**VIŠJA STROKOVNA ŠOLA ACADEMIA**

**MARIBOR**

# INTELIGENTNI SISTEM ZA IZVAJANJE TEHNIČNIH INTERVJUJEV Z AVTOMATIZIRANO ANALIZO KANDIDATOV

Kandidatka: Kristijana Marušić

Vrsta študija: študentka izrednega študija

Študijski program: Računalništvo in informatika

Mentor predavatelj: mag. Matjaž Prtenjak

Mentor v podjetju: Damjan Zajc, dipl. inž. rač. in inf.

Lektor: Špela Fele, univ. dipl. slov.

Maribor, 2026

# Izjava o avtorstvu diplomskega dela

Podpisana Kristijana Marušić sem avtorica diplomskega dela z naslovom Inteligentni sistem za izvajanje tehničnih intervjujev z avtomatizirano analizo kandidatov, ki sem ga napisala pod mentorstvom mag. Matjaža Prtenjaka.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbela, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, oktober 2026 Podpis študenta:

*I would like to express my sincere gratitude to my academic supervisor, mag. Matjaž Prtenjak, for his professional guidance and advice throughout the preparation of this thesis and the development of this project. I am also deeply grateful to the engineering team at RTV Slovenia — including my mentors and colleagues — for their trust, guidance, and the opportunity to work on meaningful engineering projects. The experience I gained during my time at RTV Slovenia has been invaluable and has significantly contributed to my professional growth as a software engineer. A special thank you goes to my high school mathematics teacher for teaching me that mathematics was never something to be afraid of. Her passion for teaching inspired me to believe in my own abilities and played an important role in shaping my path toward computer science.*

*Finally, my deepest and most heartfelt gratitude goes to my parents, my sister, my grandparents and my uncle. Thank you for always being there for me. This thesis is dedicated to them.*

*Kristijana Marušić, 2026*

# Povzetek

Tehnični intervjuji za programsko-inženirska delovna mesta pogosto vključujejo več ločenih orodij, ocenjevanje kandidatovih rešitev pa je lahko odvisno od subjektivne presoje intervjuvarja. Namen diplomskega dela je bil razviti prototip sistema, ki v enoten delovni proces povezuje video intervju, reševanje programerske naloge in avtomatizirano analizo kandidatove predaje z velikim jezikovnim modelom. Odjemalski del sistema je implementiran z ogrodjem `Next.js`, knjižnico `React` in jezikom `TypeScript`. Platforma `Convex` zagotavlja podatkovno bazo in zaledno logiko, `Clerk` avtentikacijo uporabnikov, `Stream` pa video komunikacijo. Kandidat reševanje naloge opravi v štirih korakih: seznani se z zahtevami, pripravi načrt, napiše programsko kodo in razloži rešitev. Kodo ureja v urejevalniku `Monaco`, za preverjanje njenega izhoda jo lahko prek strežniške poti izvede v ločenem okolju storitve `Piston`. Načrt, koda, razlaga in morebitni rezultat izvajanja se med intervjujem samodejno shranjujejo v zapis `submissions` v podatkovni bazi `Convex`. Ko intervjuvar zaključi intervju, sistem predajo označi kot dokončno in asinhrono sproži analizo. Zaledni del iz shranjenih podatkov sestavi strukturiran poziv, ki poleg kandidatove predaje vsebuje opis naloge, njeno težavnost, omejitve in merila ocenjevanja. Odgovor `LLM` v obliki `JSON` preveri glede na zahtevano strukturo in ga shrani kot poročilo v zapis `aiReports`. Poročilo vsebuje področne ocene, obrazložitve, prednosti, področja za izboljšanje in priporočila, skupno oceno zaledni del izračuna iz štirih področnih ocen. Kandidat in intervjuvar lahko poročilo pregledata v aplikaciji ter ga preneseta v obliki `PDF`. Funkcionalno, scenarijsko in uporabniško testiranje s `15` udeleženci je pokazalo, da prototip podpira celoten potek simuliranega tehničnega intervjuja in ustvarja povratne informacije, prilagojene vsebini kandidatove predaje. Rezultati podpirajo izvedljivost razvite rešitve v simuliranem okolju, za presojo zanesljivosti avtomatiziranih ocen pa bi bilo potrebno obsežnejše testiranje s strokovnimi ocenjevalci. Analiza temelji na kandidatovi predaji in kontekstu naloge, pri čemer uspešno izvajanje kode samo po sebi ne zagotavlja pravilnosti rešitve. Sistem je zasnovan kot podpora intervjuvarju, ki ostaja odgovoren za končno presojo kandidata.

**Ključne besede**: veliki jezikovni modeli, umetna inteligenca, tehnični intervjuji, ocenjevanje kandidatov, povratne informacije, programsko inženirstvo.

# Abstract

**Intelligent System for Conducting Technical Interviews with Automated Candidate Analysis**

Technical interviews for software engineering positions often involve several separate tools, while the assessment of candidates' solutions may depend on the interviewer's subjective judgement. The aim of this thesis was to develop a system prototype that integrates video interviews, programming tasks and automated analysis of candidate submissions using a large language model into a single workflow. The client side is implemented with `Next.js`, `React` and `TypeScript`. `Convex` provides the database and backend logic, `Clerk` handles user authentication, and `Stream` enables video communication. Candidates complete a task in four stages: reviewing its requirements, preparing a plan, writing code and explaining their solution. They write code in the `Monaco` editor and can execute it through a server route in the isolated `Piston` environment to inspect its output. The candidate's plan, code, explanation and any execution result are automatically saved to a `submissions` record in the `Convex` database during the interview. When the interviewer ends the interview, the system finalises the submission and starts an asynchronous analysis. The backend constructs a structured prompt from the stored submission, the task description, its difficulty, constraints and assessment criteria. It validates the `LLM's JSON` response against the required structure and stores the result in an `aiReports` record. The report contains scores for four assessment areas, explanations, strengths, areas for improvement and recommendations. The backend calculates the overall score from the four area scores. Both the candidate and the interviewer can view the report in the application and download it as a `PDF`. Functional, scenario-based and user testing with `15` participants demonstrated a complete simulated interview workflow and submission-specific feedback. The results support the solution's feasibility, but assessing automated score reliability requires broader testing with expert evaluators. Analysis uses the submission and task context; successful code execution alone does not guarantee correctness. The system assists the interviewer, who retains responsibility for the final candidate assessment.



**Keywords:** Large Language Models, Artificial Intelligence, Technical Interviews, Candidate Evaluation, Feedback, Software Engineering.

**KAZALO**

**1 Uvod 10**

1.1 Opis področja in opredelitev problema ... 10
1.2 Namen, cilji in osnovne trditve ... 11
1.3 Predpostavke in omejitve ... 11
1.4 Uporabljene raziskovalne metode ... 12

**2 Teoretična izhodišča 13**

2.1 Tehnični intervjuji v postopkih zaposlovanja ... 13
2.2 Subjektivnost pri ocenjevanju kandidatov ... 14
2.3 Umetna inteligenca v kadrovskih procesih ... 16
2.4 Veliki jezikovni modeli (LLM) ... 18
2.5 Na človeka usmerjeni digitalni sistemi ... 22

**3 Analiza in načrtovanje sistema 23**

3.1 Analiza zahtev ... 23
3.2 Arhitektura sistema ... 24
3.3 Uporabniške vloge in funkcionalnosti ... 26
3.4 Načrt podatkovnega modela ... 26
3.5 Načrt uporabniškega vmesnika ... 30

**4 Implementacija sistema 32**

4.1 Uporabljene tehnologije ... 32
4.2 Avtentikacija in avtorizacija uporabnikov ... 35
4.2.1 Povezava storitev Clerk in Convex ... 35
4.2.2 Sinhronizacija uporabnikov s podatkovno bazo ... 36
4.2.3 Uporabniške vloge in preverjanje dostopa ... 38
4.3 Video komunikacija ... 39
4.3.1 Varna inicializacija video odjemalca ... 39
4.3.2 Ustvarjanje takojšnjega intervjuja ... 41
4.3.3 Pridružitev video seji in nadzor dostopa ... 42
4.3.4 Intervjujski delovni prostor ... 43
4.3.5 Zaključek video seje ... 45

4.4 Programersko okolje za reševanje nalog ..... 46
4.4.1 Izbira naloge in programskega jezika ..... 47
4.4.2 Urejanje in samodejno shranjevanje rešitve ..... 48
4.4.3 Izvajanje programske kode ..... 50
4.4.4 Dokončna oddaja rešitve ..... 52
4.5 Integracija velikih jezikovnih modelov ..... 53
4.5.1 Začetek postopka analize ..... 53
4.5.2 Priprava konteksta in poziva ..... 55
4.5.3 Strukturiranje in preverjanje rezultata ..... 57
4.5.4 Obravnava napak in nadzor dostopa ..... 59
4.5.5 Odgovorna uporaba umetne inteligence ..... 59
4.6 Generiranje poročil in povratnih informacij ..... 60
4.6.1 Generiranje poročila PDF ..... 61
**5 Testiranje in evalvacija 62**
5.1 Predstavitev testnih primerov ..... 62
5.2 Preverjanje hipotez ..... 63
5.3 Analiza rezultatov ..... 65
5.4 Prednosti in omejitve sistema ..... 65
**6 SKLEP ..... 66**

## KAZALO SLIK

Slika 2.1 Osnovne arhitekture velikih jezikovnih modelov. ..... 20
Slika 2.2 Uporaba velikega jezikovnega modela pri analizi tehničnega intervjuja. ..... 21
Slika 3.1 Arhitektura razvitega sistema – odjemalski del, zaledni del in zunanje storitve. ..... 25
Slika 3.2 Shema Convex podatkovne baze. ..... 29
Slika 3.3 Začetna stran intervjuvarja. ..... 30
Slika 3.4 Začetna stran in nadzorna plošča kandidata. ..... 31
Slika 4.1 Intervjujski delovni prostor z video komunikacijo in tehnično nalogo. ..... 44
Slika 4.2 Urejevalnik programske kode s prikazom rezultata izvajanja. ..... 52
Slika 4.3 Prikaz strukturiranega poročila o uspešnosti kandidata. ..... 60
Slika 4.4 Poročilo o analizi kandidata v obliki PDF. ..... 61

## KAZALO TABEL

Tabela 4.1 Pregled uporabljenih tehnologij. ..... 34
Tabela 5.1 Rezultati funkcionalnega in scenarijskega testiranja. ..... 63

## KAZALO PSEVDOKOD

Psevdokoda 4.1 Preverjanje webhooka in sinhronizacija novega uporabnika. ..... 37
Psevdokoda 4.2 Varno generiranje uporabniškega žetona Stream. ..... 40
Psevdokoda 4.3 Ustvarjanje takojšnjega intervjuja. ..... 41
Psevdokoda 4.4 Preverjanje dostopa do intervjuja. ..... 43
Psevdokoda 4.5 Zaključevanje intervjuja in začetek generiranja poročila. ..... 45
Psevdokoda 4.6 Samodejno shranjevanje osnutka kandidatove rešitve. ..... 49
Psevdokoda 4.7 Preverjanje in izvajanje programske kode. ..... 51
Psevdokoda 4.8 Začetek analize kandidatove predaje. ..... 54
Psevdokoda 4.9 Generiranje, preverjanje in shranjevanje strukturirane analize. ..... 58

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| AI | Artificial Intelligence | umetna inteligenca |
| API | Application Programming Interface | programski vmesnik aplikacije |
| GPT | Generative Pre-trained Transformer | generativni predhodno naučeni model na osnovi arhitekture Transformer |
| HTML | HyperText Markup Language | označevalni jezik za oblikovanje spletnih strani |
| JWT | JSON Web Token | spletni žeton JSON |
| LLM | Large Language Model | veliki jezikovni model |
| NAT | Network Address Translation | prevajanje omrežnih naslovov |
| OAuth | Open Authorization | protokol za odprto avtorizacijo |
| RNN | Recurrent Neural Networks | rekurentne nevronske mreže |
| SDK | Software Development Kit | razvojni komplet programske opreme |
| WebRTC | Web Real-Time Communication | spletna komunikacija v realnem času |

# 1 Uvod

## 1.1 Opis področja in opredelitev problema

Digitalizacija poslovnih procesov pomembno vpliva na področje zaposlovanja, kjer organizacije vse pogosteje uporabljajo umetno inteligenco za podporo kadrovskim postopkom. Razvoj velikih jezikovnih modelov (LLM) je omogočil razvoj inteligentnih sistemov, ki omogočajo generiranje vprašanj, analizo odgovorov kandidatov in pripravo personaliziranih povratnih informacij. Takšni sistemi prispevajo k učinkovitejši izvedbi intervjujev in boljši podpori kandidatom pri pripravi na zaposlitveni postopek (Singh & Kumar, 2025; Tigani, 2025).

Kljub hitremu razvoju digitalnih rešitev tehnični intervjuji ostajajo eden najpomembnejših načinov preverjanja znanja kandidatov za programsko-inženirska delovna mesta. Podjetja si prizadevajo izboljšati učinkovitost ocenjevanja kandidatov in zmanjšati vpliv subjektivnih dejavnikov pri sprejemanju odločitev. Hkrati postaja vse pomembnejša tudi kakovost povratnih informacij, ki kandidatom omogočajo prepoznavanje področij za nadaljnji razvoj (Tigani, 2025). Eno od obetavnih rešitev predstavlja razvoj inteligentnih sistemov, ki združujejo video komunikacijo, programsko okolje za reševanje tehničnih nalog in uporabo velikih jezikovnih modelov za avtomatizirano analizo odgovorov kandidatov. Takšni sistemi omogočajo bolj strukturiran postopek tehničnega intervjuvanja in pripravo podrobnejših povratnih informacij, ki lahko koristijo tako kandidatom kot tudi ocenjevalcem (Brintha, Vardhan, Kumar, Babu, & Raj, 2025; Singh & Kumar, 2025).

## 1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je razviti prototip inteligentnega sistema, ki združuje izvedbo tehničnega intervjuja, reševanje programerskih nalog in samodejno analizo kandidatove uspešnosti. Sistem ne nadomešča intervjuvarja, temveč podpira njegovo presojo in kandidatu zagotovi strukturirano povratno informacijo in priporočila za izboljšanje. Uporabnost prototipa je ovrednotena s funkcionalnim in scenarijskim testiranjem predaj različnih ravni kakovosti.

Na tej podlagi so bile oblikovane naslednje hipoteze:

- **H1:** Inteligentni sistem omogoča učinkovitejšo izvedbo tehničnih intervjujev kot tradicionalni pristopi.
- **H2:** Integracija velikih jezikovnih modelov omogoča pripravo kakovostnejših in bolj strukturiranih povratnih informacij.
- **H3:** Uporaba avtomatizirane analize zmanjšuje vpliv subjektivnih dejavnikov pri ocenjevanju kandidatov.
- **H4:** Uporaba inteligentnega sistema izboljšuje uporabniško izkušnjo kandidatov in povečuje transparentnost tehničnega intervjuja.
- **H5:** Integracija video komunikacije in programerskega okolja omogoča učinkovito izvedbo tehničnega intervjuja znotraj enotne spletne platforme.

## 1.3 Predpostavke in omejitve

Pri izdelavi diplomskega dela predpostavljamo, da so sodobni modeli umetne inteligence dovolj zmogljivi za analizo tehničnih odgovorov kandidatov in pripravo kakovostnih in uporabnih povratnih informacij. Predpostavljamo tudi, da lahko avtomatizirana analiza prispeva k bolj objektivnemu in doslednemu ocenjevanju kandidatov v primerjavi z izključno ročnim vrednotenjem.

