Šolski center Celje
Srednja šola za kemijo, elektrotehniko in računalništvo

Povezovanje umetne inteligence in virtualnega avatarja

Raziskovalna naloga

Avtorji:
Vladan Railič, R-4.a
Miha Kos, R-4.a
Nico Jagodič, R-4.a

Mentor:
Matic Holobar, Inž. informatike

Mestna občina Celje, Mladi za Celje
Celje, marec 2024

IZJAVA

Mentor/-ica Matic Holobar v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom Povezovanje umetne inteligence in virtualnega avatarja, katere avtorjio smo Vladan Railič, Miha Kos in Nico Jagodič:

– besedilo v tiskani in elektronski obliki istovetno,

– pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,

– da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,

– da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,

– da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,

– da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 8.4.2024 žig šole Podpis mentorja

Podpis odgovorne osebe

*

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

Zahvala

Zahvaljujemo se vsem, ki so sodelovali pri ustvarjanju raziskovalne naloge. Posebna zahvala gre našemu mentorju, Maticu Holobarju, za njegovo inovativno idejo ter za neizmerno predanost in trud, ki ju je vložil v razvoj raziskovalne naloge. Brez njegovega vodstva in strokovnega znanja ne bi mogli doseči tega, kar smo dosegli. Hvala vam za navdihujočo podporo in usmerjanje skozi celoten proces.

POVEZOVANJE UMETNE INTELIGENCE IN VIRTUALNEGA AVATARJA

POVZETEK

Raziskovalna naloga, se osredotoča na uporabo umetne inteligence v kontekstu izboljšanja učnega procesa z uporabo virtualnega pomočnikav katerem imamo tri hipoteze, ki temeljijo na uporabi virtualnega pomočnika v primerjavi z neposrednim vprašanjem profesorju v realnem času. Te hipoteze zajemajo preferenco uporabnikov, potencialno korist za dijake z učnimi težavami ter verjetnost napake virtualnega profesorja. Raziskava se opira na znanstvene članke, raziskovalne študije in osebna mnenja ter izkušnje. Osnovni cilj je razumeti uporabniške preference in potrebe, zlasti dijakov z učnimi težavami, v kontekstu uporabe virtualnih pomočnikov v izobraževalnih okoljih. Projekt uporablja JavaScript in TypeScript v okviru ogrodja Next.JS, API-je od OpenAI-ja in VoiceLabs-a. Opisuje se tudi razlika med JavaScriptom in Javo ter razloži pomembne vidike uporabe Reacta, Node.js in Next.js. Za avtentikacijo in upravljanje pravic dostopa uporabnikov se uporablja Auth.js, ki omogoča preverjanje identitete in upravljanje s pravicami dostopa. Prisma se uporablja za strukturiranje podatkovne baze in vzpostavljanje relacij med podatkovnimi modeli. Obravnavane so tudi druge pomembne komponente, kot so poti v NextJS, delovanje strežniških in uporabniških operacij, ter definicije shem za preverjanje podatkov v različnih kontekstih.

Ključne besede: aplikacija VirtualniProfesor, OpenAI, umetna inteligenca, JavaScript

INTEGRATING ARTIFICIAL INTELLIGENCE AND VIRTUAL AVATAR

ABSTRACT

The research project focuses on the use of artificial intelligence in the context of improving the learning process through the use of a virtual assistant, with three hypotheses based on the use of a virtual assistant compared to direct questioning of the teacher in real-time. These hypotheses cover user preferences, potential benefits for students with learning difficulties, and the likelihood of error by the virtual teacher. The research relies on scientific articles, research studies, personal opinions, and experiences. The primary goal is to understand user preferences and needs, especially those of students with learning difficulties, in the context of using virtual assistants in educational environments. The project uses JavaScript and TypeScript within the Next.JS framework, APIs from OpenAI and VoiceLabs. It also describes the difference between JavaScript and Java and explains important aspects of using React, Node.js, and Next.js. Auth.js is used for authentication and managing user access rights, allowing identity verification and access rights management. Prisma is used for structuring the database and establishing relationships between data models. Other important components are also discussed, such as paths in NextJS, server-side and client-side operations, and schema definitions for data validation in various contexts.

