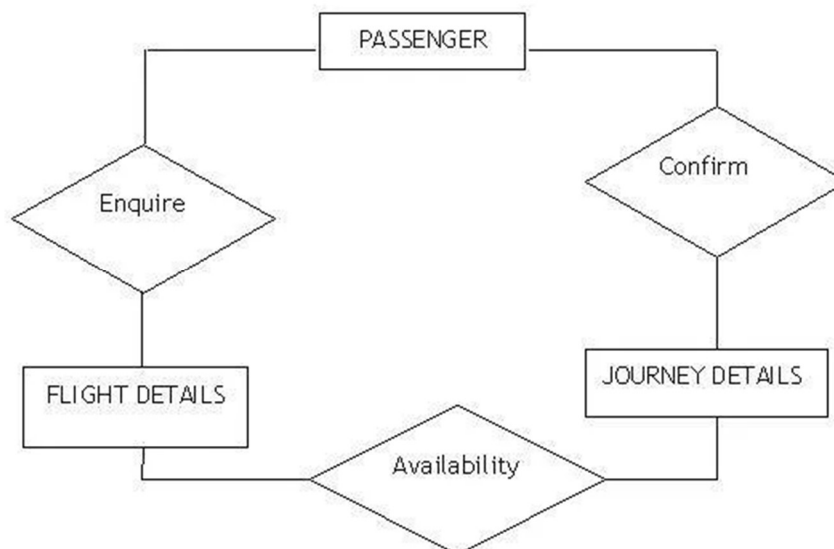
The project aims to create a user-friendly online platform that enables passengers to search for flights, make reservations, and manage their bookings. The system will be underpinned by a relational database to support flight management and reservation functions.

1.4 PROJECT SCOPE

The purpose of the online flight management system is to ease flight management and to create a convenient and easy-to-use application for passengers, trying to buy airline tickets. The system is based on a relational database with its flight management and reservation functions. We will have a database server supporting hundreds of major cities around the world as well as thousands of flights by various airline companies. Above all, we hope to provide a comfortable user experience along with the best pricing available.

Sl. 5.1. Usporedba opsega projekata

U poglavlju opći opis i ostalim poglavljima vidi se najveća razliku između ove dvije vrste izrade dokumenata, a to je mogućnost grafičkog prikaza podataka i ideja (slika 5.2.). Grafički prikazi omogućavaju lakše razumijevanje složenijih podataka i učinkovitiji prikaz veza i razlika između elemenata.



Sl. 5.2. Primjer grafičkog prikaza iz originalnog rada

Još jedno poglavlje u kojem je originalni dokument bolje napisan su funkcije proizvoda gdje je generirani dokument izostavio neke bitne funkcije kao što je načini pretrage letova, koja je prisutna u originalno dokumentu kao što je prikazano na slici 5.3. Do ovakvih pogrešaka dolazi zbog nedovoljno postavljenih pitanja korisniku. Naravno korisnik je mogao na neka pitanja odgovoriti detaljnije, ali proces inženjeringa zahtjeva inače traje duže što znači da inženjer ima vremena razmisliti o dodatnim funkcionalnostima i zahtjevima, dok je korisnik potaknut istog trenutka odgovoriti na sva pitanja pa se često neki elementi izostave. Ovaj problem bi se mogao riješiti sa

- **DESCRIPTION and PRIORITY**

The airline reservation system maintains information on flights, classes of seats, personal preferences, prices, and bookings. Of course, this project has a high priority because it is very difficult to travel across countries without prior reservations.

- **STIMULUS/RESPONSE SEQUENCES**

- Search for Airline Flights for two Travel cities
- Displays a detailed list of available flights and makes a "Reservation" or Books a ticket on a particular flight.
- Cancel an existing Reservation.

Sl. 5.3. Zahtjevi iz originalnog dokumenta

većim brojem detaljnijih pitanja, ali stariji OpenAI modeli su skloni generiranju samo određenog (vrlo često premalog) broja pitanja, što znači da što je više teksta generirano to je manja vjerojatnost da će model nastaviti generirati tekst zato što su ti modeli trenirani na kraćim dijelovima teksta (To je isto razlog zašto asistent staje sa generiranjem pitanja na brojevima između 5 i 10).