Diplomsko delo je omejeno na razvoj prototipa spletne aplikacije za izvajanje tehničnih intervjujev s programerskimi nalogami in analizo rezultatov z uporabo umetne inteligence. Raziskava se osredotoča predvsem na področje razvoja programske opreme, zato rezultatov ni mogoče neposredno posplošiti na druge vrste zaposlitvenih razgovorov. Prav tako kakovost povratnih informacij temelji na zmogljivostih uporabljenih modelov umetne inteligence in kakovosti vhodnih podatkov, zato popolne objektivnosti ni mogoče zagotoviti. Zaradi časovnih omejitev diplomskega dela sistem ne bo validiran na velikem številu kandidatov v dejanskih zaposlitvenih postopkih, temveč bo njegova uporabnost prikazana na izbranih primerih uporabe.

## 1.4 Uporabljene raziskovalne metode

Diplomsko delo združuje analizo strokovne in znanstvene literature, razvoj prototipa ter njegovo funkcionalno, scenarijsko in uporabniško testiranje. Pri uporabniškem testiranju je sodelovalo 15 oseb, ki so v simuliranem okolju izvedle tehnični intervju. Testiranje ni potekalo v okviru dejanskih zaposlitvenih postopkov. Z njim so bili preverjeni izvedljivost celotnega postopka, delovanje ključnih funkcionalnosti in odzivanje sistema na različne kandidatove predaje.

# 2 Teoretična izhodišča

## 2.1 Tehnični intervjuji v postopkih zaposlovanja

Tehnični intervju predstavlja enega od ključnih korakov v postopku zaposlovanja kandidatov za programsko-inženirska delovna mesta (angl. *Software Engineering, SWE*). Njegov osnovni namen je preveriti, ali kandidat poleg teoretičnega znanja razpolaga tudi s praktičnimi sposobnostmi reševanja problemov, logičnega razmišljanja in uporabe ustreznih tehnologij ter programskih jezikov v realnih situacijah. Za razliko od klasičnega zaposlitvenega razgovora, ki je usmerjen predvsem v predstavitev delovnih izkušenj in osebnostnih lastnosti kandidata, tehnični intervju ocenjuje strokovne kompetence, način razmišljanja ter sposobnost učinkovitega reševanja tehničnih izzivov (Brintha, Vardhan, Kumar, Babu, & Raj, 2025).

V praksi tehnični intervjuji najpogosteje vključujejo reševanje programerskih nalog, algoritmičnih problemov, odpravljanje napak v izvorni kodi, razlago arhitekturnih rešitev ali analizo že obstoječih programskih rešitev. Intervju se lahko izvaja v živo, preko videokonference ali v obliki samostojnega reševanja nalog na spletnih platformah. Ne glede na obliko izvedbe ostaja cilj enak – pridobiti čim bolj objektivno oceno kandidatovega tehničnega znanja, sposobnosti analitičnega razmišljanja in pristopa k reševanju problemov (Singh & Kumar, 2025).

Zaradi velike konkurence na področju informacijskih tehnologij podjetja tehničnim intervjujem namenjajo vse več pozornosti, saj želijo poleg preverjanja programerskega znanja oceniti tudi sposobnost komunikacije, argumentiranja tehničnih odločitev in prilagajanja novim tehnologijam. Kljub temu pa se v praksi pogosto pojavljajo razlike v načinu izvedbe intervjujev in kriterijih ocenjevanja, kar lahko vpliva na enotnost ter pravičnost postopka izbire kandidatov (Reddy, Priya, Vasavi, Ramesh, & Venu Gopal, 2026; Tigani, 2025).

## 2.2 Subjektivnost pri ocenjevanju kandidatov

Ocenjevanje kandidatov med tehničnimi intervjuji predstavlja kompleksen proces, saj poleg preverjanja tehničnega znanja vključuje tudi presojo načina razmišljanja, komunikacijskih sposobnosti, pristopa k reševanju problemov in drugih kompetenc, ki jih ni vedno mogoče objektivno izmeriti. Zaradi tega lahko na končno oceno vplivajo izkušnje, pričakovanja in osebna presoja posameznega intervjuvarja, kar lahko vodi do različnih ocen enako usposobljenih kandidatov (Tigani, 2025).

Eden od pomembnejših razlogov za subjektivnost je pomanjkanje enotnih meril ocenjevanja. Različni intervjuvarji lahko isto rešitev ali odgovor razumejo in ovrednotijo različno glede na svoje tehnično znanje, delovne izkušnje ali preference pri načinu reševanja problemov. Poleg tega lahko na končno odločitev vplivajo tudi dejavniki, ki niso neposredno povezani s strokovnim znanjem kandidata, kot so način komunikacije, stopnja samozavesti ali splošni vtis, ki ga kandidat naredi med intervjujem. Posledično se lahko zgodi, da kandidati z enako stopnjo znanja prejmejo različne ocene, kar zmanjšuje doslednost in pravičnost zaposlitvenega postopka (Reddy, Priya, Vasavi, Ramesh, & Venu Gopal, 2026). Pri tem je pomembno poudariti, da tovrstni dejavniki niso sami po sebi nezaželeni ali nepomembni – nasprotno, pri končni zaposlitveni odločitvi imajo pogosto ključno vlogo, saj delodajalci iščejo kandidate, ki poleg strokovnega znanja izkazujejo tudi dobre medosebne veščine, motivacijo in sposobnost timskega sodelovanja. Problem nastane, kadar ti dejavniki nehote ali nezavedno vplivajo na oceno tehničnega znanja, namesto da bi se presojali ločeno in namensko. Zato je namen sistema, obravnavanega v tem diplomskem delu, izključno objektivizacija vrednotenja tehničnega znanja, medtem ko presoja preostalih, enako pomembnih kompetenc ostaja v domeni kadrovskih strokovnjakov in tehničnih ocenjevalcev.

Dodaten izziv predstavlja pomanjkanje kakovostnih povratnih informacij. Po opravljenem tehničnem intervjuju kandidati pogosto prejmejo zgolj obvestilo o uspehu ali neuspehu, brez podrobnejše obrazložitve sprejete odločitve, v nekaterih primerih pa ne prejmejo nobenega odgovora. Posledično težko ugotovijo, katera področja njihovega znanja ali pristopa k reševanju nalog bi bilo treba izboljšati, kar otežuje njihovo nadaljnjo pripravo na prihodnje zaposlitvene razgovore. Zagotavljanje strukturiranih, jasnih in uporabnih povratnih informacij zato predstavlja pomemben korak k večji transparentnosti zaposlitvenega postopka ter učinkovitejšemu strokovnemu razvoju kandidatov (Brintha, Vardhan, Kumar, Babu, & Raj, 2025; Singh & Kumar, 2025).

V zadnjih letih se kot ena od možnih rešitev vse pogosteje omenja uporaba umetne inteligence za podporo procesu ocenjevanja kandidatov. Umetna inteligenca lahko pripomore k bolj dosledni analizi odgovorov, uporabi enotnih kriterijev vrednotenja in pripravi podrobnih in personaliziranih povratnih informacij. Pri tem njen namen ni nadomestiti tehničnih ocenjevalcev, temveč podpreti njihovo odločanje, zmanjšati vpliv subjektivne presoje in kandidatom zagotoviti kakovostnejšo povratno informacijo o njihovem znanju in kompetencah (Reddy, Priya, Vasavi, Ramesh, & Venu Gopal, 2026; Tigani, 2025).

## 2.3 Umetna inteligenca v kadrovskih procesih

Umetna inteligenca v zadnjih letih pomembno spreminja področje upravljanja s človeškimi viri in postaja sestavni del številnih kadrovskih procesov. Organizacije jo uporabljajo za avtomatizacijo administrativnih opravil, pregled življenjepisov, iskanje ustreznih kandidatov, razvrščanje prijav in podporo pri načrtovanju ter izvedbi zaposlitvenih postopkov. S tem se zmanjšuje čas, potreben za obdelavo velikega števila prijav, kadrovskim strokovnjakom pa se omogoča, da več pozornosti namenijo zahtevnejšim nalogam, kot sta ocenjevanje kompetenc in komunikacija s kandidati (Tigani, 2025). Poleg avtomatizacije administrativnih postopkov se umetna inteligenca vse pogosteje uporablja tudi kot podpora pri izvedbi intervjujev.

Sodobni sistemi omogočajo samodejno generiranje vprašanj, analizo odgovorov kandidatov, ocenjevanje komunikacijskih sposobnosti in pripravo strukturiranih poročil o uspešnosti posameznika. Nekatere rešitve vključujejo tudi analizo govora, sentimenta in neverbalne komunikacije, kar omogoča celovitejši vpogled v kandidatovo predstavitev med razgovorom (Brintha, Vardhan, Kumar, Babu, & Raj, 2025; Reddy, Priya, Vasavi, Ramesh, & Venu Gopal, 2026). Kljub številnim prednostim uporaba umetne inteligence v kadrovskih procesih odpira tudi pomembna vprašanja glede transparentnosti, zanesljivosti in etičnosti odločanja. Rezultati sistemov umetne inteligence so namreč odvisni od kakovosti vhodnih podatkov, načina učenja modelov in pravilne interpretacije njihovih rezultatov. Zato večina sodobnih rešitev temelji na pristopu, pri katerem umetna inteligenca deluje kot podporno orodje, končno odločitev o izbiri kandidata pa še vedno sprejme človek (Tigani, 2025).

Poleg tega umetna inteligenca ni imuna na napake, značilne za jezikovne modele, kot so halucinacije – torej generiranje na videz prepričljivih, a dejansko napačnih ali izmišljenih informacij, do česar lahko pride tudi pri analizi in ocenjevanju odgovorov kandidatov. Prav tako lahko sistemi umetne inteligence podedujejo pristranskost (angl. *bias*) iz podatkov, na katerih so bili naučeni, kar lahko privede do sistematično krivičnega ocenjevanja določenih skupin kandidatov (Armstrong, Liu, MacNeil, & Metaxa, 2024; Gallegos, et al., 2024). Poleg tega obdelava osebnih podatkov in vsebine razgovorov s strani zunanjih ponudnikov umetne inteligence odpira tudi vprašanja zasebnosti, saj morajo organizacije zagotoviti ustrezno varovanje in namensko uporabo kandidatovih podatkov. Zato je pri uporabi umetne inteligence v kadrovskih procesih ključno zagotoviti ustrezen nadzor, preverjanje rezultatov in kombinacijo avtomatiziranega ocenjevanja s človeško presojo.

Na področju tehničnih intervjujev umetna inteligenca predstavlja priložnost za bolj dosledno vrednotenje kandidatov in pripravo kakovostnih povratnih informacij. Z analizo odgovorov, programerskih rešitev in drugih relevantnih podatkov lahko pomaga prepoznati kandidatove prednosti ter področja, ki jih je treba izboljšati. Takšen pristop ne prispeva le k večji objektivnosti ocenjevanja, temveč kandidatom omogoča tudi boljšo pripravo na prihodnje zaposlitvene postopke (Singh & Kumar, 2025). Podoben pristop k uporabi velikih jezikovnih modelov za obogatitev in semantično analizo podatkov je bil uporabljen tudi v sorodnem diplomskem delu (Skubic, 2025).

## 2.4 Veliki jezikovni modeli (LLM)

Veliki jezikovni modeli (angl. *Large Language Models – LLM*) predstavljajo napredne modele umetne inteligence, zasnovane za razumevanje, analizo in generiranje naravnega jezika. Temeljijo na predhodnem učenju na velikih količinah besedilnih podatkov, zaradi česar lahko prepoznavajo jezikovne vzorce, razumejo kontekst in ustvarjajo smiselne ter koherentne odgovore. Za razliko od tradicionalnih sistemov, ki temeljijo na vnaprej določenih pravilih ali strukturiranih bazah znanja, LLM pridobiva znanje neposredno iz nestrukturiranih besedilnih virov, kar mu omogoča široko uporabnost pri različnih nalogah obdelave naravnega jezika (Aryan, 2025).

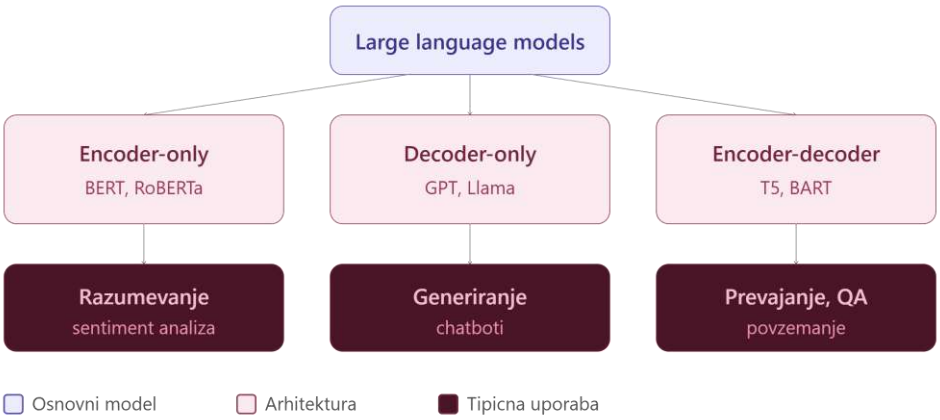
Izraz »veliki« se ne nanaša zgolj na količino učnih podatkov, temveč predvsem na število parametrov modela. Parametri predstavljajo uteži in pristranskosti nevronske mreže, ki se med učenjem prilagajajo z namenom izboljšanja natančnosti napovedi. Večje število parametrov modelu omogoča boljše razumevanje kompleksnih jezikovnih vzorcev in uspešnejše povezovanje informacij iz širšega konteksta, hkrati pa povečuje računske zahteve ter kompleksnost učenja (Aryan, 2025). Teoretična izhodišča za razumevanje jezika s pomočjo računalniških modelov segajo že v devetdeseta leta prejšnjega stoletja, ko je statistični pristop k obdelavi naravnega jezika postavil temelje za kasnejši razvoj jezikovnega modeliranja (Manning & Schütze, 1999).

Nadaljnji razvoj globokih nevronskih mrež je omogočil prehod od statističnih k nevronskim jezikovnim modelom, pri čemer so temeljne koncepte globokega učenja, kot so nevronske mreže, povratne nevronske mreže (RNN) in mehanizmi učenja predstavitev, podrobno opredelili Goodfellow, Bengio in Courville (Goodfellow, Bengio, & Courville, 2016).

Pred uveljavitvijo arhitekture Transformer je bilo razumevanje jezikovnega konteksta in zaporedne obdelave besedil podrobno obravnavano v temeljnem učbeniku s področja obdelave naravnega jezika (Jurafsky & Martin, 2009), ki je opredelil osnovne pristope k jezikovnemu modeliranju, na katerih gradijo tudi sodobni veliki jezikovni modeli. Pomemben preboj pri razvoju velikih jezikovnih modelov je predstavljala arhitektura Transformer, predstavljena leta 2017 v članku *Attention Is All You Need*. Pred njenim razvojem so se pri obdelavi zaporednih podatkov uporabljale predvsem rekurentne nevronske mreže (RNN), ki so podatke obdelovale zaporedno, zaradi česar so imele težave pri obravnavi daljših besedil in dolgoročnih odvisnosti.

Transformer uvaja mehanizem samopozornosti (angl. *self-attention*), ki omogoča hkratno obdelavo vseh tokenov v vhodnem zaporedju in učinkovitejše razumevanje njihove medsebojne povezanosti. Zaradi paralelne obdelave podatkov lahko modeli, ki temeljijo na arhitekturi Transformer, učinkoviteje obdelujejo velike količine besedila in dosegajo boljše rezultate pri nalogah razumevanja in generiranja naravnega jezika (Aryan, 2025).

Razvoj velikih jezikovnih modelov je del širšega razvoja generativne umetne inteligence, ki je od zgodnjih nevronskih modelov v štiridesetih letih prejšnjega stoletja prek več obdobij pospešenega napredka in zastoja (t. i. »AI zim«) pripeljal do sodobnih modelov, kot je družina GPT (Gollnick, 2025).

