UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Katarina Milošević

SISTEM ZA AUTOMATSKO GENERISANJE PREZENTACIJA ZASNOVAN NA VELIKIM JEZIČKIM MODELIMA

master rad

Beograd, 2026.

Mentor:

dr Aleksandar Kartelj, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Mladen Nikolić, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Staša Vujičić Stanković, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Naslov master rada: Sistem za automatsko generisanje prezentacija zasnovan na velikim jezičkim modelima

Rezime: Izrada poslovne prezentacije spaja dva zadatka koja se retko rešavaju istim alatom: sažimanje sadržaja i poštovanje vizuelnog identiteta organizacije. Rad opisuje sistem *SlideFlow*, koji iz kratkog opisa teme generiše prezentaciju u formatu PPTX preuzimajući vizuelni jezik iz postojećeg standardizovanog šablona prezentacije. Sistem je realizovan kao mikroservisna arhitektura sa tri servisa i deli rad na dve faze. U *offline* fazi šablon se rastavlja na pojedinačne slajdove; za svaki se iscrtava slika, generiše strukturirani opis namene pomoću multimodalnog jezičkog modela i izračunava vektorska reprezentacija za pretragu. U *online* fazi jezički model pravi plan sadržaja po slajdovima, semantičkom pretragom i rerangiranjem biraju se najpogodniji šablonski slajdovi, a agentski sloj zasnovan na alatu Codex CLI za svaki slajd generiše više varijanti pisanjem i izvršavanjem koda nad bibliotekom python-pptx, uz automatsku proveru izlaza i ponovne pokušaje. Rad sadrži pregled 19 poslova generisanja izvedenih sa četiri šablona i sedam tema, kao i detaljnu analizu jednog potpunog prolaska kroz sistem — pet opisanih slajdova ulaznog šablona i 15 generisanih varijanti u pet poslova, sa srednjim vremenom generisanja od 256 sekundi po varijanti i sklopljenom prezentacijom od pet slajdova — kao i kvalitativnu analizu svojstava generisanog rasporeda i pravaca daljeg razvoja.

Ključne reči: veliki jezički modeli, automatsko generisanje prezentacija, generisanje koda, agentski sistemi, semantičko pretraživanje, vektorske reprezentacije, Office Open XML, mikroservisi

Sadržaj

Glava 1

Uvod

Izrada prezentacije spaja dva zadatka koja se retko rešavaju istim alatom: sažimanje sadržaja u niz poruka i poštovanje vizuelnog identiteta organizacije. Standardizovani šabloni prezentacija po pravilu sadrže desetine unapred pripremljenih rasporeda (naslovne slajdove, slajdove za poređenje, vremenske ose, prikaze ključnih pokazatelja) i autor mora da prepozna koji od njih odgovara sadržaju, a zatim tekst ručno uklopi u zatečene okvire.

Veliki jezički modeli menjaju ovaj proces jer uspešno sažimaju i strukturiraju tekst i, što je za ovaj rad presudno, generišu kod koji se može neposredno izvršiti, oslanjajući se na postojeće biblioteke [2, 4]. Put od teksta do upotrebljive PPTX datoteke ipak nije trivijalan: format PPTX nije jedna jednostavna datoteka, već paket međusobno povezanih XML dokumenata i pratećih datoteka [7]. Zbog toga se vizuelna ispravnost slajda (odsustvo preklapanja, čitljivost, poštovanje margina) ne može proceniti samo iz teksta koji se na njemu nalazi.

1.1 Motivacija

Komercijalni alati za automatsko generisanje prezentacija po pravilu nude sopstvene, generičke šablone. Za organizacije koje primenjuju jasno definisane smernice vizuelnog identiteta, to znači da generisani rezultat mora naknadno ručno da se prilagodi, čime se gubi najveći deo vremenske uštede. Akademska istraživanja u ovoj oblasti dugo su bila usmerena na kvalitet sadržaja, dok su vizuelni izgled i strukturna koherentnost ostajali u drugom planu [32].

Problem koji ovaj rad rešava jeste kako iz kratkog opisa teme automatski proizvesti prezentaciju čiji je sadržaj relevantan, a vizuelni jezik usklađen sa konkret-

nim, već postojećim standardizovanim šablonom prezentacije. Dodatni argument za programski pristup, u kome se generiše kod koji gradi slajd umesto slike slajda, jeste to što se dobijeni rezultat može dalje menjati u programu PowerPoint, pa se takvo rešenje prirodno uklapa u postojeći tok rada korisnika, jer sistem generiše nacrt koji korisnik potom dorađuje.

Predmet ovog rada je sistem koji iz opisa teme, ciljne publike, tona izlaganja i broja slajdova generiše prezentaciju u formatu PPTX, koristeći slajdove postojećeg standardizovanog šablona kao vizuelnu referencu; sistem je nazvan *SlideFlow*. Rad postavljeni problem najpre formalizuje i smešta u odnos prema postojećim pristupima, a zatim projektuje arhitekturu koja pripremu baze šablonskih slajdova razdvaja od generisanja prezentacije na zahtev. U toj arhitekturi realizovana su tri koraka od kojih rešenje zavisi: semantičko opisivanje i vektorsko indeksiranje šablonskih slajdova pomoću multimodalnog jezičkog modela, izbor šablonskih slajdova koji vektorsku pretragu dopunjuje rerangiranjem po vizuelnim kriterijumima, i agentski sloj koji generisanjem i izvršavanjem programskog koda proizvodi više varijanti svakog slajda, uz proveru rezultata i oporavak od grešaka. Ponašanje tako izgrađenog sistema analizirano je na materijalu prikupljenom tokom njegovog izvršavanja, uz pregled svih izvedenih prolazaka i detaljan prikaz jednog od njih. Poređenje sa drugim sistemima na standardizovanim skupovima podataka nije predmet rada; opseg analize obrazložen je u glavi 7.

1.2 Struktura rada

Glava 2 izlaže pojmove, tehnike i formate na kojima rešenje počiva. Glava 3 pregleda postojeće pristupe automatskom generisanju prezentacija. Glava 4 opisuje problem i izvodi funkcionalne i nefunkcionalne zahteve. Glava 5 opisuje arhitekturu sistema, a glava 6 njenu realizaciju, po fazama pripreme baze šablonskih slajdova i generisanja prezentacije na zahtev. Glava 7 donosi eksperimentalnu analizu, a glava 8 zaključak i pravce daljeg razvoja.

Glava 2

Relevantni koncepti

Ova glava izlaže pojmove, tehnike i formate na kojima rešenje opisano u radu počiva, i za svaki od njih odmah navodi kako se u sistemu koristi.

2.1 Veliki jezički modeli i strukturirani izlaz

Veliki jezički modeli su neuronske mreže obučene na velikim količinama teksta, koje zadatak rešavaju tako što nastavljaju zadati niz. Isti model, bez dodatnog obučavanja, sažima tekst, prevodi ga, razvrstava i piše programski kod, pa se u sistemima poput ovog koristi kao opšta komponenta kojoj se zadatak saopštava opisom. Sistem opisan u radu oslanja se na model na četiri mesta: pri sastavljanju plana prezentacije, pri opisivanju šablonskih slajdova, pri izboru kandidata i pri generisanju koda koji gradi slajd. Svojstva modela zato neposredno određuju i šta se od sistema može očekivati i koje provere u njemu moraju postojati; da bi se ta svojstva izvela, a ne samo nabrojala, potrebna je preciznija definicija.

Neka je V konačan skup tokena, dobijen potpodelom teksta na česte podniske (engl. *subword tokenization*), i neka je $x = (x_1, \ldots, x_n)$, $x_t \in V$, niz tokena. Veliki jezički model je parametrizovana raspodela p_θ nad takvim nizovima, faktorisana lančanim pravilom

$$p_\theta(x) = \prod_{t=1}^{n} p_\theta(x_t \mid x_{<t}), \qquad x_{<t} = (x_1, \ldots, x_{t-1}), \tag{2.1}$$

pri čemu se svaka uslovna raspodela dobija primenom softmaks preslikavanja na vektor logita koji mreža pridružuje kontekstu $x_{<t}$. Generisanje je uzastopno uzorkovanje iz raspodela $p_\theta(\cdot \mid x_{<t})$, gde se svaki dobijeni token pridružuje kontekstu

narednog koraka. Odatle slede dve posledice bitne za ovaj rad: model ne razlikuje zadatak od podataka, jer su uputstvo, ulazni podaci i dotadašnji izlaz tokeni istog niza, a izlaz nije funkcija ulaza nego slučajna promenljiva sa raspodelom uslovljenom ulazom.

Uslovne raspodele računa transformerska mreža, čija je osnovna operacija mehanizam pažnje. Ako su $Q \in \mathbb{R}^{n\times d_k}$, $K \in \mathbb{R}^{n\times d_k}$ i $W \in \mathbb{R}^{n\times d_v}$ matrice upita, ključeva i vrednosti, dobijene linearnim preslikavanjima ulaznih reprezentacija, pažnja se definiše kao

$$\text{Attention}(Q, K, W) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) W, \tag{2.2}$$

čime se svaka pozicija predstavlja kao težinska kombinacija svih pozicija u nizu, sa težinama koje zavise od sadržaja, a ne od međusobnog rastojanja [29]. Tri posledice te definicije bitne su za rad. Pozicije se u sloju obrađuju nezavisno, pa obučavanje može da se paralelizuje po dužini niza, što je i omogućilo modele opšte namene obučene na korpusima reda veličine 10^{11} tokena. Put između bilo koje dve pozicije konstantne je dužine, pa model povezuje udaljene delove ulaza, na šta se sistem oslanja jer upiti koje sastavlja prelaze hiljadu tokena, a ograničenja sa početka upita moraju biti dovedena u vezu sa podacima na njegovom kraju. Najzad, matrica $QK^\top$ ima n^2 elemenata, pa jedan sloj zahteva $O(n^2 d)$ operacija za niz dužine n i dimenziju reprezentacije d; odatle formalno slede ograničena dužina konteksta i rast trajanja i cene poziva sa dužinom upita.

Obučavanje teče u dve faze: parametri θ najpre se određuju maksimizacijom log-verodostojnosti korpusa, a zatim se model doteruje da prati uputstva, nadgledano na demonstracijama traženog ponašanja i optimizacijom prema modelu nagrade naučenom iz ljudskih poređenja odgovora [22]. Druga faza objašnjava zašto se ponašanje modela u sistemu uređuje isključivo sadržajem upita, ali i određuje domet takvog upravljanja: praćenje uputstava je statističko svojstvo naučeno iz podataka, a ne invarijanta postupka, pa se poštovanje zadatih ograničenja mora proveriti nad izlazom (odeljak 6.3.4). U ovom radu nijedan model nije dodatno obučavan; koriste se gotovi modeli opšte namene, pa je sadržaj upita jedino sredstvo uticaja na izlaz. Na istoj sposobnosti počiva i učenje iz konteksta, kod koga se zadatak zadaje opisom, po potrebi i sa nekoliko rešenih primera, bez izmene parametara θ [2]. Uspešnost na zadacima koji zahtevaju izvođenje u više koraka raste ako se od modela traži da te korake eksplicitno ispiše, čime međurezultati ulaze u kontekst narednih koraka [30], a i sam oblik upita merljivo utiče na rezultat [14];

upiti u sistemu zato prate jedinstven obrazac: uloga, ograničenja postupka, ulazni podaci i tražena struktura odgovora.

Multimodalni modeli isti postupak primenjuju i na slike: slika se kodira u niz vektorskih reprezentacija koje ulaze u kontekst uporedo sa tokenima teksta i troše isti budžet dužine. To je za rad presudno, jer deo vizuelnog identiteta slajda nije zapisan u njegovom sopstvenom XML-u (odeljak 2.5), pa se slajd modelu prilaže i kao slika. Pošto je kontekst ograničen, sistem sve što šalje prethodno sažima: iz XML zapisa uzima odsečak od najviše 4000 znakova (odeljak 6.2.2), a strukturni opis kandidata svodi na najviše dva slajda, osam rezervisanih mesta i četrnaest oblika (odeljak 6.3.3).

Izbor narednog tokena uređuje se parametrima uzorkovanja. Ako je z vektor logita, temperatura $T > 0$ daje raspodelu

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{2.3}$$

pa niže vrednosti T zaoštravaju raspodelu ka najverovatnijem tokenu, a više je izravnavaju; granični slučaj $T \to 0^+$ odgovara pohlepnom dekodiranju, odnosno izboru $\arg\max_i z_i$. Ni tada ponovljivost nije garantovana: u uslugama koje modele izvršavaju paralelizovano i uz grupisanje zahteva redosled sabiranja u aritmetici sa pokretnim zarezom nije fiksiran, pa neznatne razlike u logitima mogu promeniti izbor tokena i, kroz faktorizaciju (2.1), ceo dalji niz. Nedeterminizam se stoga na strani klijenta ne može ukloniti. Iz njega slede dve posledice na koje se sistem oslanja: ponovni pokušaj posle neuspelog generisanja ima nezanemarljivu verovatnoću drugačijeg ishoda, što je pretpostavka mehanizma ponavljanja (odeljak 6.3.4), i nijedan izlaz se ne može smatrati ispravnim po konstrukciji, nego se svaki proverava pre nego što bude ponuđen korisniku (odeljak 6.3.4). Razlike među varijantama istog slajda posledica su tog svojstva, a ne njegova svrha.

Kriterijum obučavanja je verodostojnost niza, a ne istinitost tvrdnji, pa model može proizvesti sadržaj koji je jezički uverljiv, a neutemeljen u ulazu ili u činjenicama [12]. U ovom sistemu posledica bi bila merljiva: opis šablonskog slajda sa izmišljenim svojstvima ušao bi u vektorsku reprezentaciju i pomerio izbor kandidata, pa se u upitu za opisivanje modelu izričito nalaže da tvrdnje izvodi isključivo iz XML zapisa i slike slajda. Poziv modelu je, uz to, operacija čije trajanje raste sa brojem generisanih tokena, a naplaćuje se po broju tokena ulaza i izlaza; za postupak koji za svaki slajd traži nekoliko varijanti, a za svaku pokreće agenta koji model poziva više puta, ti pozivi čine glavninu utrošenog vremena (odeljak 7.5.2).

Kada se model koristi kao komponenta sistema, njegov izlaz mora biti strukturiran u obliku pogodnom za automatsku obradu, po pravilu u formatu JSON. U upotrebi su dva mehanizma: ograničeno dekodiranje, kod koga se u svakom koraku maskiraju tokeni koji bi narušili zadatu shemu, i zadavanje sheme u upitu uz naknadnu proveru odgovora. Sistem koristi drugi: shema se zadaje u upitu, odgovor se raščlanjuje i proverava prema modelu definisanom bibliotekom Pydantic [24]. Sintaksna ispravnost, odnosno to da je izlaz ispravan JSON, ne povlači semantičku, odnosno prisustvo svih očekivanih polja odgovarajućih tipova, pa je provera po shemi zaseban korak, prisutan svuda gde sistem od modela traži strukturiran odgovor.