Keywords: VirtualProfessor application, OpenAI, artificial intelligence, JavaScript

KAZALO VSEBINE

KAZALO SLIK

1 UVOD

Umetna inteligenca je veja računalništva, ki se posveča razvoju sistemov, sposobnih posnemati in simulirati človeško inteligenco. Glavni cilj te tehnologije je omogočiti sistemom, da se spopadajo z nalogami, ki zahtevajo človeško inteligenco, kot so zaznavanje okolja, učenje iz izkušenj, prepoznavanje vzorcev in odločanje. Trenutno to področje doživlja hitro rast, zlasti v sistemih, ki bi lahko pomagali pri splošnem učenju. Kljub temu bi morda ljudje potrebovali nekaj bolj prepoznavnega kot le besedilo.

Rešitev bi lahko bila uporaba avatarja, ki bi razlagal vaša vprašanja. Takšen pristop bi lahko pospešil proces pridobivanja znanja, saj bi dodal bolj naraven element v učni proces. Namen naše raziskovalne naloge je prav raziskati to možnost in razviti učinkovit sistem, ki bi omogočil boljše razumevanje in hitrejše usvajanje znanja z uporabo avatarja kot interaktivnega učiteljskega orodja.

1.1 HIPOTEZE

Za našo raziskovalno nalogo smo uporabili naslednje hipoteze:

1. Dijaki bodo verjetneje uporabili virtualnega pomočnika kot pa se odločili za neposredno vprašanje profesorja v realnem času.
2. Uporaba našega orodja se lahko izkaže kot potencialna rešitev za dijake z učnimi težavami.
3. Virtualni profesor bo verjetno naredil napako v približno 10% primerov.

Izbira teh hipotez izhaja iz želje razumeti resničnih preferenc uporabnikov pri izbiri med uporabo virtualnega pomočnika in neposrednim vprašanjem profesorju. Posebno nas zanima mnenje dijakov, ki imajo učne probleme in jim učenje ne zanima preveč. Naša raziskava si prizadeva raziskati in osvetliti to raznolikost v odzivih, kar bo prispevalo k boljšemu razumevanju uporabniških navad in potreb v kontekstu uporabe virtualnih pomočnikov v izobraževalnih okoljih.

1.2 METODA RAZISKOVANJA

V okviru naše raziskovalne naloge smo proučevali uporabo umetne inteligence v kombinaciji z avatarjem za izboljšanje učenja. Za raziskovanje smo se opirali na različne vire, kot so znanstveni članki in raziskovalne študije, ki so obravnavale podobne tematike. Predvsem smo se osredotočili na prakse

uporabe umetne inteligence in avatarjev v izobraževalnem okolju. Naš pristop k raziskavi je vključeval tudi osebno mnenje in izkušnje, saj smo želeli razumeti, kako bi takšna tehnologija lahko koristila posameznikom pri učenju. Skozi analizo različnih virov smo oblikovali mnenje o potencialnih prednostih in izzivih, ki bi jih prinesla uporaba AI avatarjev v izobraževalnih procesih.

2 UMETNA INTELIGENCA

Definicija umetne inteligence je da je umetna inteligenca sposobnost stroja da posnema človeške sposobnosti kot sta logično razmišljanje, učenje ter ustvarjalnost vsaj tako je bilo definirano v evropske parlamentu. Definicija poudarja vsestranskost umetne inteligence, ki ne sledi zgolj eni specifični nalogi, temveč posnema širši spekter človeških sposobnosti. Ta raznolikost omogoča uporabo umetne inteligence v različnih sektorjih, od znanosti in industrije do vsakdanjega življenja.

2.1 UPORABA UMETNE INTELIGENCE

Ta tehnologija je seveda uporabljeno v zelo velikem spektru aplikacij saj je lahko ta tehnologija zelo uporabna pri težjih procesah. Sposobnost umetne inteligence, da analizira ogromne količine podatkov, prepoznava vzorce in se uči iz izkušenj, pomeni, da lahko prispeva k učinkovitemu reševanju problemov na različnih področjih.