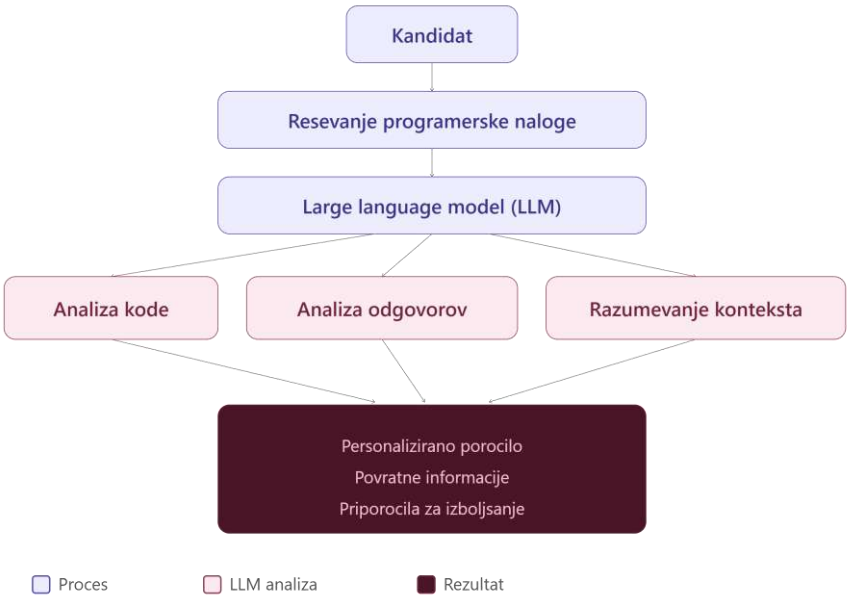



*Slika 2.1 Osnovne arhitekture velikih jezikovnih modelov.*

Vir: lastni vir.

Glede na arhitekturo lahko velike jezikovne modele razdelimo na modele z enkoderjem, modele z dekoderjem in kombinirane modele enkoder–dekoder. Modeli z enkoderjem, kot je BERT, so namenjeni predvsem razumevanju in analizi besedila ter se pogosto uporabljajo pri klasifikaciji besedil, analizi sentimenta in prepoznavanju entitet. Modeli z dekoderjem, kot je družina GPT, ustvarjajo novo besedilo na podlagi podanega poziva, zaradi česar predstavljajo osnovo sodobnih pogovornih sistemov in virtualnih pomočnikov.

Kombinirani modeli enkoder–dekoder združujejo prednosti obeh pristopov in se pogosto uporabljajo pri nalogah, kot sta strojno prevajanje in povzemanje besedil (Aryan, 2025). Po fazi predhodnega učenja se lahko veliki jezikovni modeli dodatno prilagodijo za specifične naloge s postopkom, imenovanim *fine-tuning*. Pri tem se model dodatno uči na podatkih iz izbranega področja uporabe, kar omogoča natančnejše rezultate pri reševanju konkretnih problemov. Takšen pristop omogoča uporabo enakega osnovnega modela v različnih domenah, kot so medicina, pravo, izobraževanje ali kadrovski postopki (Aryan, 2025). Zaradi sposobnosti razumevanja naravnega jezika, analize konteksta in generiranja koherentnih odgovorov so veliki jezikovni modeli postali pomembna komponenta sodobnih inteligentnih sistemov. V kontekstu tehničnih intervjujev omogočajo analizo odgovorov kandidatov, razumevanje programerskih rešitev, generiranje personaliziranih povratnih informacij in podporo objektivnejšemu vrednotenju tehničnega znanja. Prav zaradi teh lastnosti predstavljajo osrednjo tehnologijo tudi v okviru razvitega sistema, predstavljenega v tem diplomskem delu.




*Slika 2.2 Uporaba velikega jezikovnega modela pri analizi tehničnega intervjuja.*

Vir: lastni vir.

## 2.5 Na človeka usmerjeni digitalni sistemi

Na človeka usmerjeni digitalni sistemi (angl. *Human-Centered Digital Systems*) temeljijo na pristopu, pri katerem so potrebe, sposobnosti in pričakovanja uporabnikov postavljeni v središče razvoja informacijskih rešitev. Pri načrtovanju takšnih sistemov ni dovolj zagotoviti zgolj tehnične pravilnosti ali naprednih funkcionalnosti, temveč je treba poskrbeti tudi za dobro uporabniško izkušnjo, dostopnost in enostavno uporabo. Uporabniški vmesnik mora biti razumljiv različnim skupinam uporabnikov, informacije pa predstavljene na jasen in pregleden način. Takšen pristop spodbuja razvoj vključujočih digitalnih rešitev, ki omogočajo učinkovito uporabo čim širšemu krogu ljudi, ne glede na njihove sposobnosti ali okoliščine (Gilbert, 2019). Pri razvoju sodobnih sistemov umetne inteligence postaja na človeka usmerjeno načrtovanje še pomembnejše, saj sistemi pogosto sodelujejo pri sprejemanju odločitev ali podajajo priporočila uporabnikom.

Gilbert poudarja, da morajo biti digitalni sistemi zasnovani z razumevanjem potreb končnih uporabnikov, pri čemer je pomembno uporabiti pristope, kot so uporabniško usmerjeno načrtovanje (angl. *User-Centered Design*), oblikovalsko razmišljanje (angl. *Design Thinking*) in vključevanje uporabnikov v proces razvoja že od začetnih faz (Gilbert, 2019). Poleg tega morajo biti upoštevana temeljna načela oblikovanja uporabniških vmesnikov, ki izhajajo iz razumevanja človeške percepcije in kognicije (Johnson, 2010).

# 3 Analiza in načrtovanje sistema

## 3.1 Analiza zahtev

Sistem mora zagotavljati varno avtentikacijo, ločevanje vlog kandidata in intervjuvarja ter nadzor dostopa do intervjujev in poročil. Intervjuvar lahko ustvari ali načrtuje intervju z video komunikacijo, kandidat pa v skupnem delovnem prostoru izbere nalogo in programski jezik, pripravi načrt, napiše in izvede kodo ter razloži rešitev. Kandidatovo delo se samodejno shranjuje. Po zaključku intervjuja veliki jezikovni model analizira kontekst naloge, kandidatovo predajo in rezultat izvajanja ter pripravi strukturirano poročilo z ocenami, povratnimi informacijami in priporočili. Pooblaščeni uporabniki lahko dostopajo do pripadajočih intervjujev in poročilo izvozijo v obliki `PDF`.

Nefunkcionalne zahteve vključujejo zanesljivost, varnost, odzivnost in razširljivost sistema. Napake zunanjih storitev ne smejo povzročiti izgube shranjenih podatkov, dostop do intervjujev, kandidatovih rešitev in poročil pa mora biti omejen na pooblaščene uporabnike. Koda se izvaja v ločenem okolju, arhitektura pa omogoča dodajanje novih jezikov, nalog in meril ocenjevanja. Sistem je funkcionalen prototip za izvedbo in ocenjevanje tehničnih intervjujev za programsko-inženirska delovna mesta. Ne vključuje upravljanja razpisov, celotnega kadrovskega postopka ali samodejnega razvrščanja in izbire kandidatov.

## 3.2 Arhitektura sistema

Arhitektura sistema je razdeljena na odjemalski del, zaledni del in zunanje storitve, kot je prikazano na sliki 3.1. Ločitev odgovornosti omogoča lažje vzdrževanje, testiranje in razširjanje posameznih funkcionalnosti. Odjemalski del je razvit z ogrodjem `Next.js`[1], knjižnico React in programskim jezikom `TypeScript`[2], uporabniški vmesnik pa s `Tailwind CSS`[3] in komponentami `shadcn/ui`[4]. Vključuje upravljanje intervjujev, programersko okolje, video komunikacijo in prikaz poročil. Video funkcionalnosti zagotavlja SDK storitve `Stream`[5].

Zaledni del temelji predvsem na platformi `Convex`[6], ki združuje podatkovno zbirko in aplikacijsko logiko. Poizvedbe pridobivajo podatke, mutacije jih spreminjajo, notranje akcije pa izvajajo asinhrone postopke, kot je komunikacija z modelom `Gemini`. Reaktivne poizvedbe omogočajo samodejno posodabljanje uporabniškega vmesnika ob spremembah podatkov. Za posamezne postopke sta uporabljena tudi `Next.js API`-pot za izvajanje kode in strežniško dejanje za varno ustvarjanje uporabniškega žetona `Stream`. Zunanje storitve vključujejo `Clerk`[7] za avtentikacijo, `Stream` za video komunikacijo, `Piston`[8] za izvajanje kode in `Gemini API`[9] za analizo kandidatove predaje. `Clerk` je povezan prek ponudnika in webhooka, `Stream` prek odjemalskega SDK-ja in strežniško ustvarjenega žetona, `Gemini` pa prek notranje akcije `Convex`. Izvajanje kode poteka prek `Next.js API`-poti, ki preveri vhodne podatke in zahtevo posreduje storitvi `Piston`.

[1] https://nextjs.org/

[2] https://www.typescriptlang.org/

[3] https://tailwindcss.com/

[4] https://ui.shadcn.com/

[5] https://getstream.io/

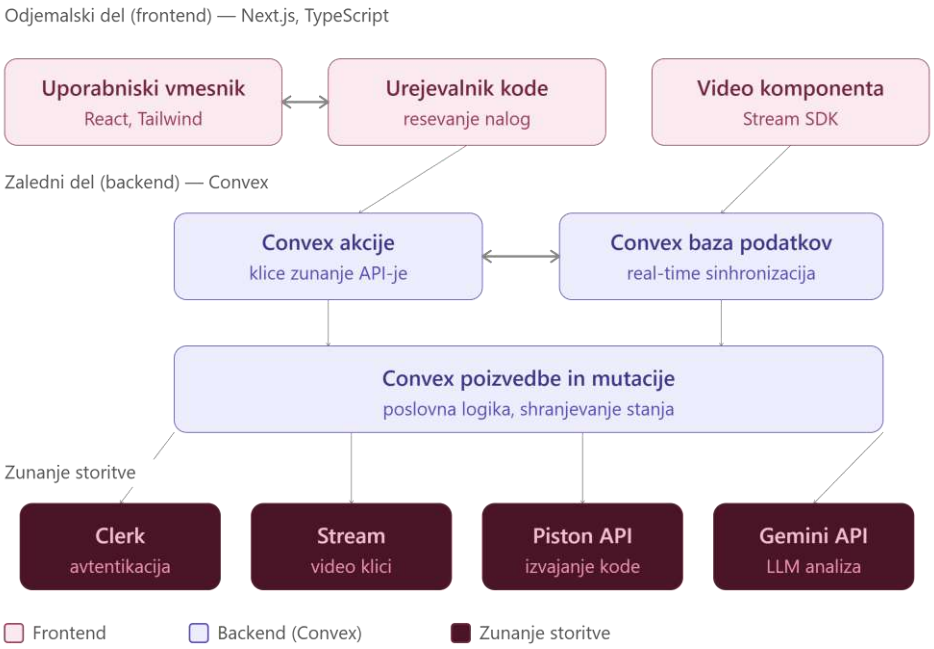
[6] https://www.convex.dev/

[7] https://clerk.com/

[8] https://piston.readthedocs.io/en/latest/api-v2/

[9] https://aistudio.google.com/welcome

Ta je v razvojnem in testnem okolju nameščena v vsebniku `Docker`[10], ki zagotavlja ločeno izvajalno okolje in potrebne različice podprtih programskih jezikov. Rezultat izvajanja se prek `API`-poti vrne uporabniškemu vmesniku. Po prijavi prek storitve `Clerk` uporabnik vstopi v intervju, kjer video komunikacija poteka prek storitve `Stream`. Kandidatov načrt, koda, razlaga in rezultat izvajanja se z mutacijami shranjujejo v `Convex`. Ob izbiri izvajanja `API`-pot posreduje kodo storitvi `Piston` in vrne njen rezultat. Po zaključku intervjuja notranja akcija `Convex` modelu `Gemini` pošlje kontekst izbrane naloge, kandidatovo predajo in rezultat izvajanja. Strukturirana analiza se shrani v podatkovno zbirko in se zaradi reaktivnih poizvedb samodejno prikaže v uporabniškem vmesniku.




*Slika 3.1 Arhitektura razvitega sistema – odjemalski del, zaledni del in zunanje storitve.*

Vir: lastni vir.

[10] https://www.docker.com/

Predstavljena arhitektura jasno ločuje predstavitveno, podatkovno in integracijsko plast sistema. `Convex` poenostavlja shranjevanje in reaktivno pridobivanje podatkov, zunanje storitve pa so vključene prek mehanizmov, prilagojenih njihovemu namenu, kot so spletni webhook, odjemalski `SDK`, strežniško dejanje, `API`-pot in notranja akcija. Takšna zasnova omogoča nadaljnje dodajanje storitev in modelov umetne inteligence.

## 3.3 Uporabniške vloge in funkcionalnosti

Sistem loči vlogi kandidata in intervjuvarja, ki sta shranjeni v uporabniškem zapisu in omogočata prilagoditev funkcionalnosti ter preverjanje dostopa v zalednem delu. Podrobnejša implementacija avtentikacije in avtorizacije je predstavljena v poglavju 4.2. Intervjuvar lahko ustvari ali načrtuje intervju, mu dodeli kandidata, vodi video sejo in po zaključku dostopa do strukturiranega poročila. Kandidat dostopa do svojih intervjujev, sodeluje v video komunikaciji in v programerskem okolju izbere nalogo ter jezik, pripravi načrt, napiše in izvede kodo ter razloži rešitev. Po zaključku lahko dostopa do lastnega poročila z ocenami in priporočili.

## 3.4 Načrt podatkovnega modela

Podatkovni model sistema je implementiran v obliki sheme `Convex` in je sestavljen iz petih osrednjih entitet: **`users`**, **`interviews`**, **`comments`**, **`submissions`** in **`aiReports`**. Posamezne entitete so med seboj povezane z identifikatorji uporabnikov, intervjujev in oddanih rešitev, kar omogoča sledenje celotnemu procesu, od registracije uporabnika in izvedbe intervjuja do avtomatizirane analize ter prikaza končnega poročila.

Entiteta **users** hrani osnovne podatke o registriranem uporabniku: ime, e-poštni naslov, neobvezno profilno sliko, uporabniško vlogo in identifikator **clerkId**. Vloga je omejena na vrednosti **candidate** in **interviewer**, identifikator **clerkId** pa povezuje zapis v podatkovni bazi z uporabnikom, ki ga upravlja storitev `Clerk`. Indeks **by_clerk_id** omogoča učinkovito pridobivanje uporabniškega zapisa na podlagi njegove avtenticirane identitete.