2.2 Vektorske reprezentacije i pretraga

Vektorska reprezentacija (engl. *embedding*) preslikava tekst u vektor takav da semantički slični tekstovi budu blizu u dobijenom prostoru. Takav prostor se ne dobija ručno već obučavanjem: model se trenira na parovima ili trojkama tekstova (na primer, dve rečenice sa istim značenjem naspram dve nepovezane) tako da rastojanje između njihovih vektora odražava njihovu stvarnu semantičku bliskost, čime dva teksta mogu biti prepoznata kao slična i kada ne dele nijednu zajedničku reč [26]. Upravo je to svojstvo neophodno sistemu opisanom u ovom radu: opis šablonskog slajda i opis sadržaja koji se generiše retko koriste isti rečnik, pa bi pretraga zasnovana na podudaranju reči (na primer, mera BM25) često promašila relevantan slajd. Sličnost dva vektora $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ meri se kosinusom ugla između njih:

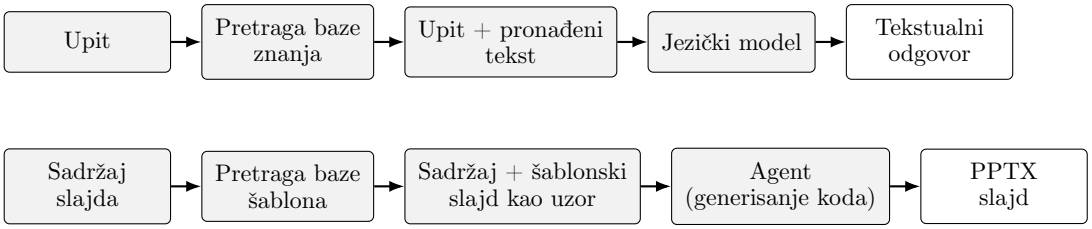
$$\mathrm{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}. \tag{2.4}$$

Vrednost pripada intervalu $[-1, 1]$; veća vrednost znači veću sličnost. Mera je neosetljiva na dužinu vektora, pa je standardna u pretraživanju informacija [16] i koristi se u sistemu za poređenje opisa šablonskog slajda sa opisom traženog sadržaja (odeljak 6.3.2). Modeli za vektorske reprezentacije proizvode vektor unapred određene dimenzije, uz mogućnost da se izlaz skrati na manju dimenziju bez značajnog gubitka tačnosti [21]. Sistem tu mogućnost i koristi: model čija je izvorna dimenzija 3072 poziva se sa zahtevanom dimenzijom 1536. Veća dimenzija ovde ne bi donela korist jer baza šablonskih slajdova ima red veličine desetina zapisa.

2.2.1 Generisanje potpomognuto pretragom

Jezički model svoje znanje nosi u parametrima fiksiranim u trenutku obučavanja; ne može da odgovori na osnovu podataka kojih tada nije bilo, a sadržaj koji izmišlja teško je razlikovati od sadržaja koji zna (odeljak 2.1). Kao rešenje uvedeno je generisanje potpomognuto pretragom (engl. *Retrieval-Augmented Generation*, RAG): pre generisanja odgovora, sistem iz spoljne baze pronalazi dokumente relevantne za upit i prosleđuje ih modelu kao kontekst, čime se odgovor oslanja na proverljiv, ažuriran izvor umesto isključivo na naučeno znanje [13]. Klasičan tok prikazan je u gornjem redu slike 2.1.

Sistem opisan u ovom radu prati istu strukturu: upit, pretraga, kontekst, generisanje, ali sa bitnom razlikom u prirodi pronađenog materijala, prikazanom u donjem redu slike 2.1: iz baze se ne dobavlja tekstualno znanje o temi, već vizuelni i strukturni obrazac, odnosno šablonski slajd koji služi kao uzor za oblikovanje, dok tekstualni sadržaj potiče iz plana koji je model prethodno sastavio, a ne iz pronađenih dokumenata. Preciznije rečeno, sistem ne dopunjuje *znanje* modela, već mu ograničava *prostor rešenja* pri generisanju koda na rasporede koji su već poznati da funkcionišu.



Slika 2.1: Klasičan tok generisanja potpomognutog pretragom (gore) i njegova varijanta u opisanom sistemu (dole): pronađeni materijal je vizuelni obrazac, ne tekstualno znanje, a generisani izlaz je izvršni kod, ne tekst.

2.2.2 Rerangiranje pomoću jezičkog modela

Pretraga po vektorskoj sličnosti brzo sužava skup kandidata, ali ne uzima u obzir kriterijume koji se teško izražavaju jednim vektorom, na primer da li će se konkretan sadržaj uklopiti u raspored bez prelivanja teksta. Za takve kriterijume koristi se drugi stepen: jezički model koji kandidate poredi detaljnije, uz pristup uporediv sa nadgledano obučenim modelima [27]. U sistemu ovaj drugi stepen dobija i sliku svakog kandidata, čime se u odluku uključuje kvalitet vizuelne hi-

jerarhije i mogućnost popunjavanja rasporeda (odeljak 6.3.2), tj. kriterijumi koje sama vektorska sličnost ne može da proceni.

2.3 Agentski pristupi i generisanje koda

Pojedinačan poziv jezičkog modela proizvodi tekst i time se okončava: model nema načina da utvrdi da li je ono što je napisao upotrebljivo. Kod zadataka čija se ispravnost pokazuje tek izvršavanjem, a generisanje PPTX datoteke je takav zadatak jer se greška u programu vidi tek kada se program pokrene, to nije dovoljno; potreban je postupak u kome model može da deluje na okruženje i da vidi ishod sopstvenog postupka. Takvi postupci nazivaju se agentskim: modelu se, uz cilj, daju i alati kojima menja stanje okruženja, a odluka o tome koji će korak izvesti prepušta se njemu. Da bi se videlo šta se time dobija, a šta gubi, postupak treba opisati preciznije.

Agentski pristup se formalno opisuje kao interakcija modela sa okruženjem u diskretnim koracima. Neka je $\mathcal{A}$ skup dopuštenih akcija, gde je svaka akcija poziv alata sa argumentima, i neka je $\mathcal{O}$ skup opažanja koja okruženje vraća. Neka je h_0 početno stanje, sastavljeno od cilja i ograničenja postupka, i neka je $h_t = h_{t-1} \oplus (a_t, o_t)$ istorija posle t koraka, pri čemu $\oplus$ označava nadovezivanje serijalizovanog para akcije i opažanja. Postupak se u koraku t odvija kao

$$a_t \sim \pi_\theta(\cdot \mid h_{t-1}), \qquad o_t = E(a_t), \tag{2.5}$$

gde je E okruženje, a politiku π_θ ostvaruje jezički model uslovljen serijalizovanom istorijom, u smislu izraza (2.1). Petlja se prekida kada model izabere akciju zaustavljanja ili kada se dostigne najveći dopušteni broj koraka $k_{\max}$.

Suštinu pristupa određuje to ko bira niz $(a_1, \ldots, a_k)$. U postupku obrade sa unapred utvrđenim redosledom poziva taj niz je zadat programom, a model popunjava pojedinačne korake; prema (2.5), međutim, niz akcija je slučajan, uslovljen ishodima prethodnih koraka, pa ni njegova dužina ni redosled nisu unapred određeni. Odatle i tri svojstva po kojima se agentski sistem razlikuje od pojedinačnog poziva modela: raspolaganje alatima, povratna sprega sa okruženjem koje vraća stvarne, a ne predviđene ishode, i prenos upravljanja sa programa na model.

Skup $\mathcal{A}$ zadaje se modelu opisom svakog alata: imenom, namenom i tipovima argumenata. Poziv alata je zato poseban slučaj strukturiranog izlaza (odeljak 2.1): model proizvodi zapis koji izvršno okruženje raščlanjuje i sprovodi, a rezultat vra-

ća kao opažanje o_t. Time se u postupak uvodi izvor informacije koji ne potiče od modela, što je i razlog zbog kojeg agentski postupak može da opovrgne sopstvenu pogrešnu pretpostavku. U okviru pristupa nazvanog ReAct pokazano je da naizmenično smenjivanje koraka izvođenja i akcija smanjuje broj grešaka u odnosu na postupak bez interakcije sa okruženjem, upravo zato što se pogrešna pretpostavka ispravlja čim je akcija demantuje, umesto da bude prenesena kroz ostatak zadatka [31].

Model između poziva ne zadržava sopstveno stanje, pa se istorija h_t u svakom koraku prosleđuje iznova. Dužina serijalizovane istorije raste sa brojem koraka, a ograničena je kontekstnim prozorom, pa broj koraka koje jedan zadatak može da obuhvati nije proizvoljan; uz to, prema (2.2), cena koraka raste kvadratno sa dužinom konteksta. Duži zadaci se zato oslanjaju na spoljnu memoriju, najčešće na datoteke i sažetke ranijih koraka, kojima agent pristupa akcijama čitanja i upisa, čime deo istorije izlazi iz konteksta i u njega se vraća samo po potrebi. U sistemu opisanom u radu tu ulogu ima radni direktorijum varijante, u kome ostaju generisani kod, ulazne datoteke i međurezultati (odeljak 6.3.3).

Agentski sistemi se razlikuju po stepenu samostalnosti i po broju učesnika. Na jednom kraju su sistemi u kojima model u svakom koraku bira akciju iz malog, unapred zadatog skupa, a na drugom oni u kojima sam razlaže cilj na podzadatke i određuje njihov redosled. Nezavisno od toga, zadatak može voditi jedan agent ili više njih, sa podeljenim ulogama i razmenom poruka. Sistem opisan u ovom radu koristi jednog agenta po varijanti slajda, kome je cilj zadat u celini, a razlaganje na korake i izbor akcija prepušteni su njemu.

Za ovaj rad poseban značaj ima slučaj u kome je akcija pisanje i izvršavanje programskog koda. Modeli obučeni i na korpusima izvornog koda proizvode funkcionalno ispravne programe iz opisa na prirodnom jeziku [4], a izvršavanje programa daje opažanje koje se ne dobija procenom nego merenjem: program se ili izvrši, ili prekine uz poruku o grešci. Postupak popravljanja tada ima oblik iterativnog preslikavanja $c_{i+1} = M(c_i, e_i)$, gde je c_i program iz i-tog pokušaja, e_i poruka dobijena njegovim izvršavanjem, a M model; takvo iterativno doterivanje merljivo povećava udeo ispravnih rešenja [5, 15]. U sistemu opisanom u radu obrazac je primenjen doslovno: agent piše program, program se izvršava, a poruka o grešci ulazi u naredni pokušaj kao deo upita.

Prenos upravljanja na model ima i cenu. Pošto je dužina niza akcija slučajna, trajanje i trošak zadatka nisu unapred poznati, a svaka akcija menja stvarno sta-

nje okruženja, pa posledice greške ostaju u datotekama, procesima ili pozvanim uslugama. Agentski postupak se zato ograničava spolja, brojem dopuštenih pokušaja, vremenskim ograničenjem izvršavanja i opsegom pristupa koji se agentu dodeljuje. Sistem opisan u radu koristi sva tri ograničenja: poziv agenta prekida se posle 600 sekundi, varijanta se generiše u najviše tri pokušaja (odeljak 6.3.4), a svakoj varijanti dodeljuje se zaseban radni direktorijum (odeljak 6.3.3).

Neuspeh agentskog postupka ne mora biti pogrešna pojedinačna akcija. Pošto se svaka odluka donosi uslovljavanjem na sopstvenu istoriju, mogući su i ishodi svojstveni samoj petlji: ponavljanje iste neuspešne akcije bez napretka, postepeno udaljavanje od zadatog cilja ka srodnom, i zaustavljanje pre nego što su svi zahtevi ispunjeni. Nijedan od njih se ne otkriva iz agentove ocene sopstvenog rada, pa se ishod proverava spolja, merilom nezavisnim od modela (odeljak 6.3.4).

Generisanje slajda u sistemu realizuje se uz pomoć alata Codex CLI, koji se pokreće lokalno, pristupa datotekama u radnom direktorijumu, raspolaže komandnom linijom i podržava neinteraktivni režim (`codex exec`) pogodan za ugradnju u automatizovan tok [20]. Time se generisanje jednog slajda poverava agentu kao zadatak sa jasno određenim izlazom, datotekom na zadatoj putanji, čija se ispravnost proverava, a u slučaju neuspeha agentu se vraća poruka o grešci za naredni pokušaj (odeljak 6.3.4).

2.4 Evaluacija generisanog sadržaja

Klasične metrike zasnovane na poređenju sa referentnim tekstom (na primer, preklapanje reči sa unapred pripremljenim „tačnim” odgovorom) pretpostavljaju da postoji jedan ispravan izlaz, što ne važi za generativne zadatke sa više podjednako prihvatljivih rešenja, a generisanje slajda je upravo takav zadatak, pošto različiti rasporedi mogu podjednako dobro preneti isti sadržaj. Takve metrike zato slabo koreliraju sa ljudskom procenom kvaliteta. Alternativa koja se poslednjih godina široko koristi jeste *LLM-as-a-judge*: jači jezički model u ulozi ocenjivača, kome se izlaz prosleđuje uz kriterijume ocenjivanja, slično rerangiranju iz odeljka 2.2.2, samo nad gotovim rezultatom umesto nad kandidatima za dalju obradu. Takve ocene se u velikoj meri poklapaju sa ljudskim preferencijama, ali su dokumentovane i sistematske pristrasnosti: prema poziciji odgovora, dužini teksta i sopstvenim generacijama modela [33], koje treba uzeti u obzir pri projektovanju rubrike za ocenjivanje.

Kod prezentacija je evaluacija dodatno složena jer kvalitet ima najmanje tri dimenzije koje se delimično sudaraju: sadržaj (da li je tekst tačan i relevantan), vizuelno oblikovanje (raspored, hijerarhija, čitljivost) i koherentnost celine (da li slajdovi zajedno čine smislenu celinu). Okvir PPTEval te tri dimenzije ocenjuje odvojeno, pri čemu se ocena dizajna izvodi iz iscrtane slike slajda, a ne iz njegovog XML zapisa, jer XML zapis, kao što pokazuje odeljak 2.5, ne mora da sadrži sve elemente koji određuju konačan izgled [32].

Sistem opisan u ovom radu ocenjivanje sprovodi kroz nekoliko ugrađenih mehanizama, od rubrike pri izboru šablonskih slajdova do determinističke provere izlaza, dok je zaseban evaluacioni sloj ovog tipa opisan u odeljku 8.1.2.

2.5 Format Office Open XML

Datoteka `.pptx` je ZIP arhiva sa skupom XML dokumenata, standardizovana kao ECMA-376, koji je preuzet i kao standard ISO/IEC 29500 [7]. Za razumevanje rada sistema bitno je nekoliko elemenata te strukture: `ppt/presentation.xml` sadrži dimenzije slajda i uređenu listu slajdova; `ppt/slides/slideN.xml` sadrži oblike, tekst i slike pojedinačnog slajda; pozicije i dimenzije zadaju se u jedinicama EMU (914400 po inču); i, najvažnije, slajd nasleđuje deo oblikovanja od rasporeda (*layout*) i mastera, pa deo vizuelnih elemenata koji se vide na slajdu nije zapisan u njegovom sopstvenom XML-u. Ova poslednja činjenica objašnjava jedno konkretno ograničenje opisano u odeljku 7.6: analiza pojedinačnog slajda ne otkriva uvek sve elemente njegovog vizuelnog identiteta.

Za programski rad sa formatom koristi se biblioteka python-pptx, koja omogućava kreiranje, čitanje i izmenu PPTX datoteka, ali ne prikazuje dokument, pa ni ne obavlja prelom teksta: svojstva samopodešavanja se samo upisuju u datoteku, a stvarni prelom određuje program koji dokument prikazuje. Biblioteka nudi jedino pomoćnu procenu (`fit_text`), koja uz pristup datoteci fonta traži veličinu slova pri kojoj tekst staje u okvir [3]. Ovo ograničenje objašnjava karakterističan obrazac u generisanom kodu, prikazan u odeljku 7.5.3: agent sam konstruiše aproksimativni model preloma teksta kako bi obezbedio vizuelno prihvatljiv raspored elemenata. Za iscrtavanje slajda u sliku koristi se LibreOffice u režimu bez grafičkog okruženja [28], a PDF izlaz se alatom `pdftoppm` iz paketa Poppler pretvara u rastersku sliku [23].