Loš učinak velikih jezičnih modela kod dužih tekstova i upita je poznat, iz tog razloga je u aplikaciju dodano naknadno komuniciranje sa asistentom koje omogućuje brzo dodavanje zahtjeva koje asistent nije generirao, ali se ta funkcionalnost oslanja na znanje korisnika što može otežati proces generiranja zahtjeva dokumenata. Još jedan način za osiguranje boljeg učinka kod generiranja dužih tekstova je upotreba jednog od novijih modela. U nastavku će biti prikazana evaluacija GPT-4o modela u generiranju dokumenata zahtjeva.

Još jedan bitno dio inženjeringa zahtjeva su nefunkcionalni zahtjevi koji su često zanemareni u tom cijelom procesu. Primjer usporedbe nefunkcionalnog zahtjeva iz originalnog i generiranog dokumenta prikazan je na slici 5.4. Taj specifičan nefunkcionalan zahtjev govori kakvo će biti korisničko iskustvo u aplikaciji. Iako su drukčije napisani oba zahtjeva jednako dobro opisuju taj zahtjev.

Usability: User interfaces should be easy to use, visually appealing, and responsive, catering to different devices and screen sizes to enhance user satisfaction and engagement with the application.

- **USABILITY:** The flight schedules should satisfy a maximum number of customers' needs. The application should be accessible on multiple devices, including desktop and mobile.

Sl. 5.4. Usporedba nefunkcionalnih zahtjeva

Na kraju se još mogu primijetiti prednosti aplikacije u usporedbi s standardnim inženjeringom zahtjeva. Upotreba aplikacije čini izradu i uređivanje dokumenata zahtjeva iznimno laganim. Aplikacija može koristiti osobama koje znaju malo o inženjeringu zahtjeva, ali iznenađujuće najviše koristi mogu naći osobe koje imaju nekog iskustva sa inženjeringom zahtjeva. Ovaj alat će

im omogućiti da lako generiraju većinski dio dokumenta, dok će im njihovo znanje omogućiti da ga nadopune. Još jedna prednost OpenAI modela jest poznavanje domene. Upoznavanje domene projekta jedan je od prvih koraka koje inženjer treba napraviti kako bi mogao krenuti sa pisanjem zahtjeva. Pošto su veliki jezični modeli trenirani na velikoj količini podataka iz svih područja imaju opsežno znanje svih domena koje bi mogle zanimati inženjera zahtjeva.

5.2 Usporedba dokumenta generiranog sa GTP-4o modelom s ostalim dokumentima

GPT-4o je OpenAI-ov najnoviji i najbolji model koji uz poboljšane mogućnosti pruža i bolju cijenu ispisa i upisa tokena od GPT-4 modela. Te poboljšane mogućnosti dolaze do izražaja u inženjeringu zahtjeva za sustav upravljanja letovima. Gdje je dokument generiran sa ovim modelom znatno kvalitetniji od onog koji je generiran sa GPT-3.5 Turbo modelom. Većina nedostataka GPT-3.5 Turbo modela kao što su premali broj postavljenih pitanja, nedovoljna količina zahtjeva su riješeni s ovim modelom. Sve prednosti nad originalnim dokumentom kao što su brzina, lakoća generiranja i poznavanje domene rada su još uvijek prisutne.

Jedan od nedostataka koji ova implementacija GPT-4o modela ne može ublažiti je manji broj opcija prikaza određenih elemenata i ideja što se u originalnom radu postiže kroz slike i dijagrame. Još je važno je napomenuti kako je GPT-4o model napisao i više zahtjeva nego što je traženo, a to su neki zahtjevi koji bi se mogli podrazumijevati s obzirom na ostale zahtjeve kao što su naprimjer zahtjevi za odabir sjedala prikazan na slici 5.5. Takav način generiranja može biti poželjan za korisnike koji nisu sigurni u sve zahtjeve koji će im trebati, ali isto tako može biti nepoželjan za korisnike koji žele veću razinu kontrole nad ispisom asistenta.