2.1.1 TIPI STROK V KATERIH JE UPORABLJEN

Umetna inteligenca (UI) se uporablja na različnih področjih v Sloveniji. Nekatera področja, kjer je umetna inteligenca že prisotna ali se pričakuje njen širši vpliv, vključujejo:

1. Zdravstvo: V medicini se uporabljajo algoritmi umetne inteligence za diagnosticiranje bolezni, analizo medicinskih slik, prilagajanje individualnih zdravstvenih načrtov ter izboljšanje učinkovitosti zdravstvenih postopkov.
2. Finančni Sistem: Umetna inteligenca se uporablja v finančnih institucijah za analizo tržnih trendov, upravljanje portfeljev, prepoznavanje goljufij in avtomatizacijo procesov.
3. Industrija: V proizvodnji se umetna inteligenca uporablja za optimizacijo proizvodnih procesov, vzdrževanje naprav, upravljanje zalogo in napovedovanje vzdrževanja opreme.

4. Izobraževanje: V izobraževanju se uporabljajo sistemi umetne inteligence za prilagajanje učnega načrta posameznim študentom, avtomatizacijo ocenjevanja in razvoj izobraževalnih platform.
5. Transport: V prometu se umetna inteligenca uporablja za optimizacijo prometnih tokov, avtomatizacijo vozil, izboljšanje varnosti in razvoj pametnih prometnih sistemov.

2.1.2 TEHNIKE UMETNE INTELIGENCE

Imamo več vrst tehnik ki se uporabljajo v umetni inteligenci, odvisno za kateri namen bos uporabil umetno inteligenco se ti bo tudi spremenilo potreba katere tehnike bi rad uporabil. Najbolj uporabljene oziroma znane tehnike ki so uporabljene so:

1. Strojno učenje – Ta tehnika učenja je uporabljena tako, da se računalnik uči iz podatkov ko so mu na voljo. Strojno učenje se deli se na tri osnovne kategorije te so nadzorovano učenje vključuje učenje iz označenih podatkov, nenadzorovano učenje omogoča razumevanje vzorcev v nepomembnih podatkih, medtem ko ojačevalno učenje poudarja učenje odločanja za maksimizacijo nagrade v določenem okolju. Te kategorije omogočajo prilagajanje različnim vidikom učenja.
2. Globoko učenje – Deluje tako da uporablja globoke nevronske mreže z več plastmi ta tehnika je zelo uporabna pri obdelavi zelo kompleksnih podatkov kot so slike in zvok. Ta tehnika je tudi podvrsta strojnega učenja.
3. Obedelava računalniškega učenja – Ta tehnika je pa je uporabljena za razumevanja jezika ter prevanjanja različnih jezikov.
4. Računalniški vid – Pri tej tehniki so pa uporabljeni različni računalniški alrgoritmi za prepoznavanje ir razumevanje vizualnih informacij. Zato je uporabljeno pri prepoznavanje predmetov na sliki oziroma videoposnetku ta tehnologija se zdaj tudi uporablja pri avtonomnih vozilah.
5. Iskanje - Iskalni algoritmi so namenjeni odkrivanju optimalnih rešitev z iskanjem prostora možnih rešitev. Ta metoda je ključna za iskanje učinkovitih rešitev v kompleksnih sistemih in optimizacijo delovanja algoritmov.

Moramo tudi izpostaviti, da nove tehnike ki se uporabljajo v umetni inteligenci se nenehno pojavljajo saj se nove in boljse pojavljajo zaradi napredkov v tehnologiji ter raziskavi.

2.2 ZGODOVINA UMETNE INTELIGENCE

Koncepti o umetni inteligenci že obstajajo dlje časa najbolj znana oseba ki je te koncepte začela je bil Alan Turing leta 1936 ko je predstavil konecpt turingovega stroja ki je uporabljena za teoretične raziskave ručunanja. Pozneje je tudi postavil vprašanje ali lahko stroj razmišlja, po tem vprasanju je pa predlagav turingov test kjer bi človek poskušal razlikovati med besedami računalnika in človeka tako da ne bi vedel kateri odgovori so od koga. Ampak izraz umetna inteligenca ki je prvič uporabljen leta 1956 v Darthmouth konferenci kjer so udeleženci med njimi John McCarthy, Marvin Minsky in Claude Shannon, postavili temelje za raziskovanje strojnega učenja in umetne inteligence kot ločenih ved.