Glava 3

Pregled postojećih pristupa

Priroda predloženih rešenja za automatsko generisanje prezentacija bitno se menjala sa razvojem jezičkih modela. U ovoj glavi dat je pregled najvažnijih pristupa i njihov odnos prema rešenju opisanom u ovom radu.

3.1 Pristupi zasnovani na sažimanju dokumenata

Rani radovi posmatraju generisanje prezentacije kao sažimanje dokumenta. Sistem PPSGen za zadati naučni rad regresionim modelom procenjuje važnost rečenica, a zatim celobrojnim linearnim programiranjem bira i raspoređuje rečenice i ključne fraze u niz slajdova [11]. Sistem DOC2PPT proširuje zadatak na generisanje cele prezentacije iz naučnog dokumenta, uključujući izbor slika, pristupom tipa *sequence-to-sequence* sa modulima za parafraziranje i predviđanje rasporeda [8].

Oba pristupa pretpostavljaju postojanje ulaznog dokumenta iz kog se sadržaj izvodi, a vizuelni rezultat ograničavaju na rasporede koje sistem sam konstruiše; poštovanje postojećeg šablona prezentacije nije bilo predmet tih radova.

3.2 Pristupi zasnovani na velikim jezičkim modelima

Sistem PPTAgent generisanje prezentacije deli u dva koraka: analizu referentne prezentacije (klasifikacija slajdova po funkcionalnom tipu, izvlačenje šeme sadržaja) i zatim iterativno generisanje *akcija izmene* nad izabranim referentnim

slajdovima, umesto građenja slajda od nule [32]. Uz njega je predstavljen i okvir PPTEval za ocenjivanje po dimenzijama sadržaja, dizajna i koherentnosti. Sličnost sa rešenjem u ovom radu je u korišćenju postojećih slajdova kao izvora vizuelnog jezika; razlika je u tome što PPTAgent menja postojeći slajd, dok sistem opisan ovde generiše nov slajd koristeći izabrane slajdove kao uzor.

Sistem AutoPresent, predstavljen zajedno sa skupom SlidesBench, iz uputstva na prirodnom jeziku generiše program na jeziku Python koji se izvršava i proizvodi PPTX slajd [9]. Programski pristup pri tome daje kvalitetnije slajdove koji se mogu dalje menjati, u odnosu na generisanje slike, a razmatrano je i iterativno samopopravljanje izlaza. Rešenje opisano u ovom radu deli tu osnovnu ideju, ali kod ne generiše jednim pozivom modela, već ga proizvodi agentski alat koji radi u izolovanom radnom prostoru, može više puta da pokrene kod i sam ispravlja greške pre nego što vrati rezultat.

3.3 Komercijalni alati

Postoji veći broj komercijalnih alata za generisanje prezentacija iz kratkog opisa teme. Njihov tehnički rad nije javno dokumentovan u recenziranoj literaturi, pa se u ovom radu ne navode pojedinačna poređenja: takva poređenja bi se mogla osloniti samo na marketinške materijale, a ne na proverljive izvore.

3.4 Pozicioniranje rada

Tabela 3.1 sažima odnos opisanih pristupa i rešenja iz ovog rada po dimenzijama koje su za postavljeni problem najvažnije.

Rešenje opisano u ovom radu se od navedenih pristupa razlikuje po tome što: ulaz je kratak opis teme, publike i tona, a ne postojeći dokument; vizuelni jezik se preuzima iz proizvoljnog standardizovanog šablona prezentacije, uz izbor konkretnih slajdova semantičkom pretragom i rerangiranjem koje uzima u obzir izgled slajda; slajd nastaje izvršavanjem koda koji piše agentski alat u izolovanom radnom prostoru, uz automatsku proveru i ponovne pokušaje; za svaki slajd se generiše više varijanti između kojih korisnik bira; i sistem je realizovan kao potpuna aplikacija sa korisničkim interfejsom, a ne kao istraživački prototip.

Poređenje u tabeli 3.1 odnosi se na pristup, a ne na kvalitet izlaza: navedeni sistemi nisu izvršeni nad istim skupom podataka, pa tabela ne daje osnovu za

Tabela 3.1: Poređenje pristupa automatskom generisanju prezentacija po ključnim dimenzijama problema.

	PPSGen [11]	DOC2PPT [8]	PPTAgent [32]	AutoPresent [9]	SlideFlow (ovaj rad)
Ulaz	naučni rad	naučni rad	dokument i referentna prezentacija	uputstvo na prirodnom jeziku	kratak opis teme, publike i tona
Način izrade slajda	izbor rečenica	neuronsko generisanje i raspoređivanje	izmene nad referentnim slajdom	generisanje programskog koda	agentsko generisanje i izvršavanje koda
Vizuelni identitet	nije u fokusu	sopstveni raspored	preuzet iz referentne prezentacije	sopstveni raspored	preuzet iz proizvoljnog šablona, uz semantički izbor slajdova
Evaluacija	metrike sažimanja i korisnička studija	automatske metrike	okvir PPTEval	skup SlidesBench	analiza materijala jednog potpunog prolaska

njihovo rangiranje.

Glava 4

Opis problema i zahtevi

4.1 Postavka problema

Neka je dat standardizovani šablon prezentacije T sa slajdovima $s_1, \ldots, s_m$, gde svaki slajd predstavlja jedan obrazac oblikovanja (naslovni slajd, sadržaj, poređenje, tabela). Neka je dalje dat korisnički zahtev Z: tema prezentacije, opis publike, ton izlaganja i broj slajdova k. Problem je konstrukcija prezentacije $P = (p_1, \ldots, p_k)$ u formatu PPTX takve da:

1. tekstualni sadržaj slajdova p_i odgovara temi i publici iz Z i čini koherentnu celinu;
2. vizuelni jezik slajdova p_i odgovara vizuelnom jeziku šablona T;
3. svaki p_i je tehnički ispravna PPTX datoteka koja se može dalje menjati;
4. raspored elemenata je vizuelno prihvatljiv: tekst ne izlazi izvan okvira, elementi se ne preklapaju.

Prva dva zahteva su suprotstavljena na netrivijalan način: sadržaj generisan bez obzira na šablon se ne uklapa u raspoložive okvire, dok sadržaj strogo prilagođen jednom šablonskom slajdu gubi slobodu u oblikovanju poruke. Rešenje ovu napetost razrešava redosledom operacija: sadržaj se generiše prvi, a zatim se za njega bira šablonski slajd čiji raspored može da ga primi, pri čemu je mogućnost popunjavanja eksplicitan kriterijum izbora (odeljak 6.3.2). Treći zahtev može se proveriti automatski, primenom determinističkog postupka; četvrti nije, jer python-pptx ne prikazuje dokument, pa se stvarne dimenzije ispisanog teksta ne mogu dobiti bez iscrtavanja [3].

4.2 Funkcionalni i nefunkcionalni zahtevi

Funkcionalni i nefunkcionalni zahtevi u ovom radu izvedeni su iz postavljenih ciljeva i planirane strukture sistema, a njihova ostvarivost potvrđena je kroz realizaciju sistema.

FZ1. Administrator učitava PPTX šablon; sistem ga obrađuje u pozadini i izveštava o napretku.

FZ2. Sistem prikazuje listu šablona, omogućava izbor podrazumevanog šablona i njegovo brisanje sa svim pripadajućim datotekama.

FZ3–FZ4. Za svaki slajd šablona sistem generiše opis namene i tehničkih karakteristika, sliku i vektorsku reprezentaciju koja se trajno čuva.

FZ5–FZ6. Na osnovu teme, publike, tona i broja slajdova sistem generiše plan prezentacije po slajdovima (naslov, cilj, ključne tačke, beleške za izlagača, opis vizuelnog rešenja, stilske smernice); plan se može menjati pre generisanja.

FZ7. Za svaki slajd plana sistem bira šablonske slajdove kombinacijom vektorske pretrage i rerangiranja koje uzima u obzir sliku.

FZ8. Za svaki slajd plana sistem generiše više varijanti, svaku kao zasebnu PPTX datoteku sa slikom za pregled.

FZ9–FZ10. Korisnik bira varijantu i redosled slajdova; sistem sklapa i omogućava preuzimanje konačne prezentacije.

NZ1. Dugotrajne operacije ne blokiraju HTTP zahtev; klijent dobija identifikator posla i prati status.

NZ2. Neuspeh jedne varijante ne prekida ceo posao; greške koje ima smisla ponoviti razlikuju se od onih koje nema.

NZ3. Svaka generisana datoteka prolazi automatsku proveru pre nego što se ponudi korisniku.

NZ4. Za svaki pokušaj generisanja čuvaju se upit, komanda i izlaz, radi kasnije analize.

NZ5. Svaki servis se isporučuje kao kontejnerska aplikacija i ima ugrađene provere dostupnosti i spremnosti.

NZ6. Pristupni podaci se ne nalaze u izvornom kodu, već u promenljivama okruženja ili sistemu za upravljanje tajnama.

4.3 Pretpostavke

Sistem pretpostavlja jedan važeći šablon prezentacije: pretraga šablonskih slajdova ograničena je na slajdove šablona označenog kao podrazumevani, a ako je više šablona označeno kao podrazumevano, postupak se prekida uz grešku (odeljak 6.3.2). Cela prezentacija time počiva na jednom vizuelnom jeziku, bez mešanja šablona. Sistem takođe pretpostavlja okruženje sa instaliranim alatima Codex CLI, LibreOffice i `pdftoppm`, i pristup uslugama Azure OpenAI, Azure Blob Storage i bazi sa MongoDB interfejsom; provera spremnosti servisa eksplicitno proverava prisustvo tih zavisnosti.

Glava 5

Arhitektura sistema

5.1 Pregled

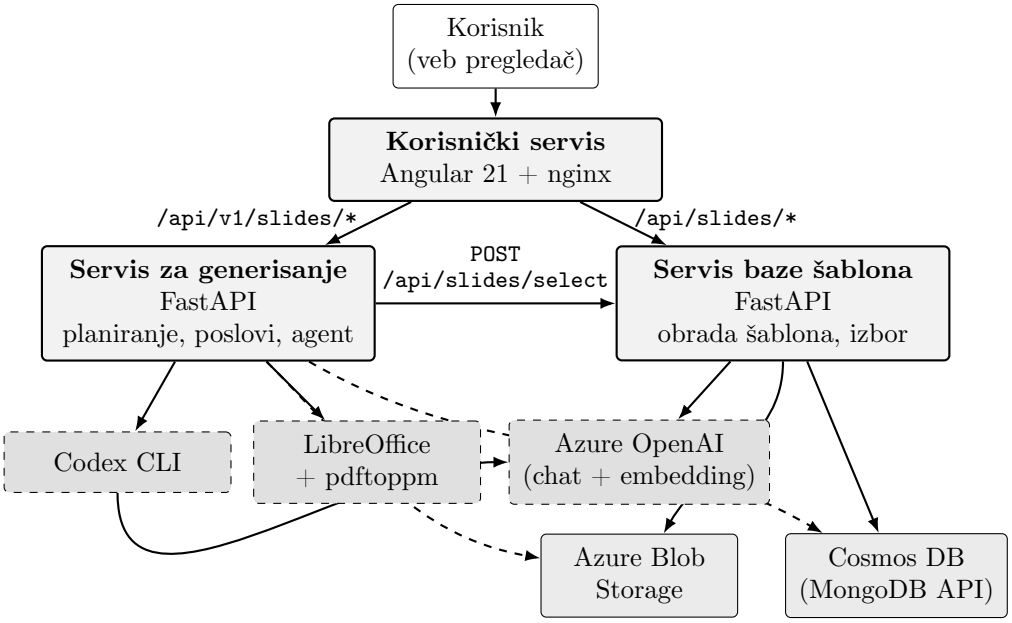
Sistem *SlideFlow* čine tri nezavisno isporučiva servisa, u radu nazvana korisnički servis, servis za generisanje i servis baze šablona (u repozitorijumu `frontend-service`, `backend-service` i `slide-database-service`), koja komuniciraju preko HTTP-a, uz oslanjanje na spoljne usluge za jezičke modele i skladištenje podataka i na lokalne izvršne programe za generisanje i iscrtavanje slajdova (slika 5.1).

Korisnički servis vodi korisnika kroz proces u pet koraka, prikazuje napredak i inicira preuzimanje rezultata; ne obavlja obradu prezentacija. **Servis za generisanje** je sloj orkestracije: planira sadržaj, traži šablonske slajdove od servisa baze šablona, pokreće agentski alat, proverava rezultat i sklapa konačnu prezentaciju. **Servis baze šablona** je jedini servis koji razume unutrašnju strukturu šablona: rastavlja ga na slajdove, opisuje ih, indeksira i po zahtevu bira najpogodnije.

Ova podela odgovara načelu da servis treba da ima jednu jasno određenu odgovornost. Praktičan razlog za nju je i tehnološki: obrada šablona je pretežno ulazno-izlazna, dok je generisanje slajdova procesorski zahtevno i traži prisustvo dodatnih izvršnih programa, pa se dva servisa mogu skalirati nezavisno.

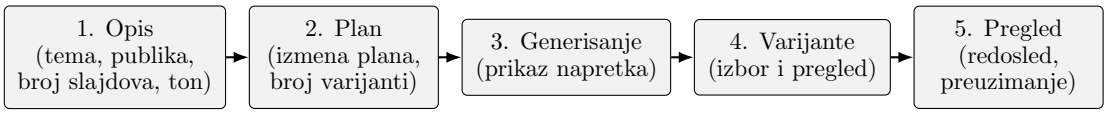
5.2 Korisnički servis

Korisnički deo je realizovan u Angular-u [10] (verzija 21), sa samostalnim komponentama i signalima za upravljanje stanjem, bez klasičnog usmeravanja; servis `NavigationService` razlikuje glavnu stranu sa višekoračnim interfejsom i administrativnu stranu za šablone. Višekoračni interfejs obuhvata pet koraka: opis



Slika 5.1: Arhitektura sistema. Isprekidane linije označavaju direktan pristup skladištu iz servisa za generisanje pri preuzimanju datoteka izabranih šablonskih slajdova (odeljak 5.6).

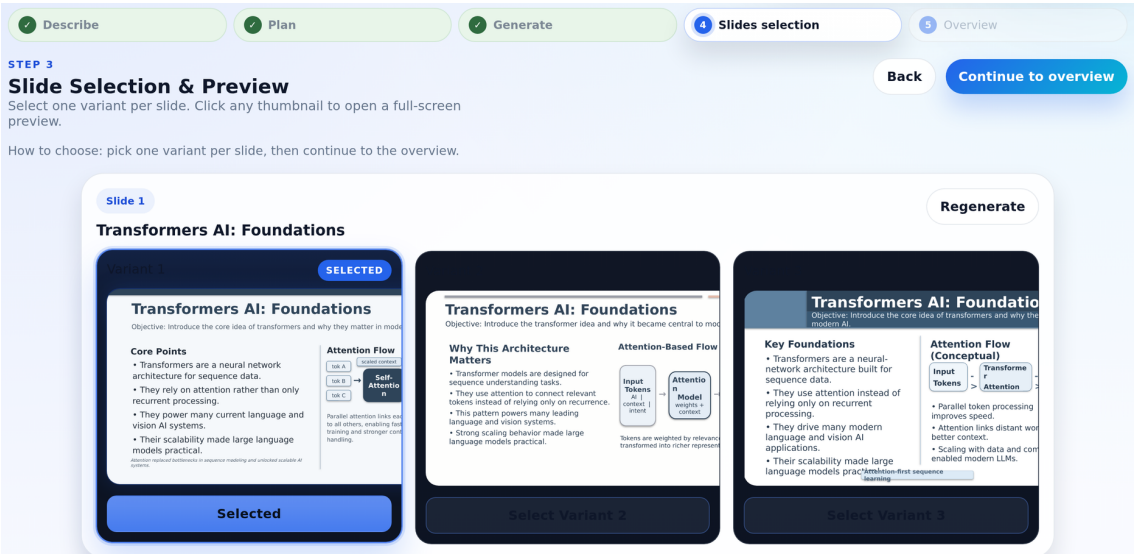
(tema, publika, ton, broj slajdova), plan (izmena plana, broj varijanti), generisanje (prikaz napretka), izbor varijanti i pregled sa preuzimanjem (slika 5.2). Stanje se čuva u `localStorage`, tako da se rad nastavlja i posle osvežavanja stranice.