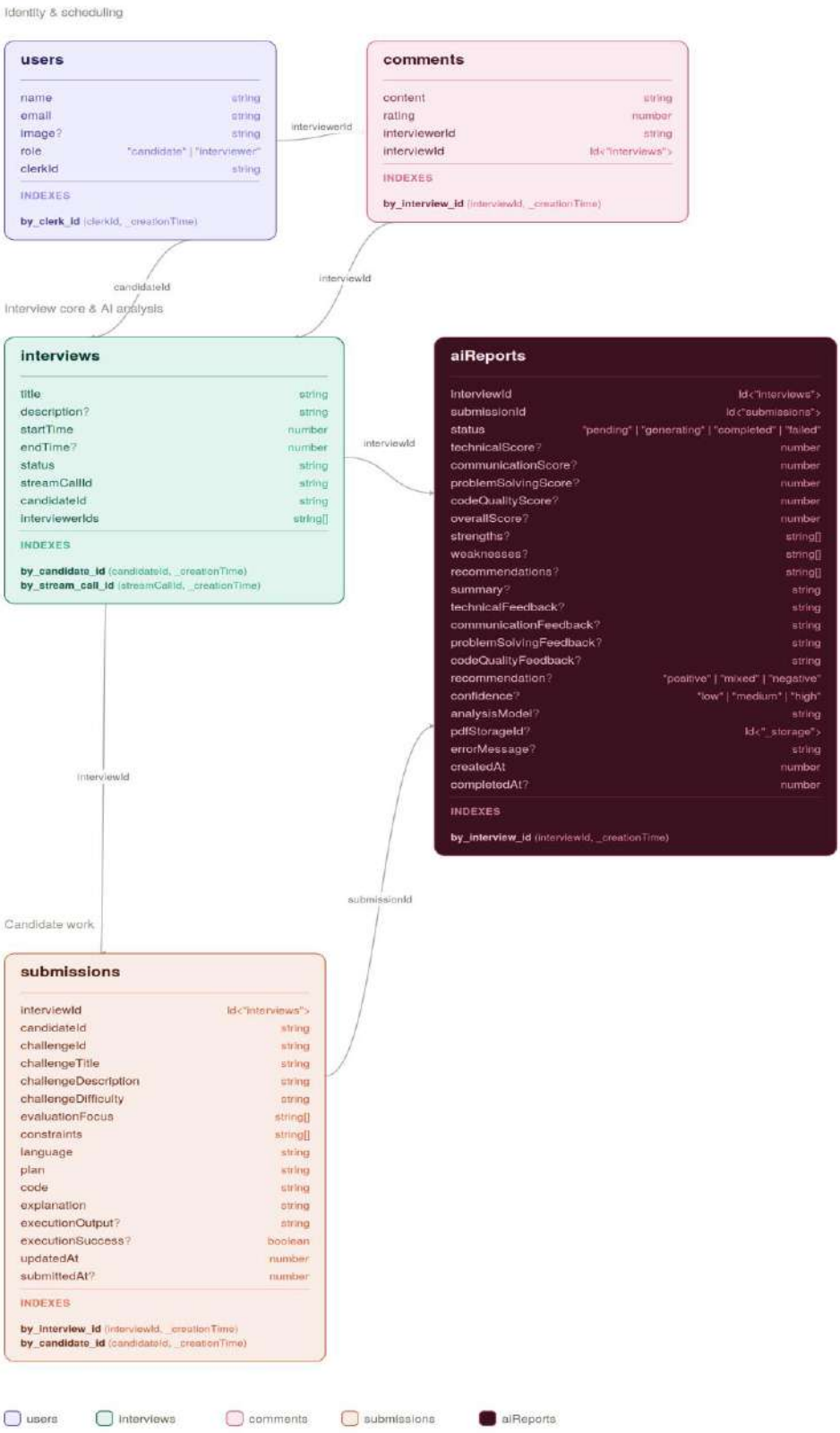
Entiteta **interviews** predstavlja posamezen tehnični intervju. Vsebuje naslov in opis intervjuja, čas začetka in neobvezni čas zaključka, trenutno stanje intervjuja ter identifikator pripadajočega videoklica v storitvi `Stream`. Zapis vključuje tudi identifikator kandidata (**candidateId**) in seznam pripadajočih intervjuvarjev (**interviewerIds**). Indeksa po identifikatorju kandidata in videoklica omogočata hitro pridobivanje kandidatovih intervjujev in povezovanje aktivnega klica z ustreznim zapisom intervjuja.

Entiteta **comments** hrani ročno povratno informacijo intervjuvarja. Posamezen komentar vsebuje besedilno obrazložitev, številčno oceno, identifikator intervjuvarja in referenco na pripadajoči intervju. Na ta način je ročno ocenjevanje ločeno od avtomatizirane analize, hkrati pa je ohranjena sledljivost avtorja povratne informacije. Indeks **by_interview_id** omogoča pridobivanje vseh komentarjev, povezanih z izbranim intervjujem.

Entiteta **submissions** predstavlja kandidatovo delo med tehničnim intervjujem in je osrednja povezava med reševanjem naloge ter avtomatizirano analizo. Vsebuje referenco na intervju, identifikator kandidata in podatke o izbrani tehnični nalogi, kot so njen identifikator, naslov, opis, težavnost, kriteriji ocenjevanja in omejitve. Poleg tega hrani izbrani programski jezik, kandidatov načrt rešitve, izvorno kodo in končno razlago pristopa. Če kandidat svojo rešitev izvede, se lahko v isti zapis shranita tudi rezultat izvajanja (**executionOutput**) in informacija o uspešnosti izvajanja (**executionSuccess**).

Polje **updatedAt** beleži čas zadnje spremembe, neobvezno polje `submittedAt` pa označuje, da je bila rešitev po zaključku intervjuja dokončno oddana in je ni več mogoče spreminjati. Indeksa po intervjuju in kandidatu omogočata učinkovito pridobivanje oddane rešitve glede na kontekst trenutnega uporabnika.

Entiteta **aiReports** hrani rezultat avtomatizirane analize kandidatove oddane rešitve. Vsako poročilo je neposredno povezano z intervjujem in pripadajočim zapisom **submissions**. Polje **status** opisuje trenutno stanje generiranja poročila in lahko vsebuje vrednosti **pending**, **generating**, **completed** ali **failed**. Med generiranjem so posamezna ocenjevalna polja neobvezna, saj še niso na voljo. Po uspešno zaključeni analizi se v poročilo shranijo ocene tehničnega znanja, komunikacije, reševanja problemov in kakovosti kode ter skupna ocena kandidata. Poleg številčnih rezultatov poročilo vsebuje povzetek uspešnosti, prepoznane prednosti in slabosti, priporočila za izboljšanje ter podrobnejšo povratno informacijo za vsako ocenjeno področje. Končno priporočilo je predstavljeno z vrednostjo **positive**, **mixed** ali **negative**, stopnja zaupanja modela pa z vrednostjo **low**, **medium** ali **high**. Sistem shrani tudi oznako uporabljenega jezikovnega modela in čas ustvarjanja oziroma zaključka analize. Če generiranje ni uspešno, se stanje spremeni v **failed**, polje **errorMessage** pa vsebuje opis napake, ki omogoča lažje diagnosticiranje in ponovni poskus analize. Neobvezno polje **pdfStorageId** omogoča razširitev s shranjevanjem dokumenta `PDF` v storitvi `Convex Storage`. V trenutni različici sistema se dokument `PDF` generira na zahtevo neposredno iz podatkov dokončanega poročila in se prenese v uporabnikovem brskalniku, zato njegovo trajno shranjevanje ni zahtevano.

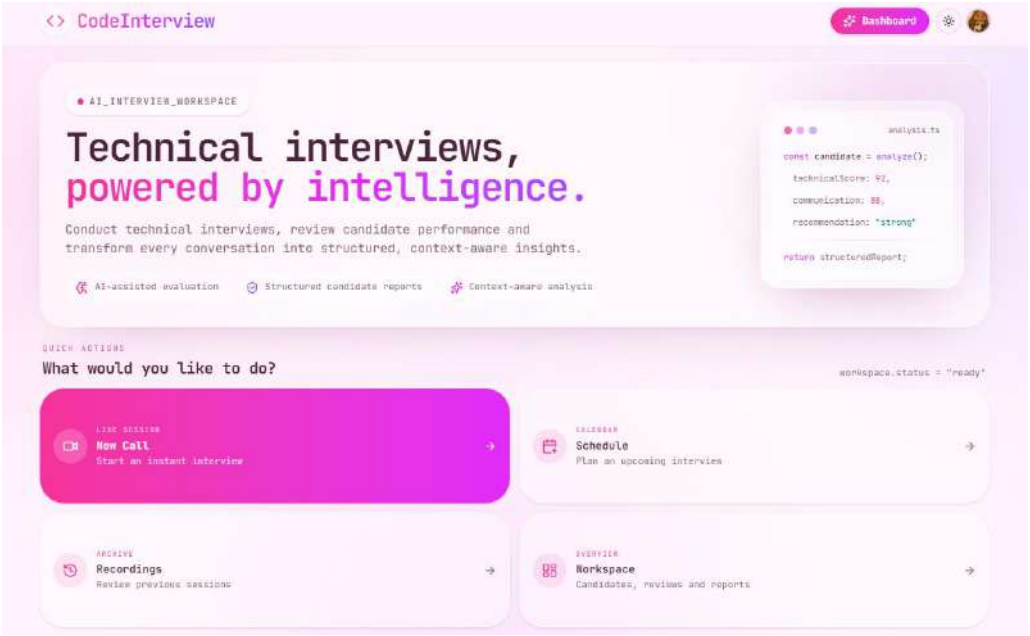



*Slika 3.2 Shema Convex podatkovne baze.*

Vir: lastni vir.

## 3.5 Načrt uporabniškega vmesnika

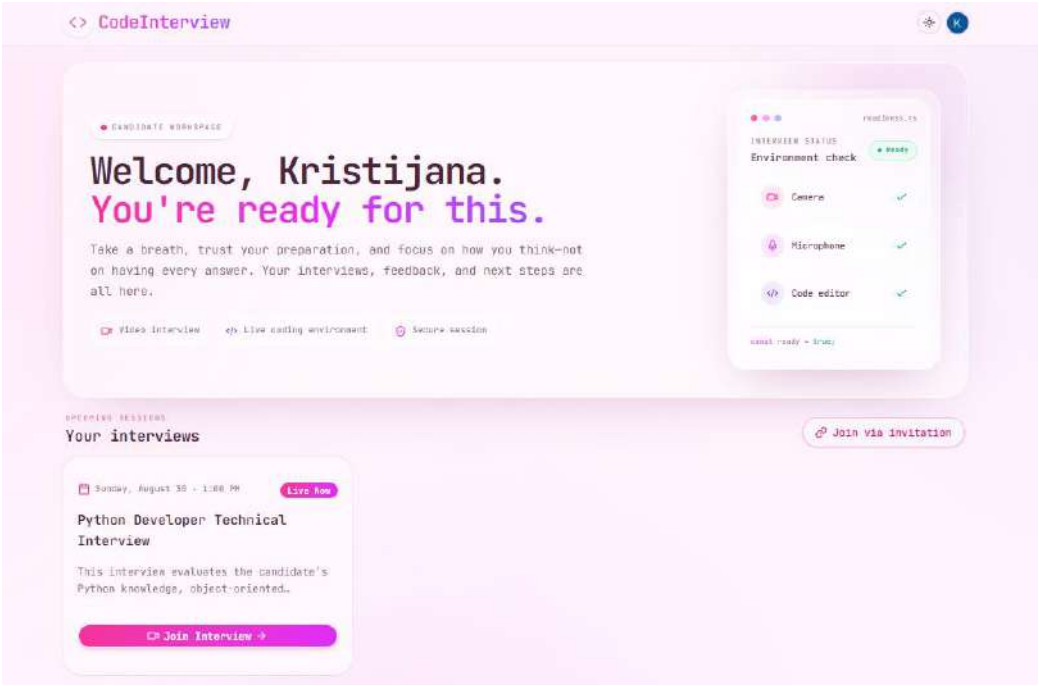
Uporabniški vmesnik sistema je zasnovan skladno z načeli na človeka usmerjenega oblikovanja. Glavni cilj je bil zagotoviti pregleden, razumljiv in vizualno dosleden dostop do ključnih funkcionalnosti tehničnega intervjuja. Javno dostopna pristajalna stran uporabniku predstavi namen sistema in omogoča nadaljevanje na registracijo ali prijavo. Vsak novo registrirani uporabnik privzeto pridobi vlogo kandidata, medtem ko lahko vlogo intervjuvarja zaradi nadzora dostopa dodeli le skrbnik sistema. Vsebina uporabniškega vmesnika se prilagaja uporabniški vlogi. Intervjuvar ima dostop do ustvarjanja in načrtovanja intervjujev, izbire kandidata, pregleda izvedenih intervjujev ter ročnih in avtomatizirano ustvarjenih poročil. Kandidat vidi samo lastne intervjuje, se jim lahko pridruži prek nadzorne plošče ali povezave s povabilom in po zaključeni analizi dostopa do lastnega poročila in povratnih informacij.




*Slika 3.3 Začetna stran intervjuvarja.*

Vir: lastni vir.

Vizualna podoba temelji na enotni rožnato-vijolični barvni paleti, zaobljenih elementih, prosojnih površinah in tipografiji monospace, ki poudarja tehnično naravo aplikacije. Enaka oblikovna pravila se uporabljajo na vseh straneh sistema, s čimer sta zagotovljeni vizualna doslednost in lažja orientacija uporabnika.




*Slika 3.4 Začetna stran in nadzorna plošča kandidata.*

Vir: lastni vir.

Vmesnik je odziven in se prilagaja namiznim, tabličnim in mobilnim napravam. Razporeditev elementov, velikosti besedila in širine komponent se spreminjajo glede na razpoložljivi prostor, pri čemer se ohranjata berljivost in dostopnost glavnih funkcij. Sistem podpira tudi svetli in temni način prikaza.

# 4 Implementacija sistema

## 4.1 Uporabljene tehnologije

Sistem je implementiran z uporabo sodobnega spletnega tehnološkega sklada, sestavljenega iz odjemalskega dela, zalednega dela in zunanjih storitev, opisanih v poglavju 3.2. Odjemalski del je razvit z ogrodjem `Next.js`, ki temelji na knjižnici `React` in razširja njene zmožnosti z usmerjanjem strani (angl. *routing*), izrisovanjem na strežniški strani (angl. *server-side rendering*) ter optimizacijo nalaganja. Uporabljen je programski jezik `TypeScript`, nadgradnja jezika `JavaScript`, ki uvaja statično tipizacijo in s tem že med razvojem odkriva morebitne napake, ki bi se sicer pojavile šele med izvajanjem programa. Oblikovanje uporabniškega vmesnika temelji na knjižnici `Tailwind CSS`, ki omogoča oblikovanje neposredno v označevalni kodi `HTML` prek vnaprej pripravljenih razredov, in na komponentah `shadcn/ui`, ki ponujajo dostopne in vizualno dosledne gradnike uporabniškega vmesnika. Za prikaz in urejanje programske kode med reševanjem nalog je uporabljena knjižnica `Monaco Editor`[11], ki je enak urejevalnik kode, kot ga uporablja `Visual Studio Code`, in ponuja barvno označevanje sintakse, samodejno dokončevanje ter podporo za več programskih jezikov.

Zaledni del temelji na platformi `Convex`, ki spada med storitve tipa »zaledje kot storitev« (angl. *Backend-as-a-Service*). `Convex` združuje podatkovno bazo, izvajanje poslovne logike in mehanizem za samodejno sinhronizacijo podatkov v realnem času med zalednim delom ter uporabniškim vmesnikom, brez potrebe po ročni implementaciji spletnih vtičnic (angl. *WebSockets*) ali periodičnem osveževanju podatkov. Funkcije `Convex` delimo na poizvedbe in mutacije, namenjene branju in spreminjanju podatkov, ter na akcije, prek katerih zaledni del komunicira z zunanjimi storitvami.

[11] https://www.npmjs.com/package/monaco-editor

Avtentikacija uporabnikov je implementirana s storitvijo `Clerk`, ki spada med storitve tipa »avtentikacija kot storitev« (angl. *Authentication-as-a-Service*). Tovrstne storitve odjemalcem odvzamejo potrebo po samostojni implementaciji občutljivih varnostnih mehanizmov, kot so varno zgoščevanje (angl. *hashing*) gesel, upravljanje spletnih žetonov `JWT`, obravnava poteka in obnavljanja uporabniških sej ter implementacija protokola `OAuth` za prijavo prek zunanjih ponudnikov identitete. `Clerk` ponuja vnaprej pripravljene komponente uporabniškega vmesnika za registracijo, prijavo in upravljanje računa, hkrati pa v ozadju izda vsakemu prijavljenemu uporabniku podpisan žeton, ki ga lahko zaledni del sistema neodvisno preveri, ne da bi moral pri tem komunicirati neposredno s strežnikom `Clerk` ob vsaki zahtevi. Uporaba take storitve namesto lastne implementacije bistveno zmanjša tveganje varnostnih ranljivosti, saj gre za funkcionalnost, pri kateri se napake najpogosteje in najbolj resno odražajo na varnosti celotnega sistema.