Prvi poskusi umetne inteligence so se pa dogajali v 50. in 60. letih najbolj pomembni programi ki so bili ustvarjeni v tem obdobju so bili Logic Theorist in General Problem Solver. Ampak potem je sledilo obdobje ki je bil interes v umetno inteligenco zmanjšan zato so bile raziskave omejene in so se samo razvijali sistemi strojnega učenja kot so ekspertni sistemi. V 90. in 2000. se pa prišlo do ponovnega vzpona umetne inteligence saj zaradi večje računalniške moči so bile ustvarjene nove metode umetne inteligence, ena izmed najbolj pomembnih je bila metoda globokega učenja ki je pripomogli v prepoznavanje vzorcev. Zdaj v sedanjosti je pa ena izmed največjih razlih sama količina podatkov ki lahko pošljemo stroji da se uči iz in kako hitro se iz teh podatkov nauči to kaj je potrebno.

2.3 FILOZOFIJA, KI SE UKVARJA S UMETNO INTELIGENCO

V umetni inteligenci so imeli tudi filozofi vpliv, saj je s to tehnologijo postavljenih zelo veliko filozofskih vprašanj. Med vsemi temi vprašanji, ki so se pojavila, se mi zdi, da so najpomembnejša ta:

1. Narava inteligence

   Veliko filozofov se spršuje kaj sploh pomeni biti inteligenten, ali je to samo izvajanje nalog in reševanje problemov ali je inteligenca zavedanje, prilagodljivost in ustvarjalnost.
2. Zavest in samozavedanje

   Filozofska vprašanja o zavesti in samozavedanju so ključna v razpravi o umetni inteligenci. Ali je mogoče ustvariti algoritemski sistem, ki dejansko dojema svoje obstoječe stanje in ima občutek zase?
3. Moralne in etične dileme

   Razvoj umetne inteligence postavlja moralna in etična vprašanja glede odločitev, ki jih algoritmi sprejemajo. Filozofi se sprašujejo, kako vgraditi moralne smernice v algoritme ter kako

preprečiti in odpraviti morebitne pristranskosti. Vprašanje etične odgovornosti v razvoju in uporabi umetne inteligence je ključno za ustvarjanje družbeno sprejemljivih tehnologij.