Slika 5.2: Pet koraka višekoračnog interfejsa.

Slika 5.3 prikazuje četvrti korak u radu, na primeru prezentacije analizirane u glavi 7. Za svaki slajd prikazane su sve generisane varijante sa slikama pregleda, uz oznaku izabrane varijante i dugme za ponovno generisanje; time su ostvareni zahtevi FZ8–FZ10.

Pošto generisanje traje znatno duže od uobičajenog HTTP zahteva, komunikacija sa serverom zasniva se na periodičnom proveravanju statusa posla (*polling*, interval jedna sekunda za generisanje, pet sekundi za obradu šablona). Aplikacija se isporučuje kao statički sadržaj koji poslužuje nginx, koji istovremeno prosleđuje putanje `/api/v1/` servisu za generisanje i putanje `/api/slides/` servisu baze



Slika 5.3: Korak izbora varijanti: za svaki slajd prikazuju se generisane varijante sa slikama pregleda, a korisnik označava jednu od njih.

šablona, pa sav saobraćaj tako ide preko istog porekla.

5.3 Servis za generisanje

Servis je realizovan u FastAPI [25]. Pri pokretanju se uspostavljaju dva upravljača poslovima, jedan za planiranje sadržaja i jedan za generisanje slajdova, koji se uredno zaustavljaju pri gašenju aplikacije. Glavni resursi, sa prefiksom `/api/v1/slides`, obuhvataju kreiranje i praćenje posla planiranja (`POST`/`GET .../content-plan/jobs`), kreiranje i praćenje posla generisanja (`POST`/`GET .../jobs`) i preuzimanje rezultata, odnosno pojedinačne varijante ili sklopljene prezentacije. Drugi prefiks, `/api/slides`, sadrži resurs za generisanje jednog slajda iz slobodnog teksta, korišćen za testiranje, van toka rada korisničkog interfejsa.

Unutrašnja struktura prati tri sloja: resursi (`src/api`), poslovna logika (`src/services`) i klijenti prema spoljnim sistemima (`src/clients`). Najvažniji moduli su `slide_job_manager.py` (red poslova, paralelno izvršavanje varijanti, ponovni pokušaji, sklapanje prezentacije), `codex_cli_service.py` (sastavljanje upita i pokretanje agentskog alata), `slide_template_context_service.py` (strukturni sažetak šablonskog slajda pomoću python-pptx) i `slide_content_planner_service.py` (generisanje plana sadržaja).

5.4 Servis baze šablona

Servis takođe koristi FastAPI. Pri pokretanju uspostavlja vezu sa kolekcijom `slides` u bazi (MongoDB API [19]) i sa dva kontejnera u Azure Blob Storage [18]: jednim za slike slajdova, jednim za PPTX datoteke sa jednim slajdom. Resursi obuhvataju otpremanje i praćenje obrade šablona (`POST /ingest`, `GET /ingest/jobs/{id}`), upravljanje šablonima (`GET /templates`, `POST /templates/default`, `DELETE /templates/{ime}`), izbor šablonskih slajdova za zadati sadržaj (`POST /select`) i preuzimanje slika i PPTX datoteka.

5.5 Model podataka

Centralni zapis je opis šablonskog slajda, koji se čuva u kolekciji `slides` (tabela 5.1). Isti model sadržaja slajda (naslov, cilj, ključne tačke, beleške za izlagača, opis vizuelnog rešenja, stilske smernice), definisan bibliotekom Pydantic [24], koristi se u tri konteksta: kao izlaz planiranja, kao ulaz za izbor šablonskih slajdova i kao ulaz za agentsko generisanje. Stanje poslova čuva se u memoriji procesa (odeljak 8.1.5).

Tabela 5.1: Polja zapisa o šablonskom slajdu.

Polje	Sadržaj
`slide_id`, `template_name`, `slide_index`	identifikator slajda, ime šablona, redni broj u šablonu
`is_default`	da li slajd pripada podrazumevanom šablonu
`business_description`	opis namene: opis, predloženi slučajevi upotrebe, podržane vrste podataka, ograničenja sadržaja
`technical_description`	brojevi izvedeni iz XML-a: naslovi, pasusi, tekstualni okviri, grafikoni, tabele, slike, konektori, dekorativni oblici
`flags`	logičke oznake (da li je slajd uputstvo, da li sadrži vizuelni sadržaj)
`selection_text`, `embedding`	tekst za pretragu i njegova vektorska reprezentacija (1536)
`image_blob_name`, `pptx_blob_name`	putanje datoteka u skladištu objekata

5.6 Komunikacija između servisa

Servis za generisanje pri obradi posla poziva resurs `/select` sa sadržajem slajda i traženim brojem kandidata. Ako odgovor sadrži samo identifikatore slajdova (stariji oblik odgovora), klijent na strani servisa za generisanje sam dopunjuje podatke, pa pronalazi zapise u bazi i preuzima datoteke neposredno iz skladišta objekata (isprekidane linije na slici 5.1). Time se binarni sadržaj ne prenosi kroz posrednički servis, po cenu toga što i servis za generisanje poznaje šemu zapisa i poseduje pristupne podatke za bazu i skladište.

5.7 Podešavanja

Oba serverska servisa učitavaju podešavanja iz promenljivih okruženja, datoteke `.env` i datoteke `config/config.json`, tim redosledom prioriteta. Pristupni podaci za bazu i skladište objekata dolaze iz promenljivih okruženja ili sistema za upravljanje tajnama; predviđeno je i pribavljanje pristupnog žetona preko `DefaultAzureCredential`, čime se izbegava čuvanje ključa u podešavanjima. Podešavanja koja neposredno utiču na ponašanje sistema i na tumačenje rezultata u glavi 7 data su u tabeli 5.2.

Tabela 5.2: Podešavanja koja utiču na ponašanje sistema i njihove vrednosti u isporučenim podešavanjima; u zagradi je vrednost na koju se sistem vraća kada podešavanje nije zadato.

Podešavanje	Vrednost	Značenje
`SLIDE_JOB_WORKERS`	2 (1)	Broj poslova koji se obrađuju uporedo.
`SLIDE_VARIANT_PARALLELISM`	3 (2)	Broj varijanti generisanih uporedo u okviru jednog posla.
`SLIDE_VARIANT_RETRY_LIMIT`	2	Broj dodatnih pokušaja po varijanti (ukupno najviše tri).
`CODEX_TIMEOUT_SECONDS`	600	Vremensko ograničenje jednog poziva agentskog alata.
`SLIDE_PREVIEW_WIDTH`	1280	Širina slike pregleda u tačkama.

5.8 Isporuka

Svaki servis ima svoju datoteku `Dockerfile` [6]. Serverski servisi dodatno instaliraju LibreOffice, alate iz paketa Poppler i podršku za fontove, neophodne za iscrtavanje slajdova, i izvršavaju se pod korisnikom bez administratorskih privilegija. Oba izlažu proveru dostupnosti (`/health/liveness`) i proveru spremnosti (`/health/readiness`); provera spremnosti servisa za generisanje eksplicitno utvrđuje da li su dostupni izvršni programi `codex`, `libreoffice` i `pdftoppm` i da li su skladišta dostupna; odsustvo bilo kog od njih se u suprotnom ispoljava tek kao greška usred generisanja.

5.9 Razmatranje arhitektonskih odluka

Podela na tri servisa izoluje znanje o šablonima u jedan servis, čime obrada šablona i generisanje mogu nezavisno da se razvijaju i skaliraju. Jedinstvena aplikacija bi bila jednostavnija za isporuku, ali bi mešala dve različite vrste opterećenja.

Agent kao zaseban proces, umesto izvršavanja koda unutar serverskog procesa, donosi izolaciju radnog direktorijuma, mogućnost da agent sam više puta pokrene i popravi kod pre nego što vrati rezultat i otpornost servera na otkaz agenta, ali po cenu zavisnosti od spoljnog izvršnog programa i otežane kontrole nad tim šta se u toku rada zaista izvršava.

Generisanje koda umesto direktnog generisanja slike daje PPTX datoteku koja se može dalje menjati, u skladu sa nalazima iz literature [9], i čini proces proverljivim jer generisani kod ostaje sačuvan u radnom prostoru posla.

Čuvanje stanja poslova u memoriji uklanja zavisnost od dodatnog skladišta i dovoljno je za jedan primerak servisa; trajno čuvanje stanja opisano je u odeljku 8.1.5.

Glava 6

Implementacija

Ova glava opisuje kako je arhitektura iz prethodne glave realizovana: najpre organizaciju izvornog koda, zatim, po fazama, pripremu baze šablonskih slajdova i generisanje prezentacije na zahtev, i najzad sklapanje konačne prezentacije, rukovanje greškama i automatske testove.

6.1 Organizacija koda i obim

Izvorni kod sistema organizovan je u jedinstvenom repozitorijumu sa tri direktorijuma, po jednim za svaki servis: `backend-service`, `slide-database-service` i `frontend-service`. Serverski servisi (Python 3.12) dele istu strukturu direktorijuma i slične klijente za jezički model, bazu i skladište objekata, čime svaki servis ostaje nezavisno isporučiv, bez zajedničke biblioteke koju bi oba morala da prate. Tabela 6.1 daje obim izvornog koda.

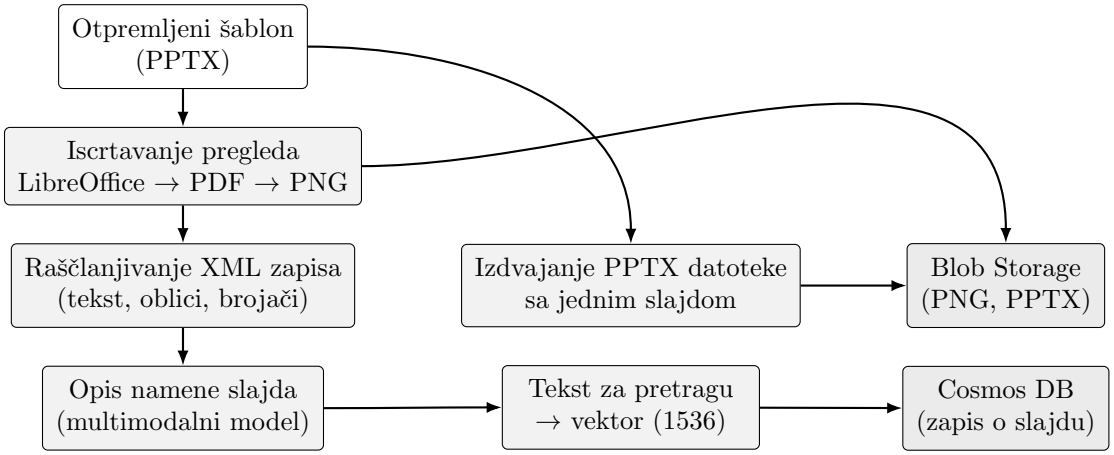
U korenu repozitorijuma nalazi se zbirni `README.md`, usklađen sa realizovanim stanjem celog sistema.

Tabela 6.1: Obim izvornog koda po servisima, u linijama.

Servis	Jezik	Broj linija
Servis za generisanje	Python (sa testovima)	3473
Servis baze šablona	Python (sa skriptama)	3020
Korisnički servis	TypeScript / HTML / CSS	2910 / 811 / 3072

6.2 Priprema baze šablonskih slajdova

Priprema baze šablonskih slajdova (*offline* faza) izvršava se jednom po šablonu i pretvara neuređenu PPTX datoteku u skup zapisa: svaki slajd je opisan tekstualno, predstavljen slikom, izdvojen kao samostalna PPTX datoteka i indeksiran vektorskom reprezentacijom (slika 6.1).



Slika 6.1: Postupak pripreme baze šablonskih slajdova, izvršava se za svaki slajd otpremljenog šablona.

Obrada se pokreće učitavanjem datoteke na `/api/slides/ingest`, koja kreira posao sa jedinstvenim identifikatorom i pokreće obradu u pozadini (kod 202). Napredak se prati na `/api/slides/ingest/jobs/{id}`; pripremne faze (čuvanje, raščlanjivanje) mapirane su na fiksne procente, a upis slajdova linearno pokriva opseg 35–95%, srazmerno broju obrađenih slajdova.

6.2.1 Iscrtavanje i raščlanjivanje

Za svaki slajd šablona proizvodi se slika: ceo šablon se konvertuje u PDF programom LibreOffice u režimu bez grafičkog okruženja, a stranice PDF-a rasterizuju alatom `pdftoppm` iz paketa Poppler [28, 23]. Pre pokretanja LibreOffice-a postavlja se izolovan profil (privremeni `HOME` i `XDG_*` promenljive), što sprečava sudare pri istovremenom pokretanju više instanci u okruženju bez grafičke sesije, kakvo je kontejner. Ako LibreOffice ili `pdftoppm` nisu dostupni, iscrtavanje se preskače i obrada nastavlja bez slike tog slajda, što onemogućava kasnije multimodalno opisivanje i rerangiranje po vizuelnim kriterijumima za taj slajd.

Umesto biblioteke python-pptx, struktura slajda se čita direktno iz XML zapisa: dimenzije platna (`p:sldSz`), tekstualni oblici sa pozicijom, dimenzijama i veličinom fonta, brojevi grafikona/tabela/slika/konektora, i odsečak izvornog XML-a (do 4000 znakova) koji se kasnije prosleđuje jezičkom modelu kao kontekst.

Naslov se prepoznaje po rezervisanom mestu (*placeholder*) tipa `title` ili `ctrTitle`; ako takav ne postoji, primenjuje se heuristika koja ocenjuje kratke tekstualne oblike smeštene u gornjih 45% visine slajda. Ocena se sastavlja od tri činioca, navedena po opadajućem uticaju: prosečne veličine fonta oblika u odnosu na najveći font na slajdu, vertikalnog položaja oblika, pri čemu su oblici bliži vrhu slajda povoljniji, i dužine teksta, pri čemu je kraći tekst povoljniji. Kao naslovi zadržavaju se najviše dva oblika čija ocena pređe zadati prag. Težine ova tri činioca i vrednost praga zadate su kao konstante u kodu, pa se ponašanje heuristike može podesiti bez izmena u ostatku postupka.