Video komunikacija med kandidatom in intervjuvarjem poteka prek storitve `Stream`, ki temelji na tehnologiji `WebRTC` (angl. *Web Real-Time Communication*) – odprtem standardu, ki spletnim brskalnikom omogoča neposreden prenos avdio in video podatkov v realnem času. Vzpostavitev zanesljive video komunikacije med dvema udeležencema v praksi zahteva reševanje več tehničnih izzivov, med drugim prehajanje omrežij za prevajanje naslovov (angl. *NAT traversal*), prilagajanje kakovosti prenosa glede na razpoložljivo pasovno širino in zagotavljanje nizke zakasnitve. Storitev `Stream` te izzive rešuje z uporabo omrežja posredniških strežnikov (angl. *media servers*), razporejenih geografsko, kar zagotavlja stabilno povezavo, ne glede na lokacijo udeležencev, in ponuja programski vmesnik (SDK), ki to kompleksnost skrije za preprostimi komponentami, vključljivimi neposredno v odjemalski del aplikacije.

Izvajanje kandidatove izvorne kode poteka prek storitve `Piston`, odprtokodnega mehanizma za izvajanje kode v izoliranem okolju (angl. *sandbox*), ki podpira izvajanje kode v več deset programskih jezikih, ne da bi bilo treba to infrastrukturo vzpostaviti in vzdrževati samostojno. Analiza odgovorov, ocenjevanje izvorne kode in generiranje strukturirane povratne informacije potekajo z uporabo velikega jezikovnega modela, dostopnega prek vmesnika `Gemini API`, ki podjetju ali posamezniku omogoča dostop do zmogljivosti jezikovnega modela brez potrebe po lastni računski infrastrukturi za njegovo gostovanje.

Pregled uporabljenih tehnologij po posameznih komponentah sistema je predstavljen v tabeli 4.1.

| ***Komponenta*** | ***Uporabljena tehnologija*** |
|---|---|
| *Odjemalski del* | React, Next.js, TypeScript, Tailwind CSS, shadcn/ui |
| *Urejevalnik kode* | Monaco Editor |
| *Zaledni del* | Convex |
| *Avtentikacija* | Clerk |
| *Video komunikacija* | Stream |
| *Izvajanje kode* | Piston |
| *Analiza z LLM* | Gemini API |

*Tabela 4.1 Pregled uporabljenih tehnologij.*

Vir: lastni vir.

## 4.2 Avtentikacija in avtorizacija uporabnikov

Avtentikacija uporabnikov je implementirana s storitvijo `Clerk`, ki upravlja registracijo, prijavo in uporabniške seje. Uporabnik se lahko prijavi z e-poštnim naslovom in geslom ali prek zunanjih ponudnikov identitete `Google` in `GitHub` z uporabo protokola `OAuth`. S tem sistem uporablja preverjen mehanizem za upravljanje identitet, ne da bi bilo treba občutljivo avtentikacijsko logiko in uporabniška gesla hraniti v lastni podatkovni zbirki. Poleg avtentikacije sistem izvaja tudi avtorizacijo. Avtentikacija določa, kdo je prijavljeni uporabnik, avtorizacija pa preverja, do katerih podatkov in funkcionalnosti lahko dostopa glede na svojo vlogo in povezavo z izbranim intervjujem.

### 4.2.1 Povezava storitev Clerk in Convex

`Clerk` je v odjemalski del aplikacije vključen prek ponudnika **`ClerkProvider`**, ki omogoča dostop do podatkov o prijavljenem uporabniku v celotni komponentni strukturi. Znotraj njega je uporabljen **`ConvexProviderWithClerk`**, ki poveže `Clerkovo` uporabniško sejo s podatkovno platformo `Convex`. Ko odjemalski del dostopa do zaščitene poizvedbe ali mutacije, `Clerk` zagotovi uporabniški žeton, `Convex` pa preveri njegovega izdajatelja in veljavnost. Konfiguracija `Convex` določa `Clerkovo` domeno kot zaupanja vrednega ponudnika identitete in povezuje izdani žeton z aplikacijo.

Po uspešnem preverjanju lahko zaledne funkcije identiteto uporabnika pridobijo iz konteksta zahteve. Obdelava zahteve prek `Clerkovega` middleware sama po sebi ne pomeni, da so vse poti samodejno zaščitene. Preverjanje dostopa se zato izvaja tudi znotraj zaščitenih strani, poizvedb, mutacij in strežniških akcij.

### 4.2.2 Sinhronizacija uporabnikov s podatkovno bazo

`Clerk` upravlja avtentikacijske podatke, aplikacija pa za svoje delovanje potrebuje tudi lasten zapis uporabnika v podatkovni zbirki `Convex`. Ta zapis vsebuje ime, e-poštni naslov, neobvezno sliko profila, identifikator **`clerkId`** in uporabniško vlogo. Sinhronizacija se izvede prek webhooka. Ko se v storitvi `Clerk` registrira nov uporabnik, `Clerk` pošlje dogodek **`user.created`** na `HTTP`-končno točko zalednega dela. Zahteva vsebuje glavo z identifikatorjem, časovnim žigom in digitalnim podpisom. Sistem pred obdelavo dogodka s knjižnico `Svix`[12] preveri pristnost podpisa. Če zahtevane glave manjkajo ali podpis ni veljaven, zahtevo zavrne.

Po uspešnem preverjanju iz dogodka pridobi identifikator, e-poštni naslov, ime in sliko uporabnika ter sproži shranjevanje v podatkovno zbirko. Pred ustvarjanjem zapisa sistem preveri, ali uporabnik z enakim identifikatorjem **`clerkId`** že obstaja. S tem prepreči podvajanje podatkov. Vsak novi uporabnik privzeto pridobi vlogo kandidata, medtem ko se vloga intervjuvarja dodeli nadzorovano.

[12] https://www.svix.com/

```
ALGORITHM SinhronizirajNovegaUporabnika(zahteva)

skrivnost ← pridobi skrivnost webhooka
glave ← pridobi identifikator, časovni žig in podpis

IF skrivnost ali zahtevane glave ne obstajajo THEN
    zavrni zahtevo
END IF

dogodek ← preveri telo in podpis zahteve s storitvijo Svix

IF podpis ni veljaven THEN
    zavrni zahtevo
END IF

IF vrsta dogodka ni "user.created" THEN
    RETURN uspešen odgovor
END IF

uporabniškiPodatki ← pridobi podatke iz dogodka
obstoječiUporabnik ← poišči uporabnika po clerkId

IF obstoječiUporabnik ne obstaja THEN
    ustvari uporabnika
        z vlogo "candidate"
END IF

RETURN uspešen odgovor

END ALGORITHM
```

*Psevdokoda 4.1 Preverjanje webhooka in sinhronizacija novega uporabnika.*

Vir: lastni vir.

### 4.2.3 Uporabniške vloge in preverjanje dostopa

Storitvi `Clerk` in `Convex` imata pri preverjanju uporabnika različne naloge. `Clerk` potrdi identiteto prijavljenega uporabnika, medtem ko `Convex` hrani domensko vlogo uporabnika, ki je lahko **`candidate`** ali **`interviewer`**. Povezava med zapisoma je vzpostavljena prek identifikatorja **`clerkId`**.

V odjemalskem delu se vloga pridobi z namensko `Reactovo` kljuko **`useUserRole`**. Ta na podlagi `Clerkovega` identifikatorja poišče pripadajoč zapis v podatkovni zbirki in vrne podatek, ali je uporabnik kandidat oziroma intervjuvar. Ker `Convex` omogoča reaktivne poizvedbe, se sprememba vloge v podatkovni zbirki odrazi tudi v uporabniškem vmesniku. Stanje nalaganja je obravnavano ločeno od dejanske uporabniške vloge. S tem se prepreči, da bi sistem med pridobivanjem podatkov uporabniku začasno prikazal elemente, ki ne pripadajo njegovi vlogi.

Preverjanje vloge v odjemalskem delu je namenjeno prilagoditvi uporabniškega vmesnika in samo po sebi ne predstavlja zadostne varnostne zaščite. Zaledne funkcije zato dodatno preverjajo avtenticirano identiteto in povezavo uporabnika z zahtevanim virom. Pri dostopu do intervjuja ali kandidatove predaje sistem na primer preveri, ali je uporabnik pripadajoči kandidat ali eden od dodeljenih intervjuvarjev. Tako sprememba odjemalskega vmesnika ne omogoči dostopa do podatkov drugega uporabnika.

## 4.3 Video komunikacija

Video komunikacija med kandidatom in intervjuvarjem je implementirana s storitvijo `Stream`, ki zagotavlja infrastrukturo za prenos zvoka in slike v realnem času. Uporaba zunanje storitve omogoča vključitev videoklica brez razvoja lastne medijske infrastrukture in reševanja zahtevnejših omrežnih problemov, kot so vzpostavljanje povezave med različnimi omrežji, prilagajanje kakovosti prenosa in upravljanje več udeležencev. V sistemu sta poslovni zapis intervjuja in video seja shranjena ločeno. Storitev `Stream` upravlja videoklic in njegove udeležence, platforma `Convex` pa hrani podatke o intervjuju, kandidatu, intervjuvarjih, času izvedbe in trenutnem stanju. Obe komponenti sta povezani z identifikatorjem **`streamCallId`**, ki je shranjen v entiteti **`interviews`**. Takšna ločitev omogoča, da je isti videoklic vedno mogoče povezati z ustreznim intervjujem, kandidatovo oddano rešitvijo in pozneje ustvarjenim poročilom.

### 4.3.1 Varna inicializacija video odjemalca

Dostop do video komunikacije je omogočen samo avtenticiranim uporabnikom. Komponenta, odgovorna za inicializacijo video odjemalca, najprej počaka, da storitev `Clerk` zaključi preverjanje uporabniške seje. Če uporabnik ni prijavljen, se video odjemalec ne ustvari, javni deli aplikacije pa ostanejo dostopni brez povezovanja s storitvijo `Stream`. Za prijavljenega uporabnika se odjemalec inicializira z javnim ključem storitve `Stream` in osnovnimi podatki uporabnika, kot so identifikator, ime ter profilna slika. Identifikator uporabnika je enak identifikatorju, ki ga zagotavlja `Clerk`, kar omogoča dosledno povezovanje uporabnika med avtentikacijskim, podatkovnim in video sistemom. Inicializacija video odjemalca se izvede šele po zaključku začetnega izrisa uporabniškega vmesnika. Med vzpostavljanjem povezave sistem prikazuje nalagalno stanje, morebitna napaka pri inicializaciji pa se obravnava ločeno.

Ob odstranitvi komponente oziroma odjavi uporabnika se povezava z odjemalcem `Stream` prekine, s čimer se preprečita ohranjanje nepotrebne seje in podvajanje povezav. Odjemalski del za vzpostavitev povezave potrebuje uporabniški žeton. Ker generiranje žetona zahteva tajni ključ storitve `Stream`, tega postopka ni dovoljeno izvajati v brskalniku. Žeton se zato ustvari v strežniškem dejanju, kjer se najprej preveri trenutno prijavljeni uporabnik. Če uporabnik ni avtenticiran, strežnik zahtevo zavrne. V nasprotnem primeru ustvari uporabniški žeton, vezan na identifikator uporabnika, in ga vrne video odjemalcu. Knjižnica `Stream` lahko funkcijo za pridobivanje žetona ponovno pokliče tudi ob njegovem poteku, zato uporabniku med daljšim intervjujem ni treba ročno obnavljati povezave.

```
ALGORITHM GenerirajVideoŽeton

uporabnik ← pridobi trenutno avtenticiranega uporabnika

IF uporabnik ne obstaja THEN
    zavrni zahtevo
END IF

inicializiraj strežniškega odjemalca Stream
    z javnim in tajnim ključem

žeton ← generiraj žeton,
    vezan na identifikator uporabnika

RETURN žeton

END ALGORITHM
```

*Psevdokoda 4.2 Varno generiranje uporabniškega žetona Stream.*

Vir: lastni vir.

### 4.3.2 Ustvarjanje takojšnjega intervjuja

Sistem poleg načrtovanih intervjujev omogoča tudi ustvarjanje takojšnjega intervjuja. Pred začetkom mora intervjuvar izbrati kandidata iz seznama registriranih uporabnikov. S tem je zagotovljeno, da video seja ni ustvarjena brez povezave s konkretnim kandidatom in zapisom intervjuja. Po potrditvi izbire sistem najprej ustvari enolični identifikator klica in z njim registrira novo video sejo v storitvi `Stream`. Nato se v podatkovno bazo `Convex` vstavi zapis intervjuja, ki vsebuje naslov, opis, čas začetka, začetno stanje, identifikator videoklica, identifikator izbranega kandidata in identifikator intervjuvarja. Šele po uspešnem ustvarjanju obeh zapisov je intervjuvar preusmerjen v intervjujski delovni prostor. Če ustvarjanje videoklica ali podatkovnega zapisa ni uspešno, sistem uporabniku prikaže obvestilo o napaki in ga ne preusmeri v intervjujski delovni prostor.

```
ALGORITHM UstvariTakojšnjiIntervju

preveri prijavljenega uporabnika
pridobi izbranega kandidata

IF kandidat ni izbran THEN
    prekini postopek
END IF

identifikatorKlica ← ustvari enolični identifikator

ustvari video klic Stream
    z identifikatorjemKlica

shrani intervju v podatkovno bazo:
    kandidat ← izbrani kandidat
    intervjuvar ← prijavljeni uporabnik
    video klic ← identifikatorKlica
    začetek ← trenutni čas
    stanje ← "prihajajoč"

preusmeri intervjuvarja
    v intervjujski prostor

END ALGORITHM
```

*Psevdokoda 4.3 Ustvarjanje takojšnjega intervjuja.*

Vir: lastni vir.

### 4.3.3 Pridružitev video seji in nadzor dostopa

Uporabnik se lahko intervjuju pridruži prek lastne nadzorne plošče ali povezave s povabilom. V obeh primerih je identifikator klica vsebovan v naslovu strani intervjuja. Na njegovi podlagi sistem pridobi ustrezen klic iz storitve `Stream` in pripadajoči zapis intervjuja iz podatkovne baze `Convex`. Pridobivanje zapisa intervjuja ni odvisno samo od poznavanja identifikatorja klica. Zaledna poizvedba najprej preveri identiteto uporabnika, nato pa dovoli dostop samo, če je uporabnik kandidat, dodeljen temu intervjuju, ali eden od pripadajočih intervjuvarjev.

Če uporabnik ni povezan z intervjujem, sistem ne vrne njegovega zapisa. Takšna preverba je izvedena na zaledni strani in zato ne temelji samo na skrivanju gumbov ali strani v uporabniškem vmesniku. Poznavanje ali spreminjanje identifikatorja klica v naslovu `URL` ne zadostuje za pridobitev podatkov drugega intervjuja. Pred vstopom v video sejo lahko uporabnik preveri kamero in mikrofon. Ko je priprava zaključena, se uporabnik pridruži klicu, uporabniški vmesnik pa se prikaže šele, ko stanje klica potrdi uspešno pridružitev. Do takrat se prikazuje nalagalno stanje.

```
ALGORITHM PridobiIntervjuPoVideoKlicu(identifikatorKlica)

identiteta ← pridobi avtenticiranega uporabnika

IF identiteta ne obstaja THEN
    zavrni dostop
END IF

intervju ← poišči intervju po identifikatorjuKlica

IF intervju ne obstaja THEN
    RETURN brez rezultata
END IF

jeKandidat ← intervju pripada uporabniku
jeIntervjuvar ← uporabnik je na seznamu intervjuvarjev

IF jeKandidat OR jeIntervjuvar THEN
    RETURN intervju
ELSE
    zavrni dostop
END IF

END ALGORITHM
```

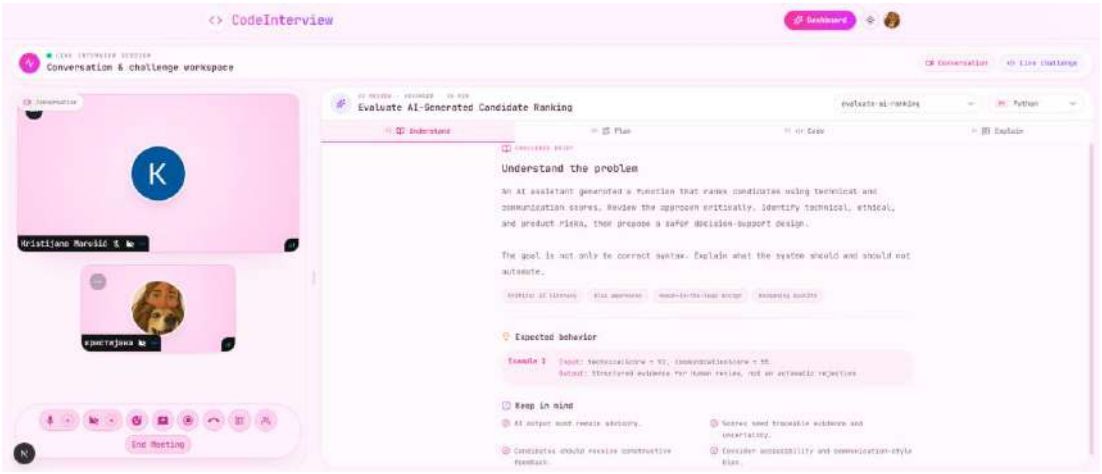
*Psevdokoda 4.4 Preverjanje dostopa do intervjuja.*

Vir: lastni vir.