2.4 PRIHODNOST UMETNE INTELIGENCE

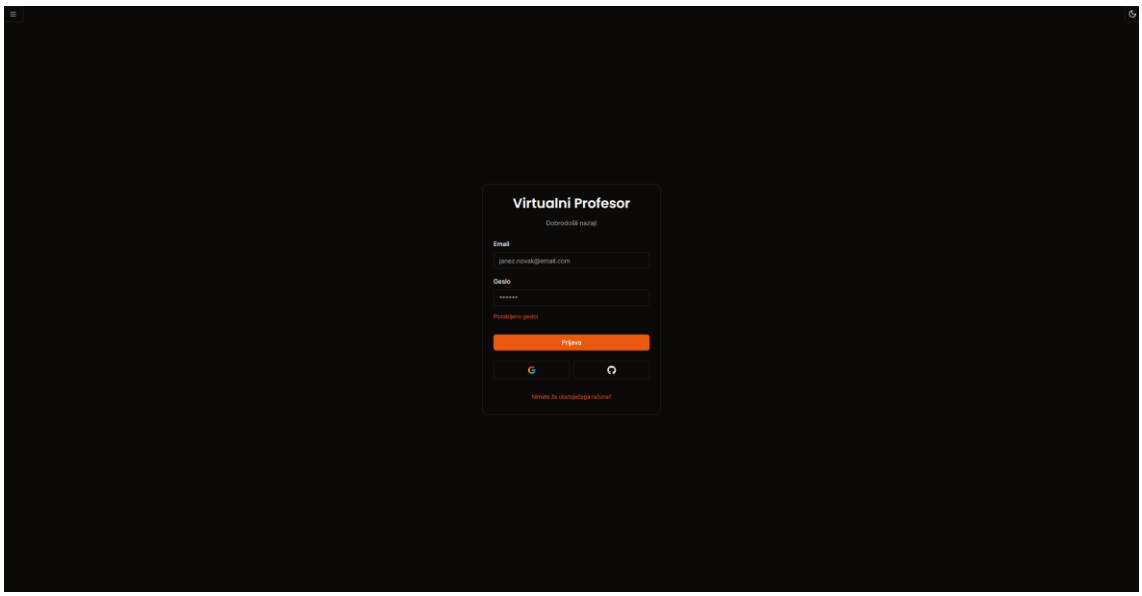
S tem razvojem umetne inteligence v zadnjem času lahko mogoce pričakuje stvari kot so avtonomni roboti, napredni algoritmi in pametna tehnologija ki bo omogočilai inovacije na vseh področjih, od zdravstva do trajnostnega razvoja. Seveda pa je pomembno, da razvijamo te tehnologije odgovorno in etično ter se zavedamo morebitnih izzivov, kot so varnost in pravičnost. Vendar s pravilnimi pristopi lahko oblikujemo svet, kjer bo umetna inteligenca del naše vsakdanje izkušnje in nas podpirala v različnih vidikih življenja. Ampak s tem, ko se veča uporaba teh umetnih inteligenc, so se določeni soočili tudi z bolj ekstremnimi situacijami, kot sta singularnost in transhumanizem. Tehnološka singularnost je točka v prihodnosti, kjer bi tehnološki napredek dosegel to raven, da bi se začel razvijati brez nadzora človeka, kar bi lahko imelo zelo velike posledice v našem življenju. Transhumanizem pa je ideja, v kateri bi poskušali povezati tehnologijo in človeka, kar bi lahko privedlo do nastanka kiborga.

3 PREDSTAVITEV APLIKACIJE

3.1 PRIJAVA

Slika 1: Okno pred prijavo

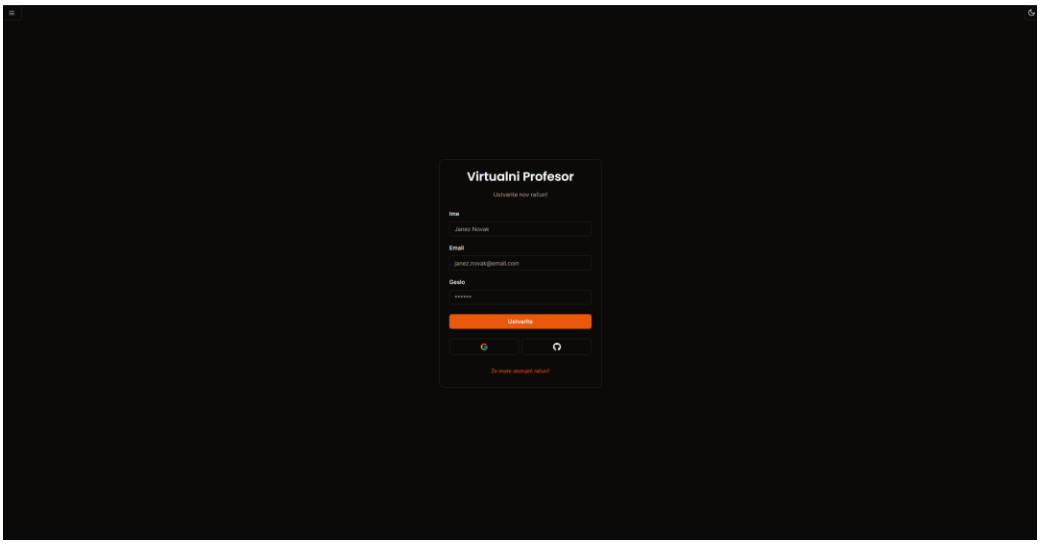
Na sliki je prikazano okno ki se prvo pojavi pred uporabnika ko sprva naloži aplikacijo. S pritiskom na gumb za prijavo se bo uporabniku naložila stran kjer lahko vpiše svoje podatke in se prijavi v samo aplikacijo ali si pa naredi nov račun če ta ni že registriran z enako e-pošto.