6.2.2 Opisivanje slajda pomoću jezičkog modela

Strukturni podaci ne govore čemu slajd služi, pa se za svaki slajd poziva multimodalni model kome se prosleđuju naslovi, tekst tela, tehničke činjenice, logičke oznake i XML odsečak, uz sliku slajda kao ugrađeni URL sa podacima (engl. data URL). Traži se JSON sa opisom namene, predloženim slučajevima upotrebe, podržanim vrstama podataka i ograničenjima sadržaja. Upit izričito nalaže da su XML podaci primarni izvor istine, da slika služi kao dopunski kontekst za raspored i da model ne sme da izmišlja sadržaj koji nije potvrđen u izvorima. To je mera protiv halucinacija koje bi direktno uticale na kasniji izbor šablona (odeljak 2.1).

Poziv se ponavlja najviše tri puta; ako svi pokušaji ne uspeju, umesto opisa se upisuje prazna struktura sa porukom o grešci u polju ograničenja, a obrada nastavlja; takav slajd i dalje ima sliku i tehnički opis, pa može biti izabran, ali je manje verovatno da će ga pronaći semantička pretraga.

6.2.3 Izdvajanje datoteka sa jednim slajdom

Da bi izabrani šablonski slajd mogao da se prosledi agentu kao referenca, izdvaja se u samostalnu PPTX datoteku direktnom izmenom ZIP arhive: iz `ppt/presentation.xml` i pripadajućih relacija uklanjaju se svi slajdovi osim traženog, a ostale komponente paketa, kao što su rasporedi, master slajdovi, tema i fontovi, ostaju netaknute. Time nova datoteka verno nosi vizuelni identitet šablona, što

je i cilj: alternativa (kreiranje nove prezentacije i kopiranje oblika) bi zahtevala posebno prenošenje nasleđenog oblikovanja.

6.2.4 Vektorizacija

Tekst za pretragu sastavlja se spajanjem opisa namene, predloženih slučajeva upotrebe, podržanih vrsta podataka, ograničenja sadržaja i teksta koji se zaista nalazi na slajdu, i pretvara u vektor dimenzije 1536. Ograničenja sadržaja ulaze u tekst za pretragu zato što nose informaciju o tipu slajda, odnosno o vrsti sadržaja za koju je slajd predviđen; ista informacija koristi se i pri rerangiranju, kroz kriterijum usklađenosti sa ograničenjima sadržaja (odeljak 6.3.2).

6.2.5 Trajno čuvanje

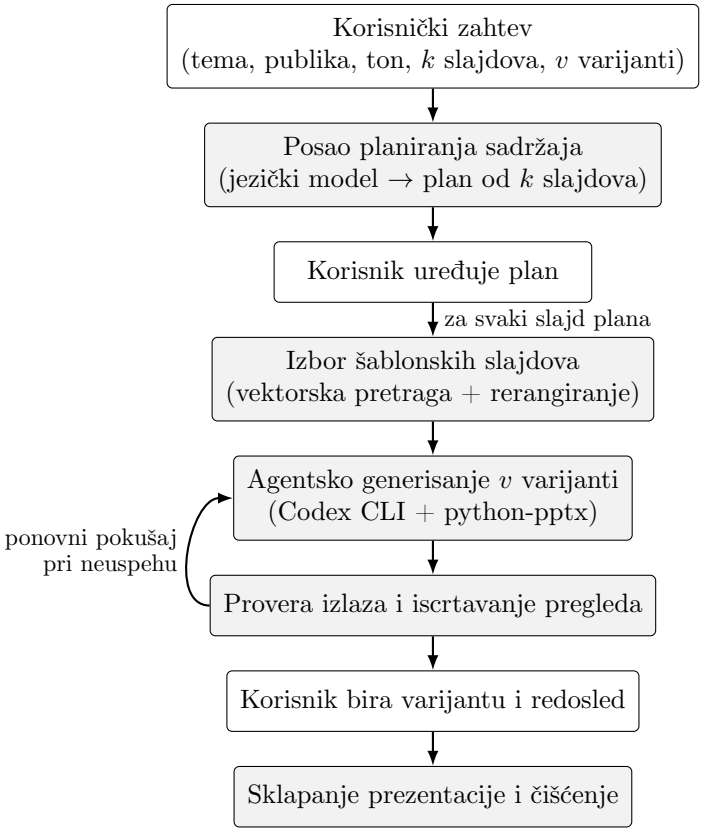
Za svaki slajd redom: izračunavanje vektora, izdvajanje datoteke sa jednim slajdom, otpremanje slike i PPTX-a u skladište, upis identifikatora slajda u metapodatke objekata i upis zapisa u bazu (obrazac imena: `{šablon}/slide_{indeks}.png/.pptx`). Ponovno otpremanje istog šablona briše sve postojeće zapise i objekte tog šablona pre upisa novih, čime je obrada idempotentna. Ako obrada pojedinačnog slajda ne uspe, uklanjaju se njegovi već otpremljeni objekti i poništava cela obrada šablona: brišu se zapisi svih slajdova unetih u toj obradi i svi objekti sa prefiksom šablona, da bi se izbeglo stanje u kome je šablon u bazi prisutan samo delimično.

6.3 Generisanje prezentacije

Generisanje prezentacije (*online* faza) pokreće se korisničkim zahtevom i odvija u dva odvojena posla: plan sadržaja, i za svaki slajd plana poseban posao generisanja (slika 6.2).

6.3.1 Planiranje sadržaja

Iz teme, publike, tona i broja slajdova (najviše 20) jezički model generiše plan: za svaki slajd naslov, cilj, ključne tačke, beleške za izlagača, opis vizuelnog rešenja i stilske smernice. Poslednja dva polja pored sadržaja slajda nose i informacije koje kasnije koristi izbor šablona (odeljak 6.3.2).



Slika 6.2: Tok generisanja prezentacije. Sivi koraci se izvršavaju na serveru, beli su interakcija sa korisnikom.

Odgovor se obrađuje u tri koraka: izdvajanje JSON-a iz teksta, provera prema Pydantic modelu i, ako bilo koji korak ne uspe, ponovni poziv sa opisom greške u upitu, najviše tri pokušaja. Ako model vrati manje slajdova nego što je traženo, plan se dopunjava slajdovima sa neutralnim sadržajem koji korisnik treba da izmeni; u interfejsu se takvi slajdovi ne razlikuju vizuelno od ostalih. Planiranje se izvršava kao posao u pozadini (identifikator, status, rezultat dostupan tek po završetku; zahtev za nezavršenim rezultatom vraća kod 409).

6.3.2 Izbor šablonskih slajdova

Pre nego što posao generisanja uđe u red, servis za generisanje od servisa baze šablona traži izbor šablonskih slajdova za sadržaj tog slajda. Broj traženih kandidata jednak je broju varijanti; početni skup za pretragu je četvorostruko veći.

Tekst upita spaja naslov, cilj, ključne tačke, beleške za izlagača, opis vizuelnog

rešenja i stilske smernice, i pretvara se u vektor istim modelom i dimenzijom kao pri obradi šablona. Iz baze se učitavaju svi slajdovi podrazumevanog šablona (ako takvih nema, ili ih ima iz više šablona, postupak se prekida uz grešku); sličnost se računa izrazom (2.4) u aplikativnom sloju. Kada vektori nisu uporedivi, jer im se dužine razlikuju ili je norma nula, funkcija sličnosti vraća -1; odbacivanjem vrednosti ispod $-0{,}5$ iz skupa kandidata uklanjaju se upravo takvi zapisi. Prag time nije semantički kriterijum, budući da se sličnost dveju upotrebljivih reprezentacija u praksi nalazi u pozitivnom delu opsega.

Broj kandidata koji ulazi u drugi stepen zavisi od veličine šablona: polazi se od polovine slajdova podrazumevanog šablona, a taj broj se zatim ograničava sa tri strane, tako da nije manji od broja kandidata koji se na kraju vraća, niti veći od traženog broja kandidata, niti veći od broja slajdova koje šablon ima. Kod malih šablona zato odlučuje donja granica, kod velikih traženi broj kandidata, a između njih polovina šablona.

Drugi stepen prosleđuje jezičkom modelu, za svakog kandidata, njegov opis namene, tehnički opis, logičke oznake i sliku (preko dva megabajta se preskaču). Zadatak je izbor kandidata, najviše do traženog broja, poređanih od najboljeg, uz obrazloženje, prema rubrici sa šest kriterijuma: kvalitet dizajna i čitljivost sa slike (*design_fit*), mogućnost popunjavanja sadržaja bez prelivanja (*fillability_fit*), usklađenost tehničkog opisa sa očekivanom gustinom sadržaja (*structure_fit*), semantička usklađenost sa ciljem i ključnim tačkama (*content_fit*), usklađenost tona (*tone_fit*) i kazna za neusklađenost sa ograničenjima sadržaja (*risk_penalty*). Prva dva kriterijuma su eksplicitno postavljena kao primarna, ispred opšte semantičke sličnosti, uz zahtev za raznovrsnošću u konačnom skupu.

Ovo odgovara dvostepenoj pretrazi opisanoj u odeljku 2.2.2: prvi stepen je jeftin, drugi uzima u obzir kriterijume koje vektor ne izražava, pre svega vizuelne. Ako poziv modela ne uspe, sistem se vraća na redosled po vektorskoj sličnosti; ako model vrati manje kandidata nego traženo, lista se dopunjava po istom redosledu.

6.3.3 Agentsko generisanje varijanti

Za svakog izabranog kandidata preuzimaju se PPTX i slika, a python-pptx izračunava strukturni sažetak (dimenzije, ime rasporeda, rezervisana mesta i oblici sa pozicijom, veličinom, uzorkom teksta i bojom popune), sažet na najviše dva slajda, osam rezervisanih mesta i četrnaest oblika po kandidatu. Za svaku va-

rijantu datoteke kandidata se kopiraju u zaseban radni direktorijum, da izmene jedne varijante ne utiču na drugu.

Upit agentu sadrži: (1) zahtev za PPTX datoteku sa tačno jednim slajdom, na tačno određenoj putanji; (2) ograničenja postupka, uključujući jezik Python i python-pptx, zabranu traženja odobrenja i menjanja datoteka izvan radnog direktorijuma; (3) zahteve za oblikovanje i samoproveru, koji obuhvataju mešanje vizuelnog jezika svih kandidata bez doslovnog kopiranja, proveru da tekst ne izlazi izvan okvira, da se elementi ne preklapaju i da se veličina fonta po potrebi programski smanji; i (4) sadržaj slajda u formatu JSON i sažete strukturne opise kandidata sa putanjama do njihovih datoteka u radnom direktorijumu. Reprezentativni, skraćeni isečak upita dat je u prilogu A.

Agent nije ograničen na podatke iz upita; pošto su PPTX datoteke i slike kandidata u njegovom radnom direktorijumu, može ih sam otvoriti i analizirati, što je bitno kada je strukturni sažetak prazan (odeljak 7.6). Ako prethodni pokušaj nije uspeo, u upit se dodaje poslednjih 1600 znakova poruke o grešci sa zahtevom da se problem otkloni.

Agent se pokreće kao poseban proces u neinteraktivnom režimu, sa upitom na standardnom ulazu:

```
codex exec --skip-git-repo-check \
     --sandbox danger-full-access \
     --dangerously-bypass-approvals-and-sandbox \
     -C <radni_direktorijum> \
     [--model <model>] -
```

Kod 6.1: Poziv agentskog alata (pojednostavljeno).

Poziv se izvršava u zasebnoj niti (vremensko ograničenje 600 sekundi), da ne blokira petlju događaja servera. Opcije koje isključuju traženje odobrenja i ograničenja izvršavanja su neophodne da agent samostalno pokreće kod bez interakcije, ali znače da se u serverskom okruženju izvršava kod koji je generisao jezički model; izolacija agenta u proizvodnoj postavci opisana je u odeljku 8.1.6.

6.3.4 Provera izlaza i oporavak od grešaka

Po završetku procesa proverava se izlazni kod, postojanje datoteke na očekivanoj putanji (u suprotnom se pretražuje radni direktorijum za datoteku istog imena) i, otvaranjem bibliotekom python-pptx, da datoteka sadrži tačno jedan slajd. Ove

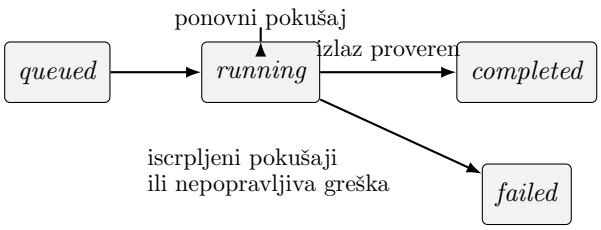
provere su deterministički deo kontrole kvaliteta: garantuju da korisniku neće biti ponuđena datoteka koja se ne može otvoriti, dok se izgled slajda ocenjuje u koraku u kome korisnik bira varijantu na osnovu slike pregleda (slika 5.3).

Svaka varijanta se generiše u najviše tri pokušaja. Sistem razlikuje greške koje ima smisla ponoviti od onih koje nema: ako poruka ukazuje da agentski alat nije pronađen, pristupni podaci nisu podešeni ili traženi šablon ne postoji, ponavljanje se prekida odmah, jer bi svaki naredni pokušaj naišao na isti problem; podela se implementira poređenjem poruke o grešci sa listom poznatih obrazaca. Neuspeh jedne varijante ne prekida generisanje ostalih varijanti istog posla i one ostaju dostupne za pregled, ali se posao označava kao neuspeo čim je bar jedna varijanta neuspela. Ta odluka ima posledicu na nivou cele prezentacije, opisanu u odeljku 6.4.

Za svaku uspešno generisanu varijantu proizvodi se slika istim postupkom kao u *offline* fazi (LibreOffice → PDF → `pdftoppm`, širina 1280 tačaka), kodirana po standardu Base64 i upisana direktno u stanje posla; korisnički deo je dobija u istom odgovoru u kome dobija i status, po cenu većeg odgovora.

6.3.5 Upravljanje poslovima

Poslovi se obrađuju preko reda zadataka: radni zadaci uzimaju poslove iz reda, a unutar posla varijante se generišu uporedo, ograničeno semaforom. Oba broja su podesiva (tabela 5.2); u isporučenim podešavanjima, na koja se odnose i merenja u glavi 7, uporedo se obrađuju dva posla i po tri varijante u okviru posla, dok se sistem bez zadatog podešavanja vraća na jedan posao i dve varijante. Slika 6.3 prikazuje stanja posla i varijante.



Slika 6.3: Stanja posla i pojedinačne varijante. Posao prelazi u *failed* ako je bar jedna varijanta neuspela.

Stanje posla obuhvata vreme kreiranja, početka i završetka, broj uspešnih i neuspešnih varijanti, listu izabranih šablonskih slajdova i, za svaku varijantu,

njeno stanje, broj pokušaja, putanju do izlaza, sliku pregleda i eventualnu grešku.

6.4 Sklapanje prezentacije

Poslovi iste prezentacije povezani su zajedničkim identifikatorom, koji korisnički deo formira od naziva teme i vremenske oznake. Pre preuzimanja korisnički deo šalje serveru svoj izbor: za svaki slajd redni broj izabrane varijante, u redosledu u kome slajdovi treba da se pojave. Kada svi poslovi uspešno dođu u završno stanje, slajdovi se ređaju po tom izboru, a za svaki se uzima izabrana varijanta; ako izbor nije poslat ili izabrana varijanta nije uspešno generisana, uzima se prva uspešna varijanta po rednom broju, čime se sklapanje uvek završava potpunom prezentacijom.