Seat Selection

Description: Allows passengers to select their preferred seats during the booking process.

Functional Requirements:

F10

The system shall display a seat map with available and occupied seats.

F11

The system shall enable users to select their preferred seats before finalizing the booking.

F12

The system shall include the seat information in the booking confirmation.

Sl. 5.5. Primjer zahtjeva vezanih uz odabir sjedala

Jedna bitna stavka u kojoj je GPT-4o model bolji od GPT-3.5 Turbo modela je kontekstni prozor. Kontekstni prozor GPT-3.5 Turbo modela je 16,385 tokena, dok GPT-4o model ima kontekstni prozor veličine 128,000 tokena. To znači da će GPT-4o biti točniji kod generiranja opširnijih dokumenata što vidimo na samom početku kreiranja novog projekta u aplikaciji. Kao što je već navedeno kad kreiramo novi projekt sa asistentom koji koristi GPT-3.5 Turbo model on generira manji broj pitanja (najčešće između 5 i 10) i dokumenti koje generira su nedovoljno opširni. Asistent koji koristi GPT-4o projekt generira veći broj pitanja (preko 20) i sami dokumenti zahtjeva su puno opširniji.

6. ZALJUČAK

Zadatak završnog rada bio je istražiti mogu li se koristiti veliki jezični modeli za generiranje kvalitetnih dokumenata zahtjeva te može li se područje inženjeringa zahtjeva približiti svim developerima. Prednosti inženjeringa zahtjeva su ogromne, dok nedovoljno dobri zahtjevi često vode do propasti projekta. Za izvršavanje tog zadatka razvijena je web aplikacija koja korisniku omogućuje generiranje dokumenata zahtjeva koristeći OpenAI asistenta. Korisnik isto tako može uređivati svoje dokumente ručno ili uz pomoć asistenta.

Proveden je eksperiment nad dokumentima koje je aplikacija generirala tako što su se uspoređivali sa dokumentom osobe koja je iskusnija u području pisanja takvih dokumenata u nadi da će rezultati pokazati može li se aplikacija koristiti u stvarnim primjenama. Rezultati su pokazali kako se aplikacija može koristiti u svrhe inženjeringa zahtjeva, dok god je korisnik svjestan ograničenja modela koji generiraju dokument. Isto tako su se međusobno uspoređivali dokumenti generirani s različitim modelima gdje se moglo vidjeti da je noviji OpenAI model detaljniji i bolji kod generiranja dokumenata zahtjeva.

Ovo je još jedan od načina upotrebe umjetne inteligencije gdje se pruža osobama sa manje znanja niža prepreka za kreiranje softvera i učenja dobrih praksi pri samom kreiranju. Također se treba naglasiti korisnost ove aplikacija onima koji su iskusni u inženjeringu zahtjeva, a žele ubrzati proces pisanja samog dokumenta.

Ovu aplikaciju je moguće nadograditi sa funkcionalnostima koje će potaknuti međusobni rad više developera. To se može postići sa mogućnosti dijeljena projekta ili mogućnosti kreiranja zajedničkog projekta. Još jedna nadogradnja koja se može implementirati je generiranje slika i dijagrama uz pomoć DALL-E modela. DALL-E je OpenAI model treniran za generiranje slika koji možemo implementirati u kombinaciji sa ostalim OpenAI modelima.