Slika 2: Prijavno okno

Na sliki je vidno prijavno okno kjer se uporabnik lahko prijavi preko e-pošte in gesla, ali pa preko Googl-a ter GitHub-a, se registrira ali obnovi geslo. Za delovanje uporablja kodo iz datoteke login.ts.

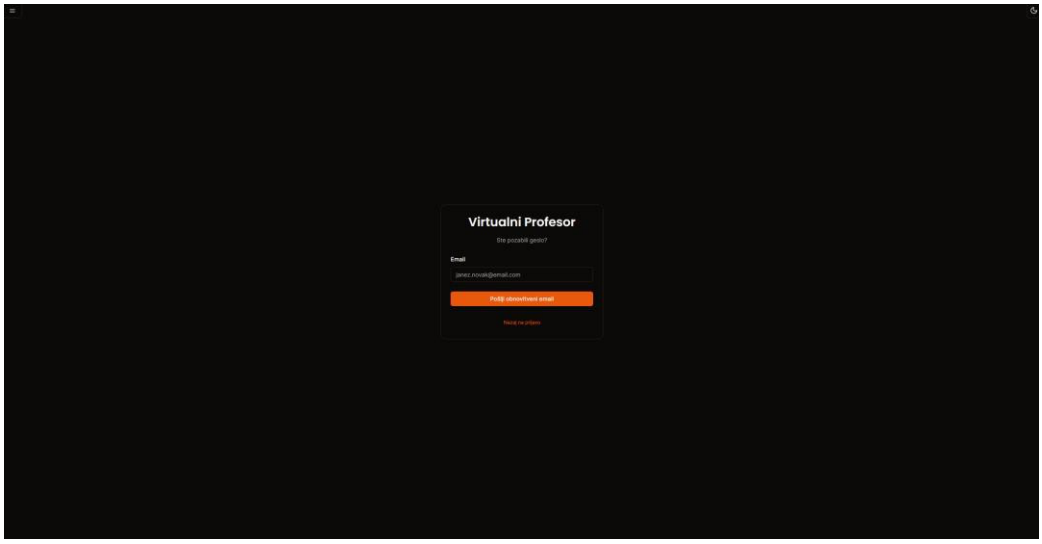
3.2 REGISTRACIJA



Slika 3: Okno za registracijo

Po pritisku “Nimate že obstoječega računa” bo uporabnika aplikacija preusmerila na registrirno stran kjer si lahko naredi nov račun. Omogoča registracijo preko imena, e-pošte in gesla, ali pa preko Googl-a in GitHub-a.