Rezervni izbor deluje unutar posla, dok se na nivou prezentacije primenjuje stroži uslov: sklapanje se pokreće tek kada svi poslovi te prezentacije uspešno dođu u završno stanje, a u suprotnom zahtev za preuzimanjem vraća grešku. Zahtev NZ2 je time ispunjen na nivou varijante, gde neuspeh jedne varijante ne zaustavlja ostale. Slajdovi se dodaju u zajedničku prezentaciju bibliotekom Aspose.Slides, jer python-pptx ne podržava kopiranje slajda između prezentacija uz očuvanje rasporeda, mastera i teme, dok Aspose.Slides ima ugrađenu operaciju kloniranja koja to omogućava.

U evaluacionom režimu Aspose.Slides u dokument dodaje vodeni žig, tj. tekstualnu oznaku o probnoj verziji [1]. Pri preuzimanju prezentacije sistem zato pokreće čišćenje: prolazi kroz sve slajdove, rasporede i master slajdove i uklanja oblike čiji tekst sadrži tu oznaku, a očišćenu datoteku pamti pored originala. Postupak se odnosi na izlazni dokument, dok se izvan probnog režima koristi licencirana verzija biblioteke ili njena zamena.

Sklapanje je mesto na kome se sastaju dva zahteva koja se lako sukobe: konačna prezentacija mora da poštuje izbor varijante i redosled koje je korisnik zadao u koracima 4 i 5 (zahtevi FZ9–FZ10), ali istovremeno mora da se sklopi i onda kada taj izbor nije potpun ili je u međuvremenu zastareo. Rešenje je prikazano u kodu 6.2.

```
1 for job, requested_variant_index in ordered_jobs:
2     variants = sorted(
3         job.get("variants") or [],
4         key=lambda variant: int(variant.get("variant_index") or 10**9),
```

```
    )
    selected_path = None

    # 1. korak: varijanta koju je korisnik izabrao
    if requested_variant_index is not None:
        for variant in variants:
            if int(variant.get("variant_index") or -1) !=
                int(requested_variant_index):
                continue
            if str(variant.get("status") or "") == "completed":
                candidate_path = Path(str(variant.get("output_pptx_path") or
                    ""))
                if candidate_path.is_file():
                    selected_path = candidate_path
            break

    # 2. korak: rezervni izbor, prva uspesna varijanta
    if selected_path is None:
        for variant in variants:
            if str(variant.get("status") or "") != "completed":
                continue
            candidate_path = Path(str(variant.get("output_pptx_path") or ""))
            if candidate_path.is_file():
                selected_path = candidate_path
                break

    if selected_path is None:
        raise RuntimeError(f"No completed slide file found for job
            {job.get('job_id')}")
    slide_paths.append(selected_path)
```

Kod 6.2: Izbor datoteke slajda pri sklapanju prezentacije, iz `slide_job_manager.py`.

Postupak je namerno dvostepen. Prvi korak poštuje korisnikov izbor, ali ga ne uzima zdravo za gotovo: proverava i da je ta varijanta zaista uspešno generisana i da datoteka postoji na disku, jer se izbor formira u pregledaču i stiže odvojenim zahtevom, pa može da opisuje stanje starije od trenutnog. Drugi korak je rezervni: ako izabrana varijanta iz bilo kog razloga nije upotrebljiva, ili ako izbor uopšte nije poslat, uzima se prva uspešna varijanta. Time sklapanje nikada ne prekida posao zbog nesaglasnosti između korisničkog izbora i stvarnog stanja, što je u skladu sa

zahtevom NZ2.

Redosled slajdova rešen je istom logikom na nivou iznad: lista `ordered_jobs` formira se prema redosledu koji je poslao korisnički deo, a poslovi koji se u tom spisku ne pojavljuju dodaju se na kraj, po rednom broju slajda iz plana. Slajd koji je nastao posle poslednje korisnikove potvrde tako ne može da ispadne iz prezentacije.

6.5 Rukovanje greškama

Sistem razlikuje tri nivoa. Na nivou zahteva, resursi vraćaju standardizovane HTTP kodove: 400 za neispravan ulaz, 404 za nepostojeći resurs, 409 kada rezultat još nije spreman ili je posao neuspeo, 502 kada servis za izbor šablona ne vrati kandidate, 503 kada zavisnosti nisu spremne. Na nivou posla, neuspeh varijante beleži se lokalno, a neuspeh koji sprečava obradu celog posla prevodi sve nezavršene varijante u neuspelo stanje. Na nivou spoljnih poziva, pozivi jezičkim modelima obavijeni su ponovnim pokušavanjem sa fiksnim čekanjem, a pristup bazi posebnim postupkom koji prepoznaje ograničenje propusnosti i čeka vreme koje baza sama predloži.

Za svaki pokušaj generisanja u direktorijum `logs/` upisuje se trajanje, identifikatori korišćenih šablonskih slajdova, putanja do upita, cela komanda i izlaz procesa; ovi zapisi su osnovni izvor podataka za analizu u glavi 7.

6.6 Testovi

Servis za generisanje ima dva automatska testa, oba usmerena na životni ciklus posla planiranja sadržaja: prvi proverava ceo tok (kreiranje, prelaz u završno stanje, sadržaj odgovora, poziv jezičkom modelu zamenjen lažnom implementacijom), drugi proverava da zahtev za nezavršenim rezultatom vraća kod 409. Za testove koji zahtevaju stvarno izvršavanje agentskog alata u podešavanjima projekta pripremljena je oznaka `integration`. Ispravnost ostatka sistema proveravana je izvršavanjem celog toka, od unosa šablona do preuzimanja gotove prezentacije; materijal nastao u tim prolascima osnova je analize u glavi 7.

Glava 7

Eksperimentalna analiza

7.1 Cilj i metodologija

Analiza je zasnovana na materijalu nastalom tokom rada sistema: dnevnicima pokušaja sa izmerenim trajanjima, upitima, generisanim kodom, izlaznim PPTX datotekama, slikama varijanti i izveštajima o obradi šablona. Predmet analize su veličine koje se mogu neposredno izmeriti iz tog materijala i zapažanja o rasporedu, u skladu sa inženjerskim karakterom rada, čiji je predmet ponašanje izgrađenog sistema.

Izlaganje ide u dva koraka. Odeljak 7.2 daje pregled izvedenih prolazaka i zapažanja koja se iz njih izvode, a ostatak glave detaljno prati jedan potpun prolazak kroz sistem, na kome se mere pouzdanost tehnički ispravnog izlaza, vremenska cena pojedinačnih koraka i svojstva generisanog rasporeda.

7.2 Pregled izvedenih prolazaka

Izlaz sistema je vizuelni dokument, pa se njegov kvalitet ne meri brojevima: da li je slajd čitljiv, da li raspored deluje uravnoteženo i da li se prezentacija uklapa u zatečeni šablon utvrđuje se gledanjem. Provera je zato tokom razvoja vođena kvalitativno. Za svaki prolazak pokretan je ceo tok, od unosa šablona do sklapanja prezentacije, a zatim su pregledane slike svih generisanih varijanti i upoređene sa zadatim sadržajem i sa izgledom šablona iz kojeg su nastale. Takav pregled je moguć i naknadno, jer svaki pokušaj generisanja ostavlja upit, program koji je agent napisao, izlaznu datoteku, sliku pregleda i zapis o trajanju (odeljak 6.5). Pregled nije sproveden po unapred zadatoj ocenskoj skali niti sa nezavisnim ocenjivačima,

pa se iz njega izvode zapažanja koja se ponavljaju kroz prolaske, a ne brojčana ocena kvaliteta; mogućnost uvođenja zasebnog ocenjivačkog sloja razmatra se u odeljku 8.1.2.

Opseg testiranih ulaza dat je u tabeli 7.1. Obuhvaćena su četiri šablona različitog vizuelnog jezika i sedam tema. Prezentacije su tražene u dužinama od dva do pet slajdova, sa dve odnosno tri varijante po slajdu.

Kolone *poslova* i *varijanti* prate podelu koju sistem pravi pri generisanju. *Posao* je jedinica obrade koja odgovara jednom slajdu tražene prezentacije, pa prezentacija od pet slajdova pokreće pet poslova, koji se obrađuju preko reda zadataka (odeljak 6.3.5). *Varijanta* je jedan predlog izgleda tog istog slajda: unutar posla agentski alat pokreće se više puta nezavisno, svaki put u zasebnom radnom direktorijumu, sa istim sadržajem slajda i istim skupom šablonskih kandidata (odeljak 6.3.3). Varijante istog posla zato nose isti sadržaj u različitim rasporedima, a korisnik za svaki slajd bira jednu od njih. Posao sa tri varijante daje tri izlazne PPTX datoteke sa po jednim slajdom, od kojih u sklopljenu prezentaciju ulazi jedna. Ukupno je izvedeno 19 poslova generisanja i 55 traženih varijanti.

Tabela 7.1: Pregled izvedenih prolazaka. Jedan posao odgovara jednom slajdu prezentacije; kolona *ispravnih* broji varijante za koje je proizvedena PPTX datoteka sa tačno jednim slajdom.

Tema	Šablon	Poslova	Varijanti	Ispravnih
Kubernetes	`test_template4`	2	6	6
GraphQL	`test_template4`	2	6	6
Kafka	`test_template2`	2	6	6
CI/CD	`test_template2`	2	6	6
WebAssembly	`test_template3`	4	10	10
Observability	`test_template3`	2	6	6
Transformeri	`test_template`	5	15	15
Ukupno		**19**	**55**	**55**

Opšta zapažanja iz pregleda su sledeća. Svih 55 proizvedenih datoteka otvara se bibliotekom python-pptx i sadrži tačno jedan slajd, sa medijanom od 15 oblika po slajdu. Nad 55 varijanti zabeleženo je tačno 55 pokušaja, dakle nijedna varijanta nije zahtevala ponovni pokušaj, a trajanje jednog poziva agentskog alata kretalo se od 56 do 499 sekundi, sa medijanom od 113 sekundi; duži pozivi su po pravilu oni u kojima je agent više puta pokretao i dorađivao sopstveni program. Nezavisno od teme i šablona, agent redovno gradi slajd od osnovnih oblika, umesto

da popunjava zatečena rezervisana mesta, a kada plan traži vizuelni element, konstruiše šematski crtež iz pravougaonika, elipsi i linija. Boje i tipografija dosledno se izvode iz šablona, pa generisani slajdovi ostaju prepoznatljivo bliski predlogu.

Nedostaci uočeni pregledom takođe se ponavljaju kroz teme i šablone i gotovo uvek su iste prirode: tekst se ne uklapa u element zadate veličine. Najčešće se javljaju preklapanje naslova sa podnaslovom kada naslov pređe u drugi red i prelamanje reči unutar uskih oblika. Uzrok je isti u svim slučajevima i opisan je u odeljku 2.5: prelom teksta određuje program koji dokument prikazuje, a ne python-pptx, pa agent ne raspolaže gotovom merom zauzetog prostora.

Upravo zato upit od agenta izričito traži samoproveru rasporeda (odeljak 6.3.3): da tekst ne izlazi izvan okvira, da se elementi ne preklapaju i da se veličina fonta po potrebi programski smanji. Agent taj zahtev ispunjava tako što u sopstveni program ugrađuje približan model preloma — procenu broja redova iz prosečne širine znaka i smanjivanje fonta dok tekst ne stane — a onda program pokreće, pregleda dobijeni slajd i dorađuje ga pre nego što zadatak proglasi završenim (odeljak 7.5.3). Ta petlja samoprovere i ispravke odvija se u pozadini, bez učešća korisnika, i uklanja veći deo ovakvih problema još pre nego što varijanta bude ponuđena; na kraju sistem deterministički potvrđuje da je datoteka ispravna i da sadrži tačno jedan slajd (odeljak 6.3.4). Odstupanja koja i posle toga ostaju posledica su toga što je model preloma procena, a ne merenje; merenje geometrije rasporeda kao dodatna provera opisano je u odeljku 8.1.1.

Iz opisanog skupa izdvojen je jedan prolazak, koji se u nastavku glave prati detaljno, kao primer na kome se vidi kako sistem radi. Na njemu se ceo tok može ispratiti u celini, od unosa šablona do preuzimanja gotove prezentacije.

7.3 Postavka

Detaljno analizirani prolazak vodi od unosa šablona do preuzimanja gotove prezentacije. Kao ulaz je poslužio šablon `test_template.pptx` sa pet slajdova, koji pokriva tipične uloge u poslovnoj prezentaciji: naslovnu stranu, dva slajda sa naslovom i nabrajanjem, slajd za poređenje dve stavke i pretežno vizuelni slajd sa fotografijom. Šablon sadrži dvanaest rasporeda, a njegov vizuelni identitet čine tamnoplava paleta, traka sa fotografijom u zaglavlju i beli tekst na tamnoj podlozi. Vektorske reprezentacije su dimenzije 1536.

Zadata tema bila je „Transformers AI”, sa planom od pet slajdova. Za svaki

slajd pokrenut je po jedan posao generisanja; svakom su prosleđena tri šablonska kandidata i tražene su tri varijante, što daje 15 pokušaja generisanja varijanti sa 15 izlaznih PPTX datoteka i slika pregleda, 12 sačuvanih programa koje je agent napisao i jednu sklopljenu prezentaciju od pet slajdova. Svi poslovi izvršeni su sa istim podešavanjima, pa se izmerena trajanja odnose na jedinstvenu postavku.

7.4 Rezultati pripreme baze šablona

Svih pet poziva jezičkog modela za opisivanje slajdova šablona završeno je uspešno, bez ponovnih pokušaja, za ukupno 19,6 sekundi, odnosno prosečno oko 3,9 sekundi po slajdu. Ova vremena obuhvataju samo opisivanje jezičkim modelom, ne i iscrtavanje, vektorizaciju i upis, koji nisu pojedinačno zabeleženi.

Uspešnost od 100% odnosi se na formalnu ispravnost odgovora: model je u svakom pozivu vratio JSON tražene strukture. Sami opisi pokazuju i da je model prepoznao namenu pojedinačnih slajdova: slajd za poređenje opisan je kao pogodan za predstavljanje dve ponude sa kratkim spiskom svojstava, a pretežno vizuelni slajd kao slajd sa velikom slobodnom površinom predviđenom za naknadni sadržaj. Primer opisa dat je u prilogu B.

7.5 Rezultati generisanja slajdova

7.5.1 Pouzdanost

Svih 15 posmatranih pokušaja generisanja varijante zabeleženo je kao prvi pokušaj; nijedan dnevnik ne odgovara drugom ili trećem pokušaju niti sadrži zapis o grešci. Svaka varijanta prošla je deterministička proveru izlaza (postojanje datoteke i tačno jedan slajd), pa je iz svih pet poslova sklopljena potpuna prezentacija od pet slajdova. Mehanizam ponovnih pokušaja u ovom prolasku nije bio potreban, pa se izmerene vrednosti odnose na osnovni tok generisanja, bez ponavljanja; isto važi i za ostale zabeležene prolaske (odeljak 7.2), u kojima nijedna varijanta nije zahtevala ponovni pokušaj.

7.5.2 Trajanje generisanja

Tabela 7.2 sažima izmerena trajanja poziva agentskog alata.

Tabela 7.2: Trajanje generisanja jedne varijante slajda, na osnovu 15 zabeleženih pokušaja.

Veličina	Vrednost (s)
Najkraće / najduže trajanje	166,2 / 499,3
Prvi kvartil / medijana / treći kvartil	180,2 / 225,5 / 287,0
Aritmetička sredina	255,5
Standardna devijacija	96,4

Raspodela je asimetrična udesno: polovina pokušaja završena je za manje od 226 sekundi, dok je najduži trajao nešto preko osam minuta. Razlog je to što agent sam odlučuje koliko će puta pokrenuti i doraditi kod pre nego što proglasi zadatak završenim, a u ovom pokretanju svakom pokušaju prosleđena su tri šablonska kandidata, što povećava količinu materijala koji agent pregleda.