### 4.3.4 Intervjujski delovni prostor

Po uspešni pridružitvi se uporabniku prikaže intervjujski delovni prostor, ki združuje video komunikacijo in okolje za reševanje tehnične naloge. Video del omogoča prikaz udeležencev, programski del pa kandidatu omogoča prehod med fazami razumevanja problema, načrtovanja, implementacije in razlage rešitve. Postavitev je razdeljena na dva prilagodljiva panela. Na večjih zaslonih sta video komunikacija in tehnična naloga prikazani druga ob drugi, na manjših napravah pa se razporeditev spremeni v navpično. Uporabnik lahko velikost obeh delov spreminja glede na trenutno potrebo. Video del podpira dva načina prikaza.

Mrežni pogled prikaže več udeležencev hkrati, pogled govorca pa izpostavi trenutno aktivnega udeleženca. Uporabnik lahko odpre tudi seznam prisotnih oseb in uporablja standardne kontrole za mikrofon, kamero in zapustitev klica. Kandidat in intervjuvar uporabljata isti delovni prostor, vendar nimata enakih dovoljenj. Kandidat rešuje nalogo in sproti shranjuje svoje delo, intervjuvar pa spremlja potek intervjuja. Možnost zaključka celotnega intervjuja je prikazana samo uporabniku z vlogo intervjuvarja, ki je hkrati tudi lastnik videoklica. Na sliki 4.1 je prikazan intervjujski delovni prostor s sočasnim prikazom video komunikacije in tehnične naloge.




*Slika 4.1 Intervjujski delovni prostor z video komunikacijo in tehnično nalogo.*

Vir: lastni vir.

### 4.3.5 Zaključek video seje

Možnost **End Meeting** zaključi intervju za vse udeležence in je na voljo samo intervjuvarju, ki je lastnik klica. Sistem preveri njegova dovoljenja, dokončno odda kandidatovo rešitev, posodobi stanje intervjuja in sproži generiranje poročila. Nato zaključi video sejo in uporabnika preusmeri na začetno stran. Med postopkom je gumb onemogočen, ob napaki pa se seja ne zaključi. Podrobnosti analize so predstavljene v poglavjih 4.5 in 4.6.

```
ALGORITHM ZaključiIntervju

preveri, ali je uporabnik intervjuvar
preveri, ali je uporabnik lastnik video klica

IF uporabnik nima dovoljenja THEN
    zavrni zaključek
END IF

onemogoči ponovno izbiro zaključka
dokončno oddaj kandidatovo rešitev
posodobi stanje intervjuja
začni generiranje poročila
zaključi video klic za vse udeležence
obvesti uporabnika o začetku analize
preusmeri uporabnika na začetno stran

END ALGORITHM
```

*Psevdokoda 4.5 Zaključevanje intervjuja in začetek generiranja poročila.*

Vir: lastni vir.

## 4.4 Programersko okolje za reševanje nalog

Programersko okolje za reševanje tehničnih nalog je zasnovano kot strukturiran štiristopenjski proces, ki kandidata vodi od razumevanja problema do predstavitve končne rešitve. Vmesnik je med intervjujem razdeljen na video komunikacijo in interaktivni delovni prostor. Kandidat lahko v zgornjem delu izbere tehnično nalogo in enega od programskih jezikov, ki jih izbrana naloga podpira. Postopek sestavljajo štirje koraki, opisani v nadaljevanju.

- **Understand the problem** – kandidat se seznani z opisom naloge, pričakovanim vedenjem rešitve, vhodnimi in izhodnimi primeri, merili ocenjevanja ter pomembnimi omejitvami.
- **Plan** – kandidat pred začetkom implementacije opiše načrt rešitve, predvidene podatkovne strukture, glavne korake in robne primere.
- **Code** – kandidat implementira rešitev v urejevalniku `Monaco Editor` in jo lahko izvede ter preveri njen rezultat.
- **Explain** – kandidat pojasni delovanje rešitve, utemelji ključne odločitve in navede časovno in prostorsko zahtevnost oziroma morebitne kompromise.

Takšna zasnova omogoča vrednotenje širšega procesa reševanja problema in ne samo končne izvorne kode. Intervjuvar lahko tako pridobi vpogled v kandidatovo razumevanje naloge, načrtovanje, implementacijo in sposobnost tehnične razlage.

### 4.4.1 Izbira naloge in programskega jezika

Tehnične naloge so v sistemu vnaprej strukturirane. Vsaka naloga vsebuje identifikator, naslov, opis, težavnost, predvideni čas reševanja, primere vhodnih in izhodnih podatkov, omejitve, merila ocenjevanja ter začetno kodo za podprte programske jezike. Sistem podpira izvajanje kode v programskih jezikih `JavaScript`, `Python`, `Java` in `SQL`, vendar posamezna naloga ne podpira nujno vseh razpoložljivih jezikov. Ob izbiri naloge se zato kandidatu prikažejo samo jeziki, ki so zanjo določeni.

Če trenutno izbrani jezik pri novi nalogi ni podprt, sistem samodejno izbere prvi razpoložljivi jezik. Ob spremembi naloge se delovni prostor ponastavi in naloži začetno kodo izbrane naloge. Ponastavijo se tudi kandidatov načrt, razlaga in rezultat predhodnega izvajanja. Podobno se ob spremembi programskega jezika naloži ustrezna začetna koda in izbriše rezultat prejšnjega izvajanja.

### 4.4.2 Urejanje in samodejno shranjevanje rešitve

Za implementacijo programske rešitve je uporabljen `Monaco Editor`, ki kandidatu omogoča urejanje kode s prikazom številk vrstic, označevanjem sintakse, samodejnim prilagajanjem velikosti in podporo za oblikovanje kode glede na izbrani programski jezik. Kandidatovo delo se med intervjujem samodejno shranjuje v podatkovno zbirko `Convex`. Sistem spremlja spremembe načrta, programske kode, razlage, izbrane naloge, programskega jezika in rezultata izvajanja. Po `1,2` sekunde brez nove spremembe se sproži shranjevanje trenutnega stanja. Med izvajanjem kode je samodejno shranjevanje začasno ustavljeno, da se ne shrani nepopoln rezultat.

Pred shranjevanjem zaledni del preveri identiteto uporabnika in njegovo povezavo z intervjujem. Rešitev lahko spreminja samo kandidat, ki je dodeljen izbranemu intervjuju. Spreminjanje ni dovoljeno po zaključku intervjuja ali po dokončni oddaji rešitve. Če predaja za intervju še ne obstaja, sistem ustvari nov zapis. V nasprotnem primeru posodobi obstoječega in zabeleži čas zadnje spremembe. Posamezna predaja vsebuje podatke o nalogi, izbranem jeziku, načrtu, kodi, razlagi, rezultatu in uspešnosti izvajanja.

```
ALGORITHM ShraniOsnutekRešitve(podatkiPredaje)

počakaj 1,2 sekunde od zadnje spremembe

IF uporabnik ni kandidat OR izvajanje kode poteka THEN
    prekini shranjevanje
END IF

identiteta ← pridobi avtenticiranega uporabnika
intervju ← pridobi pripadajoči intervju

IF identiteta ne obstaja OR intervju ne obstaja THEN
    zavrni zahtevo
END IF

IF uporabnik ni kandidat izbranega intervjuja THEN
    zavrni shranjevanje
END IF

IF intervju je zaključen THEN
    zavrni spreminjanje
END IF

predaja ← poišči predajo za intervju

IF predaja je dokončno oddana THEN
    zavrni spreminjanje
END IF

IF predaja obstaja THEN
    posodobi vsebino in čas zadnje spremembe
ELSE
    ustvari novo predajo
END IF

RETURN identifikator predaje

END ALGORITHM
```

*Psevdokoda 4.6 Samodejno shranjevanje osnutka kandidatove rešitve.*

Vir: lastni vir.

### 4.4.3 Izvajanje programske kode

Kandidat lahko v fazi `Code` izvede napisano programsko kodo in preveri njen rezultat. Uporabniški vmesnik pošlje izbrani jezik in kodo notranji strežniški poti aplikacije, ki zahtevo posreduje storitvi `Piston`. Kandidatova koda se tako ne izvaja neposredno v uporabnikovem brskalniku ali v glavnem aplikacijskem procesu. Pred posredovanjem zahteve sistem preveri, ali je izbrani jezik podprt, ali koda vsebuje vsebino in ali njena dolžina ne presega `20.000` znakov. Podprti jeziki se preslikajo v ustrezna izvajalna okolja in imena datotek.

Čas prevajanja je omejen na deset sekund, čas izvajanja pa na tri sekunde. Za obe fazi je določena pomnilniška omejitev `128 MB`. Po izvedbi sistem združi izpis prevajanja in izvajanja. Rezultat se uporabniku prikaže v izhodnem delu urejevalnika skupaj s podatkom, ali je bilo izvajanje uspešno. Izpis in uspešnost se nato vključita v samodejno shranjeni osnutek in pozneje predstavljata del vhodnih podatkov za analizo kandidatove rešitve.

```
ALGORITHM IzvediProgramskoKodo(jezik, koda)

IF jezik ni podprt THEN
    RETURN napaka o nepodprtem jeziku
END IF

IF koda je prazna THEN
    RETURN napaka o manjkajoči kodi
END IF

IF dolžina kode presega dovoljeno omejitev THEN
    RETURN napaka o preseženi dolžini
END IF

izvajalnoOkolje ← določi okolje za izbrani jezik

rezultat ← pošlji kodo storitvi Piston
    z omejitvami časa in pomnilnika

IF storitev zavrne zahtevo THEN
    RETURN sporočilo o napaki
END IF

izpis ← združi izpis prevajanja in izvajanja
uspešno ← prevajanje uspešno AND izvajanje uspešno

RETURN izpis in podatek o uspešnosti

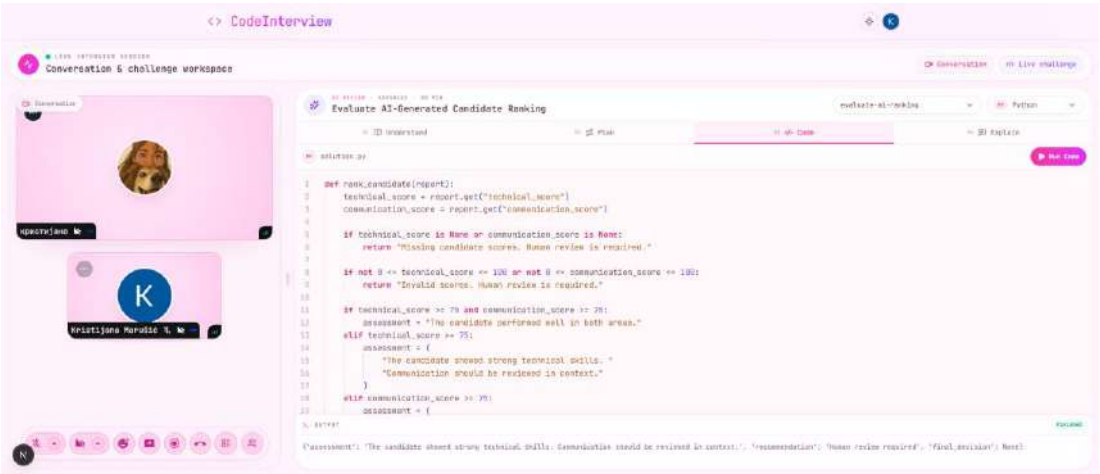
END ALGORITHM
```

*Psevdokoda 4.7 Preverjanje in izvajanje programske kode.*

Vir: lastni vir.

### 4.4.4 Dokončna oddaja rešitve

Kandidat v trenutni izvedbi nima ločenega gumba za dokončno oddajo. Ko pooblaščeni intervjuvar zaključi intervju, sistem poišče pripadajočo kandidatovo predajo in ji zabeleži čas dokončne oddaje. Od tega trenutka predaje ni več mogoče spreminjati. Če kandidat med intervjujem ni ustvaril nobenega osnutka, zaključevanje z generiranjem poročila ni mogoče, saj sistem nima podatkov za analizo. Uspešno zaključena predaja skupaj z opisom naloge, načrtom, programsko kodo, razlago in rezultatom izvajanja predstavlja vhod v postopek samodejnega ocenjevanja, opisan v naslednjem poglavju.




*Slika 4.2 Urejevalnik programske kode s prikazom rezultata izvajanja.*

Vir: lastni vir.

## 4.5 Integracija velikih jezikovnih modelov

Veliki jezikovni model je v sistem vključen za samodejno analizo kandidatovega dela po zaključku tehničnega intervjuja. Za generiranje analize je uporabljen model `Gemini`, ki na podlagi strukturiranega poziva ovrednoti kandidatovo razumevanje naloge, načrt rešitve, programsko kodo, razlago in rezultat izvajanja. Model ne sprejema končne odločitve o izbiri kandidata. Njegov rezultat predstavlja podporno oceno, ki intervjuvarju omogoča hitrejši in bolj strukturiran pregled kandidatovega dela. Končna presoja zato ostaja odgovornost človeka.

### 4.5.1 Začetek postopka analize

Generiranje analize se začne, ko dodeljeni intervjuvar zaključi intervju. Pred začetkom sistem preveri identiteto uporabnika in njegovo povezavo z intervjujem. Postopek lahko sproži samo intervjuvar, katerega identifikator je naveden na seznamu intervjuvarjev izbranega intervjuja. Sistem nato preveri, ali kandidatova predaja obstaja. Če kandidat ni ustvaril nobenega osnutka, analize ni mogoče izdelati, saj model nima podatkov za ocenjevanje. Če predaja obstaja in še ni bila dokončno oddana, sistem zabeleži čas oddaje. Istočasno spremeni stanje intervjuja v zaključeno in shrani čas njegovega konca. Za vsak intervju obstaja največ eno poročilo. Če je poročilo že v stanju `pending`, `generating` ali `completed`, sistem ne ustvari novega zapisa, temveč vrne obstoječega. Če je prejšnji poskus končal z napako, se lahko isti zapis ponovno uporabi in postopek analize ponovno začne. Ob ustvarjanju poročilo najprej pridobi stanje `pending`. Generiranje se nato prek razporejevalnika `Convex` zažene kot notranje asinhrono opravilo. S tem uporabniku ni treba čakati, da se celoten poziv modelu zaključi znotraj zahteve za končanje intervjuja.

```
ALGORITHM ZačniAnalizoIntervjuja(identifikatorIntervjuja)

identiteta ← pridobi avtenticiranega uporabnika
intervju ← pridobi izbrani intervju

IF identiteta ali intervju ne obstaja THEN
    zavrni zahtevo
END IF

IF uporabnik ni dodeljeni intervjuvar THEN
    zavrni zahtevo
END IF

predaja ← poišči kandidatovo predajo

IF predaja ne obstaja THEN
    prekini postopek
END IF

poročilo ← poišči poročilo za intervju

IF poročilo je v obdelavi ali že zaključeno THEN
    RETURN identifikator obstoječega poročila
END IF

dokončno oddaj kandidatovo predajo
označi intervju kot zaključen

IF neuspešno poročilo obstaja THEN
    ponastavi njegovo stanje na "pending"
ELSE
    ustvari novo poročilo s stanjem "pending"
END IF

asinhrono zaženi generiranje analize

RETURN identifikator poročila

END ALGORITHM
```

*Psevdokoda 4.8 Začetek analize kandidatove predaje.*

Vir: lastni vir.