LITERATURA

- [1] H. F., Hofmann, F., Lehner, „Requirements engineering as a success factor in software projects“, *IEEE Softw.*, izd. 4, sv. 18, str. 58–66, srp. 2001.
- [2] M., Ali Ramdhani, D., Sa’adillah Maylawati, A., Syakur Amin, H., Aulawi, „Requirements Elicitation in Software Engineering“, *IJET*, izd. 2.29, sv. 7, str. 772, svi. 2018.
- [3] P. A., Laplante, *Requirements Engineering for Software and Systems, Third Edition*. Auerbach Publications, 2017.
- [4] A., Aurum, C., Wohlin, „The Fundamental Nature of Requirements Engineering Activities as a Decision Making Process. Information and Software Technology 45, 945-954“, *Information and Software Technology*, sv. 45, str. 945–954, stu. 2003.
- [5] R., Elliott, E. B., Allen, „A methodology for creating an IEEE standard 830-1998 software requirements specification document“, *Journal of Computing Sciences in Colleges*, sv. 29, str. 123–131, 2013.
- [6] „IEEE Recommended Practice for Software Requirements Specifications“, *IEEE Std 830-1998*, str. 1–40, lis. 1998.
- [7] M. R., Douglas, „Large Language Models“. arXiv, 06-lis-2023.
- [8] S. M., Kerner, „What are Large Language Models? | Definition from TechTarget“ [online]. Dostupno na: <https://www.techtarget.com/whatis/definition/large-language-model-LLM>. [Pristupljeno: 10.6.2024.].
- [9] A., Vaswani *i ostali*, „Attention Is All You Need“. arXiv, 01-kol-2023.
- [10] „Product Tour - aqua cloud“ [online].
- [11] F., Redmond, „Project management of an online Change Laboratory using Notion“, *Bureau de Change Laboratory*, 2023.
- [12] „Get started with Tara AI | Features, workflows & integrations“ [online]. Dostupno na: <https://tara.ai/how-it-works/>. [Pristupljeno: 5.6.2024.].
- [13] „Engineering Requirements Management DOORS“ [online], 03-ožu-2023. Dostupno na: <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=overview-doors>. [Pristupljeno: 5.6.2024.].
- [14] „WriteMyPrd | Make Writing PRDs a Breeze with ChatGPT“ [online]. Dostupno na: <https://writemyprd.com/>. [Pristupljeno: 5.6.2024.].
- [15] R., Bandakkanavar, „Software Requirements Specification document with example“ [online], 10-sij-2023.
- [16] C., Han *i ostali*, „LM-Infinite: Zero-Shot Extreme Length Generalization for Large Language Models“. arXiv, 24-lip-2024.

SAŽETAK RADA

Naslov: Primjena OpenAI modela u procesu inženjeringa zahtjeva za web aplikacije

Sažetak: Ovaj rad se bavi istraživanjem uspješnosti generiranja kvalitetnih dokumenata zahtjeva korištenjem velikih jezičnih modela. Cilj je bio procijeniti može li se proces inženjeringa zahtjeva olakšati implementacijom OpenAI modela u kreiranju web aplikacija. Kroz razvijenu web aplikaciju koja koristi OpenAI asistenta, omogućeno je generiranje i uređivanje dokumenata zahtjeva na brz i jednostavan način. Evaluacija je provedena usporedbom dokumenata generiranih različitim OpenAI modelima s dokumentom iskusnog inženjera zahtjeva. Rezultati pokazuju da se aplikacija može koristiti za inženjering zahtjeva, uz svjesnost korisnika o ograničenjima modela. Ova aplikacija može olakšati proces kreiranja softvera osobama s manje iskustva, ali i ubrzati rad iskusnih inženjera.

Ključne riječi: inženjering upita, inženjering zahtjeva, OpenAI API, OpenAI asistent, veliki jezični modeli

ABSTRACT

Title: Use of OpenAI models in the process of engineering requirements for web applications

Abstract: This paper investigates the effectiveness of generating quality requirement documents using large language models. The goal was to evaluate whether the requirements engineering process can be made easier by implementing OpenAI models in creating web applications. Through a developed web application utilizing the OpenAI assistant, it enables quick and easy generation and editing of requirement documents. Evaluation was conducted by comparing documents generated with different OpenAI models with a document created by an experienced requirements engineer. The results show that the application can be used for requirements engineering, with users being aware of the model's limitations. This application can facilitate the software creation process for less experienced individuals and speed up the work of experienced engineers.

Keywords: prompt engineering, requirements engineering, OpenAI API, OpenAI assistant, large language models