Zbir svih izmerenih trajanja je 3833 sekunde, odnosno oko 64 minuta rada agentskog alata sabranog po varijantama. Pošto su u ovom pokretanju uporedo obrađivana dva posla, a unutar svakog posla sve tri varijante (odeljak 6.3.5), stvarno proteklo vreme je znatno manje: od pokretanja generisanja prvog posla do sklopljene datoteke proteklo je oko osamnaest minuta. Generisanje time traje znatno duže od uobičajene interaktivne operacije, što je neposredna posledica toga što agent za svaku varijantu piše i izvršava sopstveni program; mogućnosti za skraćivanje opisane su u odeljku 8.1.5. Trajanja u ovom prolasku nalaze se pri gornjem kraju raspona zabeleženog na širem skupu, u kome je medijana 113 sekundi (odeljak 7.2); prolazak je i najzahtevniji od zabeleženih, jer jedini traži tri varijante za svaki od pet slajdova.

7.5.3 Osobine generisanog koda

Od 15 radnih prostora, u 12 je sačuvan program koji je agent napisao, dužine 378–792 linije (prosek 592, medijana 608). U preostala tri radna prostora agent je, pošto je varijanta bila gotova i proverena, sam obrisao svoj program kao pomoćni artefakt — upit to ne sprečava, jer traži samo izlaznu prezentaciju. I te tri varijante uspele su iz prvog pokušaja, pa izostanak programa umanjuje samo uzorak za analizu koda, ne i uspešnost generisanja. Pregledom tih dvanaest programa uočava se da svi grade slajd od nule: postavljaju pozadinu, tekstualne okvire i oblike umesto da popunjavaju rezervisana mesta iz šablona; da palete boja i tipografiju izvode iz strukturnog sažetka i slike kandidata; i da veći deo programa čine po-

moćne funkcije koje procenjuju broj redova teksta na osnovu prosečne širine znaka i smanjuju font dok tekst ne stane (prilog C). Ovaj poslednji obrazac je direktna posledica toga što python-pptx ne izračunava prelom teksta (odeljak 2.5): agent sam gradi približan model preloma da bi ispunio zahtev iz upita; njegova tačnost razmatra se u odeljku 7.6.

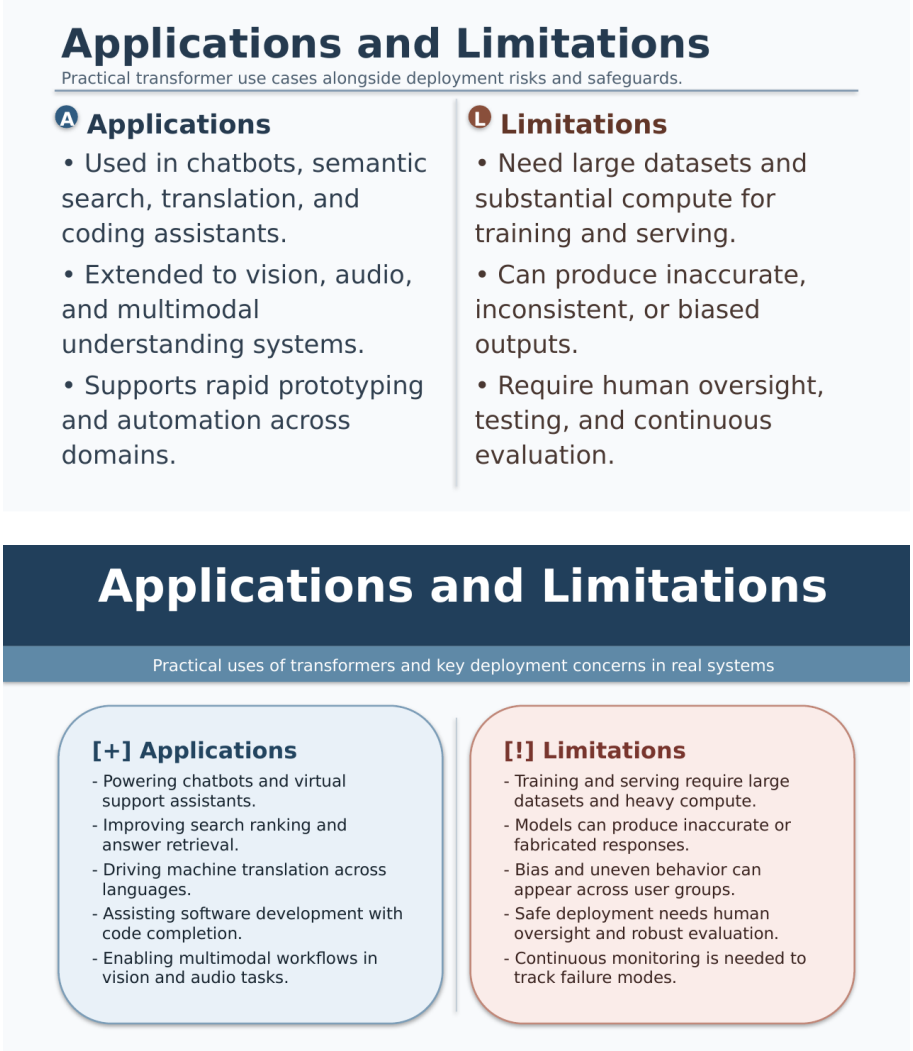
7.6 Kvalitativna analiza generisanih slajdova

Slike svih 15 generisanih varijanti pregledane su i upoređene sa zadatim sadržajem slajda. U svih 15 slučajeva slajd nosi zadati naslov i cilj i pokriva sve ključne tačke iz plana, pa nijedna varijanta nije izostavila zadati sadržaj. Agent te tačke po pravilu ne prenosi doslovno: parafrazira ih, deli na više stavki ili spaja, a u pojedinim varijantama dodaje i tvrdnju koje u planu nema (na primer, stavku o brzoj izradi prototipa na gornjoj varijanti slike 7.1). Takve dopune su tematski prihvatljive i pokazuju da agent plan tumači kao sadržinski okvir, a ne kao doslovan predlog.

Slika 7.1 prikazuje dve varijante istog slajda iz istog posla. Obe nose zadati naslov i cilj, obe pokrivaju isti skup ključnih tačaka i obe preuzimaju tamnoplavu paletu šablona, ali se bitno razlikuju i po broju stavki i po rasporedu: prva sadržaj deli u dve kolone označene kružnim oznakama, dok druga uvodi punu traku zaglavlja i sadržaj smešta u dve zaobljene kartice. Razlika potiče od nedeterminizma modela, svojstva koje sistem ne kontroliše; sistem mu se prilagođava tako što korisniku prikazuje sve varijante i prepušta mu izbor.

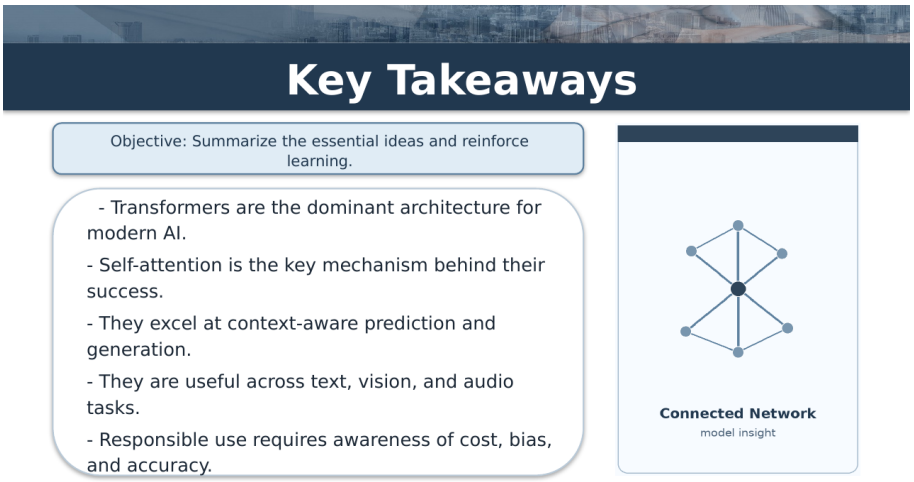
Slika 7.2 prikazuje varijantu drugog slajda iz istog pokretanja. Raspored je uravnotežen, tekst se uklapa u okvire, agent je od osnovnih oblika konstruisao šematski crtež mreže na osnovu opisa vizuelnog rešenja iz plana, a u zaglavlju je zadržao traku sa fotografijom iz šablona, čime je vidljivo očuvan njegov vizuelni identitet.

Odstupanja uočena u pregledu odnose se na precizno uklapanje teksta u uske elemente. Agent sam gradi približan model preloma teksta, jer ga python-pptx ne izračunava (odeljak 2.5), pa se procena zauzetog prostora u pojedinim varijantama razlikuje od stvarnog preloma pri prikazivanju. Isti oblik odstupanja javlja se i na drugim temama i šablonima (odeljak 7.2), pa nije svojstvo ovog prolaska nego postupka. Merenje geometrije rasporeda, opisano u odeljku 8.1.1, odnosi se upravo na tu veličinu.



Slika 7.1: Dve varijante istog slajda iz istog posla, na osnovu istog sadržaja i istog skupa šablonskih kandidata. Obe preuzimaju paletu šablona, ali se razlikuju u organizaciji sadržaja.

Vizuelni identitet šablona dosledno je prenet na generisane slajdove: sve varijante preuzele su tamnoplavu paletu šablona, a nekoliko njih i traku sa fotografijom iz zaglavlja. Tome doprinosi to što strukturni sažetak kandidata iz ovog šablona sadrži geometriju rezervisanih mesta i uzorke teksta (prilog A), pa agent raspolaže merljivim opisom rasporeda, a ne samo slikom. Kod šablona čiji vizuelni elementi potiču iz rasporeda i master slajda isti sažetak ostaje prazan, jer ih čitanje pojedinačnog slajda bibliotekom python-pptx ne otkriva (odeljak 2.5); u takvim slučajevima agent se oslanja prvenstveno na sliku kandidata.



Slika 7.2: Generisan slajd sa uravnoteženim rasporedom i trakom zaglavlja preuzetom iz šablona.

7.7 Sklopljena prezentacija

Iz pet poslova sklopljena je prezentacija od pet slajdova, u redosledu i sa izborom varijanti koje je korisnik zadao u interfejsu. Sklapanje je preuzelo prvu varijantu za prvi slajd, drugu za drugi i treći, a treću za četvrti i peti slajd, čime je potvrđeno da se korisnički izbor prenosi do sklapanja. Gotova datoteka sadrži 61 oblik na pet slajdova, zadržava dimenzije slajda iz šablona i otvara se bez greške bibliotekom python-pptx i programom LibreOffice, kojim je i iscrtana radi pregleda. Oznake probne verzije biblioteke Aspose.Slides uklonjene su postupkom čišćenja opisanim u odeljku 6.4, što je provereno pretragom teksta u izlaznoj datoteci.

Glava 8

Zaključak

Rad opisuje sistem koji iz kratkog opisa teme generiše prezentaciju u formatu PPTX, preuzimajući vizuelni jezik iz postojećeg standardizovanog šablona prezentacije. Osnovni problem, u kome sadržaj mora odgovarati zahtevu korisnika, a oblikovanje zatečenom šablonu, rešen je razdvajanjem posla na pripremu baze šablonskih slajdova, koja se izvršava jednom, i generisanje na zahtev, u kome se za planirani sadržaj bira odgovarajući šablonski slajd, a zatim agentski alat generisanjem i izvršavanjem koda proizvodi slajd koji taj raspored primenjuje na novi sadržaj.

Ciljevi postavljeni u uvodu su ispunjeni. Ocenjivanje predviđeno prijavom teme, po principu *LLM-as-a-judge*, u sistemu je raspoređeno na tri ugrađena mehanizma: rubriku pri izboru šablonskih slajdova, zahteve za samoproveru u upitu agentu i deterministička proveru izlaza; oblik koji bi takvo ocenjivanje imalo kao zaseban sloj izložen je u odeljku 8.1.2. Analiza materijala iz glave 7 pokazala je da sistem u posmatranom uzorku pouzdano proizvodi tehnički ispravan izlaz iz prvog pokušaja, uz generisanje reda veličine nekoliko minuta po varijanti, i da se vizuelni identitet šablona dosledno prenosi na generisane slajdove. Preciznost uklapanja teksta u okvire počiva na približnom modelu preloma koji agent sam gradi, pošto ga python-pptx ne izračunava.

Rad je inženjerskog karaktera; doprinos nije u novom algoritmu, već u projektovanju i implementaciji sistema koji semantičko pretraživanje, multimodalno opisivanje i agentsko generisanje koda povezuje u funkcionalnu celinu, i u analizi njenog ponašanja koja je osnova za dalji razvoj (odeljak 8.1). Analiza pokazuje i gde je težište kvaliteta izlaza: ne u jačini jezičkog modela, već u povratnoj sprezi između generisanog slajda i njegove izmerene vizuelne ispravnosti, čiju osnovu

iscrtavanje pregleda i upit sa opisom prethodne greške u sistemu već čine.

8.1 Pravci daljeg razvoja

Postavka na kojoj je sistem izgrađen otvorena je za nekoliko prirodnih proširenja, koja su u nastavku poređana po odnosu očekivane koristi i obima zahvata.

8.1.1 Programska provera rasporeda

Sistem proverava da je izlazna datoteka ispravna i da sadrži tačno jedan slajd, dok se zahtevi za kvalitetom rasporeda agentu zadaju prirodnim jezikom, kroz zahteve za samoproveru (odeljak 6.3.3).

Ta kontrola prirodno se proširuje merenjem geometrije rasporeda nad iscrtanom slikom ili nad oblicima u izlaznoj datoteci, pri čemu izmereno odstupanje ulazi u ponovni pokušaj kao opis konkretnog problema [5, 15]. Takvo proširenje zatvara povratnu spregu između generisanog slajda i njegove izmerene ispravnosti, a oslanja se na ono što sistem već ima: pregledi se iscrtavaju, a upit agentu prima opis prethodne greške.

8.1.2 Evaluacioni sloj

Ocenjivanje je u sistemu raspoređeno na tri mehanizma: jezički model pri izboru šablonskih slajdova ocenjuje kandidate po eksplicitnoj rubrici (odeljak 6.3.2), upit agentu sadrži zahteve za samoproveru rasporeda (odeljak 6.3.3), a sistem sprovodi determinističu proveru izlaza (odeljak 6.3.4).

Ista uloga može se izdvojiti u zaseban sloj po principu *LLM-as-a-judge*, koji gotovu prezentaciju ocenjuje po sadržaju, dizajnu i koherentnosti, po ugledu na PPTEval [32], i odbija je pre isporuke. Poznate pristrasnosti takvog ocenjivanja [33] takav sloj svrstavaju u dopunu programskoj proveri, a ne u njenu zamenu.

8.1.3 Bogatije razumevanje šablona

Strukturni opis šablonskog slajda obuhvata elemente samog slajda. Proširenje na raspored, master slajd i temu daje agentu potpuniju sliku vizuelnog identiteta, naročito kod šablona kod kojih se pozadine, logotipi i dekorativni oblici nalaze isključivo na tim nivoima (odeljak 2.5).

Srodan pravac je rad sa više šablona istovremeno, uz izbor šablona po prezentaciji, i prosleđivanje izabranog vizuelnog stila iz interfejsa sve do upita agentu. Srodan pristup je izmena postojećeg slajda umesto građenja novog, kako to radi PPTAgent [32]; on po prirodi bolje čuva identitet šablona, po cenu manje slobode u oblikovanju.