### 4.5.2 Priprava konteksta in poziva

Pred pošiljanjem zahteve modelu sistem iz podatkovne zbirke pridobi intervju in pripadajočo kandidatovo predajo. Pri tem modelu ne posreduje samo kandidatove programske kode, temveč tudi kontekst konkretne tehnične naloge.

Podatki o nalogi vključujejo:

- naslov, opis in težavnost naloge,
- merila ocenjevanja,
- omejitve, ki jih mora rešitev upoštevati,
- izbrani programski jezik.

Podatki o kandidatovem delu vključujejo:

- načrt rešitve iz faze **Plan**,
- programsko kodo iz faze **Code**,
- končno razlago iz faze **Explain**,
- izpis izvajanja in podatek o njegovi uspešnosti.

Faza **Understand** je v analizi zastopana prek opisa izbrane naloge, njene težavnosti, meril ocenjevanja in omejitev. Model lahko zato presoja, ali kandidatov načrt, koda in razlaga ustrezajo zahtevam konkretne naloge. Brez tega konteksta bi lahko tehnično rešitev ocenjeval v napačni smeri. Iz vseh navedenih podatkov sistem sestavi strukturiran poziv. Modelu je v njem dodeljena vloga nepristranskega izkušenega programskega inženirja. Navodila zahtevajo, da oceni samo predložene dokaze in ne sklepa o kandidatovi identiteti, osebnosti, kulturnem ozadju ali drugih osebnih značilnostih.

Model mora posebej oceniti:

- razumevanje problema,
- načrt in pristop k reševanju,
- tehnično pravilnost in upoštevanje omejitev,
- algoritmično zahtevnost in obravnavo robnih primerov,
- čitljivost, strukturo in vzdržljivost programske kode,
- jasnost končne razlage.

Če posamezen del predaje ni izpolnjen, ga mora model obravnavati kot manjkajoč dokaz. Analiza temelji na podatkih, ki se med intervjujem samodejno shranjujejo v zapis `submissions`: kandidatovem načrtu, programski kodi, razlagi, in če je kandidat kodo izvedel, tudi na prikazanem rezultatu izvajanja. Storitev `Piston` kandidatu omogoča, da med reševanjem preveri izhod kode. Shranjevanje predaje in priprava podatkov za model potekata ločeno prek aplikacijskega zaledja. Uspešen zagon programa sam po sebi ne dokazuje pravilnosti rešitve za vse možne vhode, saj lahko koda kljub pravilnemu izhodu pri enem primeru odpove pri robnih primerih ali ne izpolnjuje vseh zahtev naloge. Tudi odsotnost izhoda ne pomeni nujno, da je izvorna koda nepravilna. Model mora zato ugotovitve utemeljiti z razpoložljivo vsebino predaje in razlikovati med lastnostmi, ki jih je mogoče razbrati iz kode, ter vedenjem, ki je bilo dejansko preverjeno z izvajanjem.

### 4.5.3 Strukturiranje in preverjanje rezultata

Zahteva modelu `Gemini` je izvedena v notranjem dejanju `Convex`. Dostopni ključ je shranjen kot okoljska spremenljivka `GEMINI_API_KEY` in se ne posreduje odjemalskemu delu aplikacije. Pred pošiljanjem zahteve sistem stanje poročila spremeni iz `pending` v `generating`. Od modela se zahteva odgovor v obliki strukturiranega objekta `JSON`, ki mora ustrezati vnaprej določeni shemi.

Rezultat vsebuje štiri ocene od `0` do `100`:

- tehnično pravilnost,
- komunikacijo,
- reševanje problemov,
- kakovost programske kode.

Ocena komunikacije temelji na jasnosti pisnega načrta in končne razlage. Sistem ne analizira zvoka, videoposnetka ali prepisa ustnega pogovora. Zato ta ocena ne predstavlja celovite presoje kandidatovega ustnega komuniciranja.

Poleg številskih ocen model vrne:

- konkretne prednosti in slabosti,
- priporočila za izboljšanje,
- povzetek uspešnosti,
- podrobne povratne informacije za vsako področje,
- splošno priporočilo `positive`, `mixed` ali `negative`,
- stopnjo zaupanja `low`, `medium` ali `high`.

Po prejemu odgovora sistem preveri, ali odgovor obstaja, ali ga je mogoče pretvoriti iz zapisa JSON in ali vsebuje vsa zahtevana polja. Skupna ocena se ne prepusti modelu, temveč jo sistem izračuna kot aritmetično sredino štirih področnih ocen:

$$\text{skupna ocena} = \frac{\text{tehnicna ocena} + \text{komunikacija} + \text{resevanje problemov} + \text{kakovost kode}}{4}$$

```
ALGORITHM GenerirajInShraniAnalizo(identifikatorPoročila)

nastavi stanje poročila na "generating"

poskus ← izvedi naslednje:

    pridobi intervju in kandidatovo predajo
    preveri obstoj ključa za dostop do modela

    kontekstNaloge ← pripravi opis, težavnost,
        merila in omejitve izbrane naloge

    kandidatovoDelo ← pripravi načrt, kodo,
        razlago in rezultat izvajanja

    poziv ← združi navodila za ocenjevanje,
        kontekstNaloge in kandidatovoDelo

    odgovor ← pošlji poziv modelu Gemini
        in zahtevaj strukturiran rezultat JSON

    IF odgovor je prazen ali neveljaven THEN
        sproži napako
    END IF

    preveri strukturo in dovoljene vrednosti odgovora

    skupnaOcena ← povprečje štirih področnih ocen

    shrani ocene, povratne informacije,
        priporočilo in stopnjo zaupanja

    nastavi stanje poročila na "completed"
    zabeleži čas zaključka

ob napaki:

    nastavi stanje poročila na "failed"
    shrani sporočilo o napaki

END ALGORITHM
```

*Psevdokoda 4.9 Generiranje, preverjanje in shranjevanje strukturirane analize.*

Vir: lastni vir.

### 4.5.4 Obravnava napak in nadzor dostopa

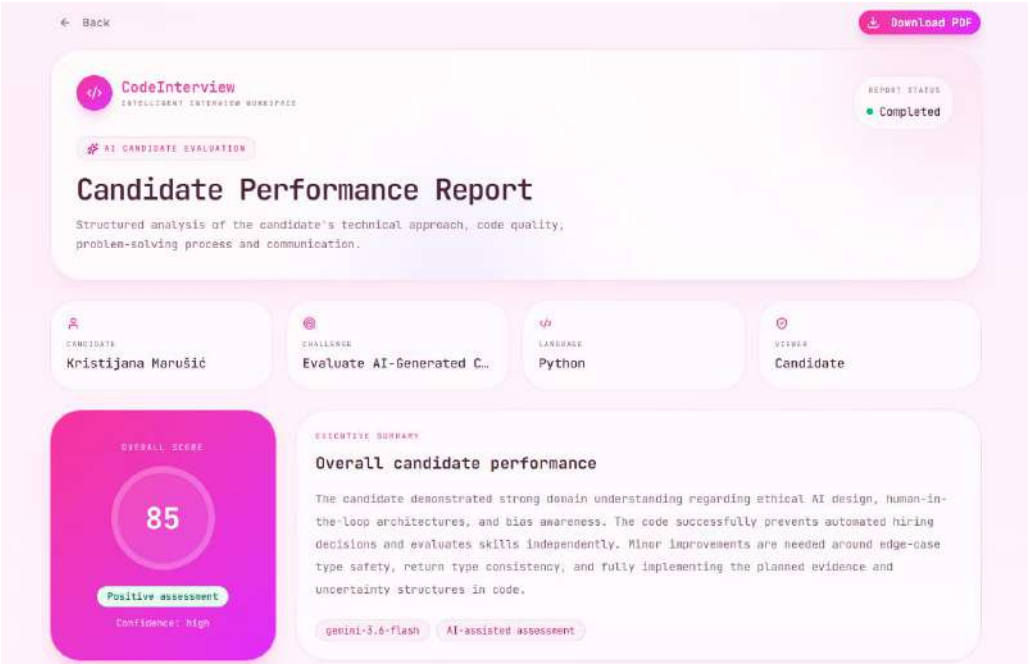
Če okoljska spremenljivka z dostopnim ključem ni nastavljena, če storitev `Gemini` zavrne zahtevo, vrne prazen odgovor ali rezultat ne ustreza pričakovani strukturi, sistem poročila ne označi kot uspešno ustvarjenega. Namesto tega njegovo stanje spremeni v `failed`, shrani opis napake in zabeleži čas zaključka poskusa. Takšno upravljanje stanj omogoča uporabniškemu vmesniku razlikovanje med poročilom, ki čaka na obdelavo, poročilom v procesu generiranja, uspešno zaključenim poročilom in neuspešnim poskusom. Po neuspešnem poskusu je mogoče generiranje ponovno sprožiti. Dostop do analize je omejen na kandidata, na katerega se intervju nanaša.

### 4.5.5 Odgovorna uporaba umetne inteligence

Samodejno ustvarjeno poročilo je namenjeno podpori intervjuvarju in ne nadomešča njegove strokovne presoje. Čeprav poročilo vsebuje splošno priporočilo, sistem na njegovi podlagi kandidata samodejno ne sprejme ali zavrne. Končno odločitev sprejme človek ob upoštevanju celotnega razgovora, kandidatove rešitve in drugih okoliščin. Poziv modelu izrecno omejuje analizo na predloženo tehnično delo in prepoveduje sklepanje o osebnih ter zaščitenih značilnostih kandidata. Kljub temu rezultata ni mogoče obravnavati kot popolnoma objektivnega, saj lahko veliki jezikovni modeli ustvarijo netočne, nedosledne ali pristranske ocene. Posebno previdno je treba razlagati oceno komunikacije. Ta temelji samo na pisnem načrtu in razlagi ter lahko nanjo vplivajo jezikovno znanje, način izražanja, dostopnost in stres med intervjujem. Zato mora intervjuvar posamezne ocene vedno preveriti glede na dejansko vsebino in širši kontekst intervjuja.

## 4.6 Generiranje poročil in povratnih informacij

Po zaključku analize se rezultati uporabniku prikažejo na namenski strani poročila. Podatki se pridobijo iz platforme `Convex` na podlagi identifikatorja intervjuja. Dostop je omogočen samo pripadajočemu kandidatu in dodeljenim intervjuvarjem, medtem ko sistem pri neobstoječem poročilu ali nezadostnih dovoljenjih ne razkrije njegove vsebine. Med generiranjem prikazuje obvestilo o poteku analize in se zaradi reaktivnih poizvedb `Convex` samodejno posodobi po njenem zaključku. Ob neuspešni analizi se prikaže sporočilo o napaki, dodeljeni intervjuvar pa lahko postopek ponovno sproži. Zaključeno poročilo vsebuje skupno oceno, splošno priporočilo in stopnjo zaupanja modela. Posamezne kartice prikazujejo ocene tehničnega znanja, reševanja problemov, kakovosti kode in komunikacije.

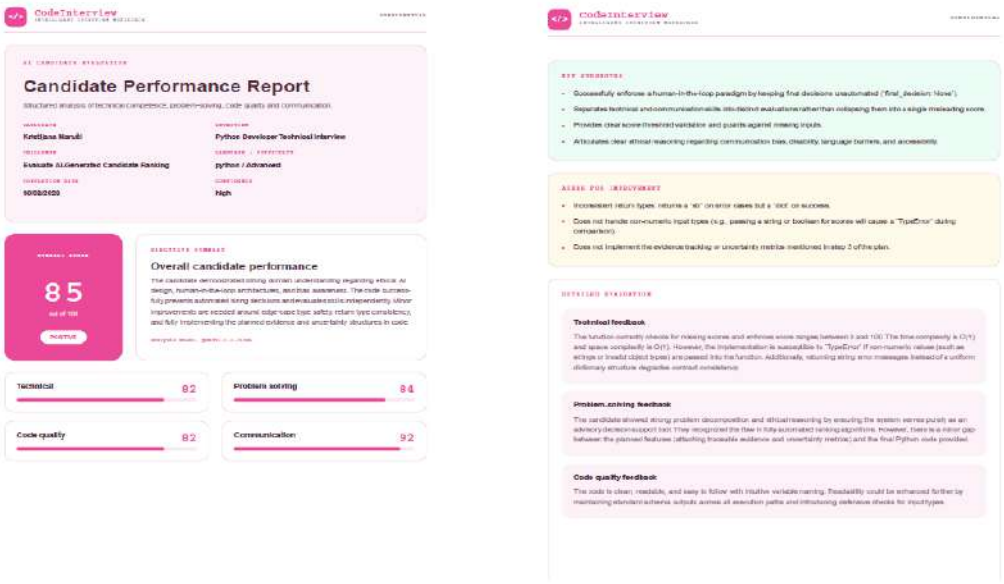



*Slika 4.3 Prikaz strukturiranega poročila o uspešnosti kandidata.*

Vir: lastni vir.

### 4.6.1 Generiranje poročila PDF

Dokument se na zahtevo uporabnika ustvari v brskalniku s knjižnico `React PDF`[13] in se ne shranjuje trajno v podatkovno zbirko. Sistem podatke poročila pretvori v dokument formata `A4` in pripravi datoteko za prenos. `PDF` vsebuje podatke o kandidatu, intervjuju, izbrani nalogi, uporabljenem programskem jeziku, težavnosti naloge in datumu zaključka. Vključuje skupno in področne ocene, povzetek, prednosti, slabosti, podrobne povratne informacije in priporočila. Dodani so tudi oznaka zaupnosti, uporabljeni analizni model in številčenje strani. Po prenosu se začasni spletni naslov odstrani. Dokument vsebuje opozorilo, da poročilo predstavlja podporo pri odločanju, končna ocena kandidata pa ostaja odgovornost intervjuvarja.




*Slika 4.4 Poročilo o analizi kandidata v obliki PDF.*

Vir: lastni vir.

[13] https://www.npmjs.com/package/react-pdf

# 5 Testiranje in evalvacija

## 5.1 Predstavitev testnih primerov

V evalvaciji prototipa je sodelovalo 15 uporabnikov, izbranih med avtoričinimi sodelavci, prijatelji in znanci. Testiranje ni potekalo v okviru dejanskih zaposlitvenih postopkov, temveč v nadzorovanem testnem okolju. Namen testiranja je bil preveriti tehnično delovanje prototipa, izvedljivost celotnega postopka tehničnega intervjuja in odzivanje sistema na različne kandidatove predaje. Vsak sodelujoči je uporabljal svoj uporabniški račun in izvedel simulirani tehnični intervju. Postopek je obsegal prijavo v sistem, dostop do dodeljenega intervjuja, pridružitev video seji, izbiro programerske naloge in programskega jezika, pripravo načrta rešitve, pisanje in izvajanje programske kode, dodajanje razlage ter zaključek intervjuja. Po oddaji je sistem kandidatovo delo analiziral z velikim jezikovnim modelom in pripravil strukturirano poročilo. Poleg uporabniškega preizkusa je bilo izvedeno funkcionalno in scenarijsko testiranje posameznih funkcionalnosti. Preverjeni so bili registracija in sinhronizacija uporabnikov, nadzor dostopa, ustvarjanje in pridružitev video seji, samodejno shranjevanje kandidatovega dela, izvajanje programske kode, zaključek intervjuja, avtomatizirana analiza predaje ter izvoz poročila v obliki PDF.