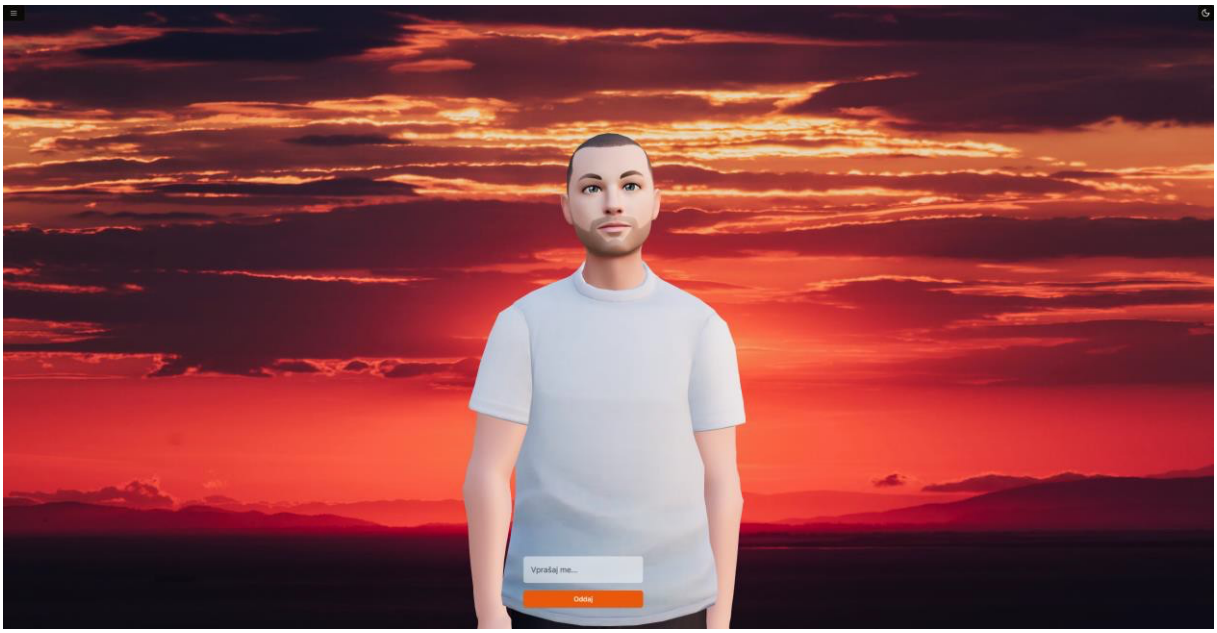
3.3 OBNOVITEV GESLA



Slika 4: Okno za obnovitev gesla

Ko uporabnik klikne na “Pozabljeno geslo?” kjer dokler ima obstoječ račun si lahko pošlje e-pošto za obnovitev gesla. Če vnese napačno e-pošto torej da ni registrirana mu ne bo poslalo obnovitvene pošte.

3.4 VIRTUALNI PROFESOR



Slika 5: Podoba virtualnega profesorja

Po uspešni prijavi uporabnika aplikacija avtomatsko da na to stran kjer je virtualni profesor. Uporabnik vpiše svoje vprašanje v polje in klikne gumb Pošlji. Ko klikne ta gumb se sporočilo pošlje na strežnik, kjer ga naš API obdela in nato vrne rezultat v obliki .mp3 datoteke in sporočila. Potem ko bo njegovo sporočilo sprejel in bo ustrezno odgovoril bo izpisal odgovor ter zvočna datoteka se bo predvajala kjer profesor posnema glas aktualnega profesorja iz šole. To zvočno datoteko si lahko potem uporabnik naloži na svojo napravo in jo potem posluša ko jo potrebuje.

4 KATERE TEHNOLOGIJE SMO UPORABILI?

Z našo aplikacijo smo uporabili kar veliko naprednih in aktualnih metod za izdelavo in objavo spletne strani. Celoten projekt je postavljen na osnovi JavaScripta (v nadaljevanju JS) in TypeScripta (v nadaljevanju TS), ki ju uporablja ogrodje Next.JS. Uporabili smo tudi API (application programming interface) od OpenAI-ja zato da lahko naš program odgovarja na zastavljena vprašanja, pa poleg tega tudi od VoiceLabs-a, ki je uporabljen za posnemanje glasa profesorja ter vračanja zvočne datoteke odgovora.

4.1 JAVASCRIPT

»JavaScript je objektni skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani.« (Wikipedia, JavaScript, 2021)

Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture.

Jezik je nastal neodvisno od Jave, vendar deli številne lastnosti in strukture z njo. Glede na anketo, ki jo je izvedla spletna stran StackOverflow, je JavaScript izjemno razširjen programski jezik. Po tej anketi je JavaScript zasedel prvo mesto med vsemi programskimi jeziki, kar kaže na njegovo naraščajočo priljubljenost v zadnjem času.

4.1.1 RAZLIKA JAVA IN JAVASCRIPT

Posebej pomembna je zadnja poved, ker je pogosta napaka pri omembi JavaScripta, da je zamešana z Javo. Ta poved sicer ni napačna, ne pravilna. Na spodnjem primeru je razvidna razlika.

```
System.out.print("   ");
```

Slika 6: Primer kode v Javi

```
console.log("test")
```

Slika 7: Primer kode v JS

Razvidno je, da je pri Javi potrebno podpičje na koncu stavka, pri JS pa ne. Razlika pa ni samo v sintaksi, temveč tudi v njuni uporabi. Uporabljajo se za različne stvari.