8.1.4 Diversifikacija varijanti

Svakoj varijanti prosleđuje se isti skup kandidata i isti sadržaj, pa razlike među njima potiču isključivo od nedeterminizma modela, dakle od svojstva nad kojim sistem nema kontrolu. Kontrolisana raznovrsnost dobila bi se eksplicitnim zadavanjem različitih rasporeda ili stilskih smernica po varijanti, čime bi razlike među ponuđenim rešenjima postale namerne i predvidive.

8.1.5 Performanse i skaliranje

Generisanje jedne varijante traje u proseku oko 256 sekundi (odeljak 7.5.2), a podrazumevano se generišu tri po slajdu. Skraćenje je moguće povećanjem stepena uporednosti, kraćim upitom sa manjim brojem kandidata i keširanjem rezultata za ponovljene teme.

Na nivou sistema, prenos stanja poslova iz memorije procesa u trajno skladište omogućio bi horizontalno skaliranje, oporavak posle restarta i uklanjanje završenih poslova. Pretraga po vektorskoj sličnosti sada je linearan prolazak kroz sve zapise u aplikativnom sloju, što je prihvatljivo za šablone od nekoliko desetina slajdova; za veće baze primenjuje se ugrađeno vektorsko indeksiranje [17].

8.1.6 Priprema za proizvodno okruženje

Prelazak iz istraživačke u proizvodnu postavku obuhvata dva zahvata. Prvi je izolacija agentskog alata, u kontejneru bez pristupa mreži i tajnama, umesto uputstva agentu da ne menja datoteke izvan radnog direktorijuma (odeljak 6.3.3). Drugi je autentifikacija i autorizacija korisnika: u istraživačkoj postavci servis radi sa unapred određenim korisnikom u administratorskoj ulozi, dok prijava i korisničke uloge administrativne operacije, poput brisanja šablona, vezuju za ovlašćenog korisnika.

Literatura

[1] Aspose Pty Ltd. Licensing — Aspose.Slides for Python via .NET documentation, 2026. https://docs.aspose.com/slides/python-net/licensing/.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. Curran Associates, Inc., 2020. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[3] Steve Canny. python-pptx documentation, 2024. https://python-pptx.readthedocs.io/.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. Technical report, OpenAI, 2021. arXiv preprint arXiv:2107.03374. https://arxiv.org/abs/2107.03374.

[5] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to Self-Debug. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024. https://openreview.net/forum?id=KuPixIqPiq.

[6] Docker, Inc. Docker documentation, 2026. https://docs.docker.com/.

[7] Ecma International. ECMA-376: Office Open XML file formats, 5th edition, 2021. https://ecma-international.org/publications-and-standards/standards/ecma-376/.

[8] Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. DOC2PPT: Automatic presentation slides generation from scientific documents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):634–642, 2022. DOI: https://doi.org/10.1609/aaai.v36i1.19943.

[9] Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, and Trevor Darrell. AutoPresent: Designing structured visuals from scratch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2025)*, 2025. https://openaccess.thecvf.com/content/CVPR2025/papers/Ge_AutoPresent_Designing_Structured_Visuals_from_Scratch_CVPR_2025_paper.pdf.

[10] Google. Angular documentation, 2026. https://angular.dev/.

[11] Yue Hu and Xiaojun Wan. PPSGen: Learning-based presentation slides generation for academic papers. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1085–1097, 2015. DOI: https://doi.org/10.1109/TKDE.2014.2359652.

[12] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023. Article 248. DOI: https://doi.org/10.1145/3571730.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 9459–9474. Curran Associates, Inc., 2020. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[14] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023. Article 195. DOI: https://doi.org/10.1145/3560815.

[15] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, et al. Self-Refine: Iterative refinement with Self-Feedback. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates, Inc., 2023. https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

[16] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008. https://nlp.stanford.edu/IR-book/.

[17] Microsoft. Integrated vector database in Azure Cosmos DB, 2026. https://learn.microsoft.com/en-us/azure/cosmos-db/vector-database.

[18] Microsoft. Introduction to blob (object) storage — Azure Storage, 2026. https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction.

[19] MongoDB, Inc. MongoDB manual, 2026. https://www.mongodb.com/docs/manual/.

[20] OpenAI. Codex CLI — official documentation, 2026. https://learn.chatgpt.com/docs/codex/cli.

[21] OpenAI. Embeddings — OpenAI API documentation, 2026. https://developers.openai.com/api/docs/guides/embeddings.

[22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. Curran Associates, Inc., 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

[23] Poppler Developers. Poppler: PDF rendering library and command-line utilities, 2026. https://poppler.freedesktop.org/.

[24] Pydantic Services Inc. Pydantic documentation, 2026. https://docs.pydantic.dev/.

[25] Sebastián Ramírez. FastAPI documentation, 2026. https://fastapi.tiangolo.com/.

[26] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992. Association for Computational Linguistics, 2019. DOI: https://doi.org/10.18653/v1/D19-1410.

[27] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? Investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14918–14937. Association for Computational Linguistics, 2023. DOI: https://doi.org/10.18653/v1/2023.emnlp-main.923.

[28] The Document Foundation. Starting the LibreOffice software with parameters — LibreOffice help, 2026. https://help.libreoffice.org/latest/en-US/text/shared/guide/start_parameters.html.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[30] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. Curran Associates, Inc., 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023. https://openreview.net/forum?id=WE_vluYUL-X.

[32] Hao Zheng, Xinyan Guan, Hao Kong, Wenkai Zhang, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. PPTAgent: Generating and evaluating presentations beyond text-to-slides. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14402–14418. Association for Computational Linguistics, 2025. https://aclanthology.org/2025.emnlp-main.728/.

[33] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*. Curran Associates, Inc., 2023. `https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html`.

Prilog A

Primer upita i slajdova nastalih po njemu

U prilogu je prikazan reprezentativni, skraćeni isečak upita koji sistem prosleđuje agentskom alatu. Prikazani su samo delovi koji su relevantni za analizu u odeljcima 6.3.3 i 7.6.

```
Generate exactly one-slide PPTX file.
Write output file to this exact path: variant_01.pptx

Variant target: 1/3
Requirements:
- Use Python with python-pptx.
- Create exactly one slide.
- Use all provided template assets as inspiration and guidance.
- Do not copy any one template directly.
- No text overflow is permitted.
- No elements may overlap.
- If text overflow is detected, you must programmatically reduce font size
  or increase bounding box dimensions until text fits.
- Do not ask for approval.
- Do not modify files outside the current working directory.

Slide content JSON:
{
  "slide_index": 4,
  "title": "Applications and Limitations",
  "objective": "Present practical uses of transformers and key concerns in
               their deployment.",
  "key_points": [
    "Used in chatbots, search, translation, and coding tools.",
    "Applied in vision, audio, and multimodal systems.",
    "Require large datasets and significant compute resources.",
```

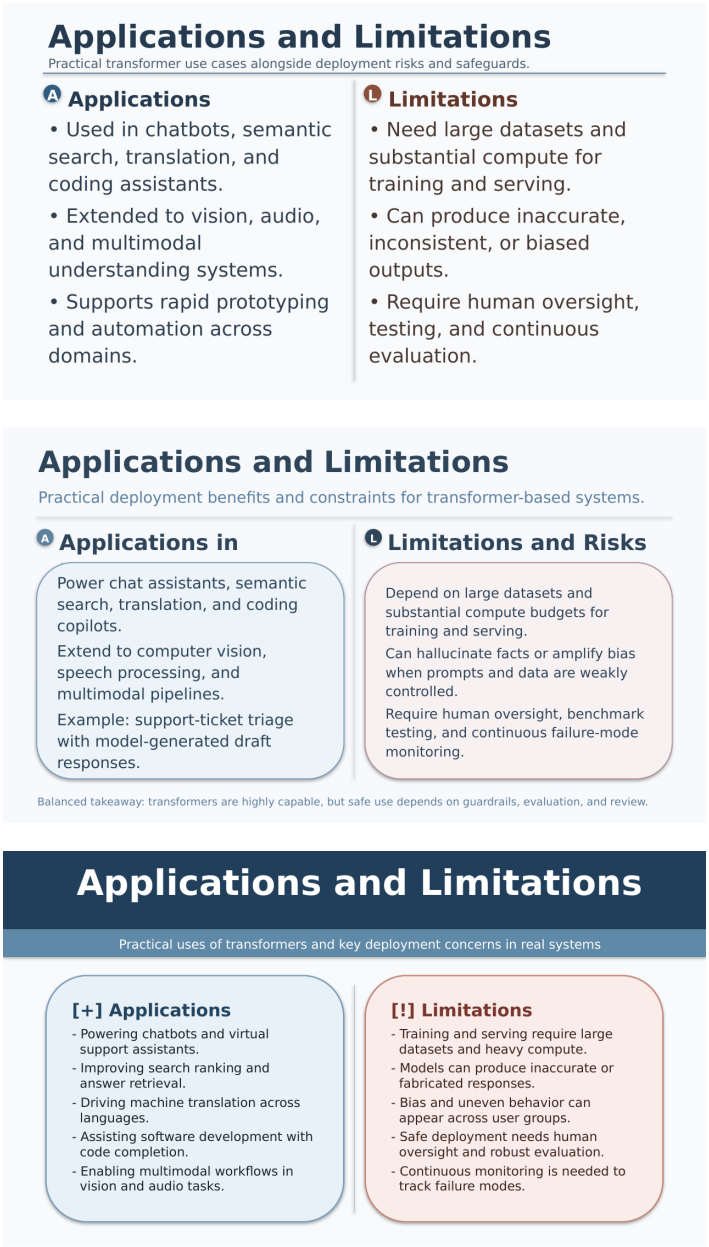
```
    "Can generate inaccurate or biased outputs.",
    "Need human oversight and evaluation."
  ],
  "visual_brief": "Two-column comparison slide: applications on one side,
                   limitations on the other.",
  "style_hints": ["academic", "balanced framing", "clear contrast"]
}

Template candidates (use all of them):
- template_id: test_template.pptx
  pptx_path: <workspace>/candidate_01_test_template_pptx.pptx
  image_path: <workspace>/candidate_01_test_template_pptx.png
  summary_json: {
    "slide_size": {"width": 9144000, "height": 5143500},
    "slides": [{
      "layout_name": "Comparison",
      "placeholders": [
        {"name": "Title 3", "placeholder_type": 1,
         "left": 448964, "top": 1, "width": 7787955, "height": 891994},
        {"name": "Text Placeholder 4", "placeholder_type": 2,
         "left": 536877, "top": 1655520, "width": 4040188, "height": 458115},
        {"name": "Content Placeholder 5", "placeholder_type": 7,
         "left": 536877, "top": 2113635, "width": 4040188, "height": 2290575}
      ],
      "shapes": [
        {"name": "Title 3", "text_sample": "Slide Title"},
        {"name": "Text Placeholder 4", "text_sample": "Product A"},
        {"name": "Content Placeholder 5",
         "text_sample": "Feature 1\nFeature 2\nFeature 3"}
      ]
    }]
  }
```

Kod A.1: Skraćeni primer upita prosleđenog agentskom alatu.

Isečak pokazuje da agent, pored sadržaja slajda, dobija i geometriju rezervisanih mesta izraženu u jedinicama EMU, nazive rasporeda i uzorke teksta iz šablona. Za šablon korišćen u glavi 7 strukturni sažetak je popunjen, pa agent ima merljiv opis rasporeda; kod šablona čiji se vizuelni elementi nalaze isključivo u rasporedu i master slajdu isti sažetak ostaje prazan, jer ih čitanje pojedinačnog slajda bibliotekom python-pptx ne otkriva (odeljak 2.5).

Slika A.1 prikazuje sve tri varijante slajda koje je agent proizveo po ovom upitu. Sadržaj im je isti, jer potiče iz istog plana, a razlikuju se u načinu na koji je organizovan: prva ga deli u dve kolone sa nabrajanjem, druga u dve zaobljene kartice sa zaključnom rečenicom u podnožju, a treća uvodi punu traku zaglavlja u tamnoplavoj boji šablona. Prva i treća varijanta analiziraju se u odeljku 7.6.



Slika A.1: Tri varijante slajda koje je agent proizveo po upitu prikazanom u ovom prilogu, redom prva, druga i treća.

Prilog B

Primer opisa šablonskog slajda

U nastavku je prikazan skraćeni zapis četvrtog slajda šablona `test_template.pptx`, onako kako se čuva u bazi (odeljak 7.4). Vektorska reprezentacija i identifikatori datoteka su izostavljeni.


```
{
  "slide_id": "8f3c1a04-2d77-4e19-9b52-6c0ae4d18f77",
  "template_name": "test_template.pptx",
  "slide_index": 4,
  "is_default": true,
  "business_description": {
    "business_description": "A simple side-by-side product comparison slide
      for presenting two offerings, options, or solutions with a short list
      of key features under each.",
    "suggested_use_cases": [
      "compare Product A vs. Product B",
      "summarize feature differences between two options"
    ],
    "data_suitability": [
      "product or solution names with a few bullet-point features",
      "short text summaries rather than charts or detailed metrics"
    ],
    "content_constraints": [
      "Works best with two items only",
      "Not suited for quantitative charts, tables, or KPI reporting"
    ]
  },
  "technical_description": {
    "titles_count": 1,
    "paragraphs_count": 8,
    "text_boxes_count": 5,
    "charts_count": 0,
    "tables_count": 0,
    "images_count": 0
```

```
  },
  "flags": {
    "is_instructional_slide": false,
    "has_visual_content": false
  }
}
```

Kod B.1: Skraćeni zapis šablonskog slajda iz baze.

Prilog C

Isečak koda koji je generisao agent

Funkcija za procenu broja redova koje će tekst zauzeti u okviru zadate širine, iz programa koji je agent napisao za drugu varijantu prvog slajda analiziranog pokretanja (odeljak 7.5.3), karakterističan je primer približnog modela preloma teksta, izgrađenog jer ga python-pptx ne izračunava. Komentar u kodu napisao je sam agent.

```
def measure_wrapped_line_count(text: str, font_pt: float, width_pt: float) ->
    int:
    # Heuristic average character width; good enough for
    # deterministic overflow guard.
    avg_char_w = max(1.0, font_pt * 0.52)
    max_chars = max(1, int(width_pt / avg_char_w))

    total_lines = 0
    for raw_line in text.split("\n"):
        line = raw_line.strip("\r")
        if not line:
            total_lines += 1
            continue

        words = line.split()
        current = 0
        lines = 1
        for w in words:
            word_len = len(w)
```

```
19            if current == 0:
20                current = word_len
21            elif current + 1 + word_len <= max_chars:
22                current += 1 + word_len
23            else:
24                lines += 1
25                current = word_len
26        total_lines += lines
27
28    return total_lines
```

Kod C.1: Procena broja redova pri prelomu teksta, iz programa koji je generisao agent.

Dobijeni broj redova množi se visinom reda i poredi sa raspoloživom visinom okvira, a font se smanjuje dok tekst ne stane. Pretpostavka da je prosečna širina znaka nešto više od polovine veličine fonta aproksimacija je koja zavisi od konkretnog pisma i sadržaja, pa je upravo ona veličina na koju se odnosi merenje rasporeda opisano u odeljku 8.1.1.

Biografija autora

Katarina Milošević završila je osnovne akademske studije informatike na Matematičkom fakultetu Univerziteta u Beogradu, gde je upisala i master studije informatike (broj indeksa 1019/2023). Profesionalno se bavi razvojem softvera, sa fokusom na primenu velikih jezičkih modela u poslovnim aplikacijama.