Testiranje je vključevalo tudi predaje različnih ravni kakovosti, s katerimi je bilo preverjeno, ali se ustvarjene ocene in povratne informacije smiselno prilagajajo vsebini kandidatovega načrta, kode, razlage in rezultata izvajanja. Evalvacija je preverjala uspešnost funkcionalnosti, dokončanje postopka brez kritične napake, skladnost poročila s predajo, razlikovanje med kakovostjo predaj in preglednost ter razumljivost vmesnika. Sodelujoči so po testiranju podali ustne odzive, ki so bili povzeti opisno. Ker ni bilo kontrolne skupine, časovne primerjave ali neodvisne strokovne presoje poročil, rezultatov ni mogoče posplošiti na dejanske zaposlitvene postopke.

| Testni primer | Pričakovani rezultat | Rezultat |
|---|---|---|
| Registracija novega uporabnika | Ustvarjen zapis z vlogo kandidata | Uspešno |
| Dostop do tujega intervjuja ali poročila | Dostop je zavrnjen | Uspešno |
| Video intervju in programsko okolje | Obe funkcionalnosti delujeta v skupnem vmesniku | Uspešno |
| Shranjevanje in izvajanje kode | Osnutek in rezultat se shranita | Uspešno |
| Analiza različno kakovostnih predaj | Ocene se prilagodijo kakovosti dokazov | Uspešno |
| Izvoz zaključenega poročila | Ustvari se dokument PDF | Uspešno |

*Tabela 5.1 Rezultati funkcionalnega in scenarijskega testiranja.*

Vir: lastni vir.

## 5.2 Preverjanje hipotez

**H1** na podlagi izvedenega testiranja ni bilo mogoče v celoti potrditi. Rezultati so pokazali, da sistem znotraj enotne spletne platforme uspešno povezuje organizacijo intervjuja, video komunikacijo, reševanje tehnične naloge, samodejno shranjevanje in pripravo poročila. To kaže na potencial za zmanjšanje števila ločenih orodij in ročnih korakov. Ker pa čas, poraba virov in uspešnost izvedbe niso bili neposredno primerjani s tradicionalnim postopkom tehničnega intervjuja, ni mogoče zanesljivo trditi, da je razviti sistem učinkovitejši od tradicionalnih pristopov.

**H2** je bila delno podprta na izbranih testnih primerih. Sistem je za predaje različnih ravni kakovosti ustvaril strukturirane ocene, povzetke, prednosti, slabosti in priporočila, ki so se vsebinsko prilagajali kandidatovemu načrtu, kodi, razlagi in rezultatu izvajanja. S tem je bila potrjena zmožnost sistema za pripravo strukturiranih in kontekstualno prilagojenih povratnih informacij. Ker njihova kakovost ni bila formalno primerjana s povratnimi informacijami neodvisnih strokovnih ocenjevalcev, ni mogoče dokazati, da so kakovostnejše od povratnih informacij pri tradicionalnem pristopu.

**H3 ni bila empirično potrjena.** Uporaba enakih meril in izhodne strukture lahko prispeva k večji doslednosti ocenjevanja, vendar brez primerjave več neodvisnih človeških ocenjevalcev ni mogoče zanesljivo dokazati, da sistem zmanjšuje vpliv subjektivnih dejavnikov.

**H4** je bila kvalitativno podprta na podlagi uporabniškega testiranja s `15` sodelujočimi. Vsi sodelujoči so uspešno izvedli simulirani tehnični intervju in uporabniški vmesnik ocenili kot pregleden, vizualno privlačen in enostaven za uporabo. Pozitivno so ocenili tudi razdelitev reševanja naloge na štiri razumljive korake in pregleden prikaz rezultatov, kar podpira trditev, da sistem izboljšuje uporabniško izkušnjo in transparentnost tehničnega intervjuja.

**H5** je bila potrjena s funkcionalnim testiranjem. Testni uporabniki so se lahko pridružili video seji in znotraj iste spletne platforme uporabljali programsko okolje, pripravili načrt, napisali in izvedli kodo ter dodali razlago. Po zaključku intervjuja je sistem uspešno ustvaril strukturirano analizo in poročilo `PDF`, kar potrjuje izvedljivost celotnega integriranega postopka.

## 5.3 Analiza rezultatov

Rezultati so bili vsebinsko skladni z vnaprej pripravljeno kakovostjo testnih predaj. Ocene kakovosti kode so se gibale od `0` do `95`, ocene pisne komunikacije pa od `0` do `98`. Pri nepopolnih predajah je model prepoznal manjkajoče dokaze, pri kakovostnejših rešitvah pa izpostavil pravilne tehnične odločitve, strukturo kode in jasnost razlage. Rezultati kažejo, da sistem smiselno razlikuje med različnimi ravnmi kakovosti, vendar ne predstavljajo statistične potrditve njegove objektivnosti.

## 5.4 Prednosti in omejitve sistema

Glavna prednost sistema je povezovanje celotnega tehničnega intervjuja v enoten delovni proces. Strukturirana analiza zagotavlja sledljive ocene in konkretne povratne informacije, končna odločitev pa ostaja intervjuvarju. Omejitve evalvacije predstavljajo majhno število testnih primerov, odsotnost formalne ankete, časovne primerjave in več neodvisnih ocenjevalcev. Model prav tako analizira pisno predajo, ne pa videa ali govora. Za zanesljivejšo potrditev hipotez bi bilo potrebno testiranje z več kandidati in strokovnjaki v dejanskem zaposlitvenem okolju.

# 6 SKLEP

V diplomskem delu je bil razvit funkcionalen prototip sistema za izvedbo tehničnih intervjujev, ki povezuje video komunikacijo, strukturirano reševanje programerskih nalog in analizo kandidatove predaje z velikim jezikovnim modelom. Izhodišče dela je bila razdrobljenost tehničnih intervjujev med več ločenih orodij in potreba po preglednejšem zbiranju ter obravnavi kandidatovih rešitev. Razviti prototip te aktivnosti združuje v enoten delovni proces, v katerem lahko kandidat rešuje nalogo, intervjuvar pa spremlja intervju in po njegovem zaključku pregleda poročilo. S tem je bil dosežen osrednji razvojni cilj diplomskega dela: vzpostaviti sistem, ki podpira izvedbo tehničnega intervjuja in strokovno presojo kandidatovega dela. Sistem omogoča varno prijavo in ločevanje uporabniških vlog, ustvarjanje intervjujev, video komunikacijo, izbiro programerske naloge in jezika, samodejno shranjevanje kandidatovega dela ter izvajanje programske kode. Kandidatovo reševanje je organizirano v štiri korake: razumevanje naloge, načrtovanje rešitve, pisanje kode in razlago. Takšna zasnova omogoča zbiranje podatkov o postopku reševanja, saj poleg končne kode vključuje tudi kandidatov načrt in pojasnilo sprejetih odločitev. Samodejno shranjevanje podpira ohranjanje oddanih vsebin med delom in zagotavlja podatkovno osnovo za poznejšo analizo.

Po zaključku intervjuja `Gemini` analizira kontekst naloge in kandidatovo predajo ter ustvari strukturirano poročilo, ki ga lahko pooblaščeni uporabnik pregleda v aplikaciji ali izvozi v `PDF`. Pri analizi je pomembno, da se posamezne ugotovitve povezujejo s konkretnimi dokazi v kandidatovi predaji. Uspešno izvajanje kode samo po sebi še ne dokazuje pravilnosti rešitve, odsotnost posameznega dela predaje pa omejuje možnosti presoje tega vidika kandidatovega znanja. Poročilo je zato treba razumeti kot pomoč pri pregledu zbranih vsebin, pri čemer mora intervjuvar preveriti utemeljenost ugotovitev in jih umestiti v širši kontekst intervjuja. Končna odločitev ostaja v pristojnosti človeka.

V uporabniškem testiranju je sodelovalo 15 oseb, ki so v simuliranem okolju izvedle tehnični intervju. Rezultati funkcionalnega, scenarijskega in uporabniškega testiranja so pokazali, da prototip omogoča izvedbo celotnega postopka intervjuja in pripravo strukturiranih ter vsebinsko prilagojenih poročil. H4 je bila kvalitativno podprta z uporabniškim testiranjem, H5 pa s funkcionalnim testiranjem. H2 je bila delno podprta, medtem ko H1 in H3 zaradi odsotnosti primerjalnega oziroma kontroliranega raziskovalnega postopka ni bilo mogoče empirično potrditi. Rezultati tako podpirajo izvedljivost razvite rešitve v preizkušenih scenarijih, ne omogočajo pa dokončnih sklepov o njeni učinkovitosti v dejanskih zaposlitvenih postopkih.

Pri interpretaciji rezultatov je treba upoštevati omejen obseg raziskave. Testiranje je potekalo na majhnem vzorcu in v simuliranem okolju, brez kontrolne skupine ter neodvisnega strokovnega vrednotenja avtomatiziranih ocen. Zato ni bilo mogoče zanesljivo ugotoviti, ali sistem v primerjavi z običajnim tehničnim intervjujem zmanjša čas ocenjevanja, izboljša kakovost presoje ali poveča njeno objektivnost. Za preverjanje teh učinkov bi bilo treba določiti merljive kazalnike in primerjati rezultate pri enakih nalogah ter primerljivih pogojih. Pomembna omejitev je tudi uporaba velikega jezikovnega modela, katerega ugotovitve so lahko nepopolne ali napačne, četudi so predstavljene v prepričljivi in strukturirani obliki.

Prispevek diplomskega dela je predvsem v zasnovi in izvedbi celovitega prototipa, ki povezuje komunikacijo, reševanje nalog, shranjevanje kandidatovega dela in pripravo poročila. Razvoj je pokazal, kako je mogoče različne storitve vključiti v skupen uporabniški proces in avtomatizirano analizo povezati s kontekstom konkretne naloge. Hkrati je delo izpostavilo pomen jasnega ločevanja uporabniških vlog, preverjanja dovoljenj in nadzorovanega dostopa do kandidatovih podatkov. Te zahteve so pomembne za nadaljnji razvoj sistema, zlasti ob morebitnem prehodu iz prototipnega v produkcijsko okolje.

Nadaljnji razvoj bi moral vključevati testiranje z več kandidati in strokovnimi ocenjevalci, formalno merjenje uporabniške izkušnje ter primerjavo človeških in avtomatiziranih ocen. Posebno pozornost bi bilo smiselno nameniti skladnosti ocen, ponovljivosti analize in primerom, v katerih se strokovna presoja razlikuje od ugotovitev modela. Takšna primerjava bi omogočila natančnejše prepoznavanje prednosti in omejitev avtomatiziranega poročanja ter izboljšanje navodil in meril, na katerih temelji analiza.

Smiselne funkcionalne nadgradnje so tudi skupno spremljanje kode v realnem času, ponovno nalaganje shranjenega osnutka in produkcijska namestitev storitve `Piston`. Sistem bi bilo mogoče dodatno prilagoditi nalogam in merilom ocenjevanja posameznih organizacij. Pri tem bi bilo treba ohraniti preglednost meril in zagotoviti, da poročila jasno prikazujejo, na katerih delih kandidatove predaje temeljijo posamezne ugotovitve. Takšne izboljšave bi podprle uporabo sistema v različnih intervjujskih scenarijih in olajšale strokovni pregled rezultatov. Možna nadaljnja razširitev je analiza govora oziroma prepisa intervjuja, ki v trenutnem prototipu ni vključena. Ta bi lahko zagotovila dodatne podatke o kandidatovem načinu razmišljanja in ustni razlagi rešitve, vendar bi zahtevala jasno soglasje uporabnikov, ustrezno varovanje posnetkov ter previdno obravnavo jezikovnih razlik in komunikacijskih posebnosti. Razviti prototip predstavlja osnovo za nadaljnje raziskovanje takšnih možnosti. Njegova uporabna vrednost je v podpori preglednemu tehničnemu intervjuju in obravnavi kandidatovega dela, pri čemer strokovna odgovornost za ocenjevanje in končno odločitev ostaja pri intervjuvarju.

# Literatura


Armstrong, L., Liu, A., MacNeil, S., & Metaxa, D. (2024). The silicon ceiling: Auditing GPT's race and gender biases in hiring. *arXiv*, 1–18. Retrieved from https://doi.org/10.48550/arXiv.2405.04412

Aryan, A. (2025). *LLMOps: Managing large language models in production.* O'Reilly Media.

Brintha, N., Vardhan, A., Kumar, E., Babu, C., & Raj, G. (2025). Design and implementation of an AI-powered interview preparation platform using advanced LLM architectures. *2025 4th International Conference on Automation, Computing and Renewable Systems (ICACRS)* (pp. 1721–1725). IEEE. Retrieved from https://doi.org/10.1109/ICACRS67045.2025.11324218

Clerk. (n.d.). *Clerk documentation*. https://clerk.com/docs

Convex. (n.d.). *Convex developer hub*. https://docs.convex.dev/

Docker, Inc. (n.d.). *Docker documentation*. https://docs.docker.com/

Gallegos, I., Rossi, R., Barrow, J., Tanjim, M., Kim, S., Dernoncourt, F., . . . Ahmed, N. (2024). Bias and fairness in large language models: A survey. *Computational Linguistics, 50*(3), 1097-1179. Retrieved from https://doi.org/10.1162/coli_a_00524

Gilbert, R. (2019). *Inclusive design for a digital world: Designing with accessibility in mind.* Berkeley, CA: Apress.

Gollnick, B. (2025). *Generative AI with Python: The developer's guide to pretrained LLMs, vector databases, retrieval augmented generation, and agentic systems.* Rheinwerk Computing.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* Cambridge, MA: MIT Press.

Google. (n.d.). *Gemini API documentation*. https://ai.google.dev/gemini-api/docs

Johnson, J. (2010). *Designing with the mind in mind: Simple guide to understanding user interface design rules.* Burlington, MA: Morgan Kaufmann.

Jurafsky, D., & Martin, J. (2009). *Speech and language processing.* Upper Saddle River, NJ: Pearson.

Manning, C., & Schütze, H. (1999). *Foundations of statistical natural language processing.* Cambridge, MA: MIT Press.

Microsoft. (n.d.-a). *Monaco Editor*. https://microsoft.github.io/monaco-editor/

Microsoft. (n.d.-b). *TypeScript documentation*. https://www.typescriptlang.org/docs/

Piston. (n.d.). *Piston API documentation*. https://piston.readthedocs.io/en/latest/api-v2/

React-pdf. (n.d.). *React-pdf documentation.* https://react-pdf.org/

Reddy, K., Priya, K., Vasavi, A., Ramesh, A., & Venu Gopal, S. (2026). A context-aware generative AI framework for automated interview evaluation and intelligent feedback generation. *International Journal of Creative Research Thoughts*, 823–830. Retrieved from https://ijcrt.org/viewfulltext.php?p_id=IJCRT2604823

shadcn. (n.d.). *shadcn/ui*. https://ui.shadcn.com/

Singh, A., & Kumar, A. (2025). Interview assistant—AI powered interview preparation and assessment platform. *International Journal of Scientific Research in Engineering and Management, 9*(10), 1-9. Retrieved from https://www.researchgate.net/publication/397139306_Interview_Assistant_-_AI_Powered_Interview_Preparation_and_Assessment_Platform

Skubic, V. (2025). *Sistem za pametno iskanje zaposlitve.* Univerza v Ljubljani, Fakulteta za računalništvo in informatiko. Ljubljana: Repozitorij Univerze v Ljubljani. Retrieved from https://repozitorij.unilj.si/IzpisGradiva.php?id=174277

Stream. (n.d.). *Video and audio API documentation*. https://getstream.io/video/docs/

Svix. (n.d.). *Svix documentation*. https://docs.svix.com/

Tailwind Labs. (n.d.). *Tailwind CSS documentation*. https://tailwindcss.com/docs

Tigani, S. (2025). Next-gen HR interview process parametric AI copilot design based on pretrained LLM and sentiment analysis deep learning models. *Pattern Analysis and Applications, 28*, Article 205. Retrieved from https://doi.org/10.1007/s10044-025-01581-z

Vercel. (n.d.). *Next.js documentation*. https://nextjs.org/docs