Java je uporabljena za izdelavo igric, programiranje v oblaku, obdelava velike količine podatkov, umetna inteligenca in pa IoT.

JS pa se uporablja predvsem v spletnih aplikacijah v povezavi z HTML programskim jezikom. HTML sam po sebi nima veliko opcij za interaktivnost, zato pa poskrbi JS, ki doda funkcionalnost in interaktivnost.

4.2 TYPESCRIPT

TS je izboljšava JS, ker JS sama po sebi ne preverja tipa podatkov. Kar pomeni, da lahko v primeru narobe nastavljene kode pride tudi do težav, ko določen zunanji model ne sprejema tipa, ki ga je vrnila funkcija. TS to napako odpravi tako, da zahteva, da se pri deklaraciji in inicializaciji poda tip.

```
const stevilka: number = 2;
```

Slika 8: Primer inicializacije ob ddeklaraciji v TS

Če ne podamo tipa ali pa ga trenutno ne poznamo, ni priporočljivo uporabljati tipa ANY, ker s tem onemogočimo celoten pomen TS. Bolj priporočljivo je uporabiti tip UNKNOWN ali pa celo sestaviti svoj tip.

```
type matura = {
    leto: string
    ocena: number
}

const Miha = {
    leto: 2024,
    ocena: 0,
}
```

Slika 9: Primer kreacije svojega tipa matura

```
const Janez: matura = {
    leto: 2024,
    ocena: 0,
}
```

Slika 10: Primer uporabe svojega tipa matura

Kljub temu, da je na sliki 4 že ustvarjen tip matura, ga TS ne prepozna in zato ne javi napake, da je pod spremenljivko leto shranjena številka. Ko pa mi dodamo tip spremenljivke (slika 5) pa TS prepozna napako in jo podčrta. TS je zelo uporaben pri aplikacijah, kjer je potrebno zagotoviti tip podatkov, zato smo ga tudi izbrali za naš projekt.

4.3 NODE.JS

Node.JS je sistem za pogon JS aplikacij na strežniku. Ta sistem je pomemben za razvoj projektov z vsemi JS ogrodji. Deluje asinhrono in lahko sprejme veliko klicev na strežnik. Vključuje tudi paket pod imenom npm, ki je uporabljen za nalaganje odvisnosti in zunanjih knjižnic ter paketov iz spletne strani https://www.npmjs.com. Na tej spletni strani so odprto-kodni paketi, ki jih lahko uporabnik naloži z ukazom »npm i [ime_paketa]«.

4.4 NEXTJS

»Next.js je ogrodje React za gradnjo celovitih spletnih aplikacij. Za gradnjo uporabniških vmesnikov se uporabljajo komponente , za dodatne funkcije in optimizacije pa Next.JS.« (Vercel, 2024)

NextJS je pravzaprav nadgradnja ogrodja React, ker ponuja tudi možnost obdelovanja kode na strežniku.

4.4.1 REACT

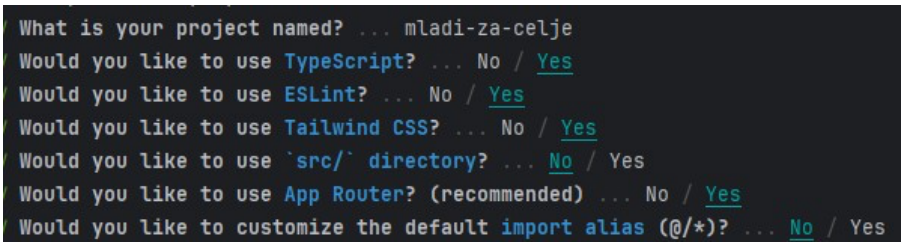
React je ogrodje programskega jezika JS. Ko pišemo kodo HTML ne pomislimo, da bi jo pisali z JS. React to možnost omogoča preko komponent in izvajanja kode na uporabnikovi strani. Tako lahko tudi prikažemo podatke, ki morda niso obstajali, ko smo naložili spletno stran (Dynamic Data Rendering). Odvisno, ali uporabljamo JS ali TS, imajo datoteke različne končnice. Za JS je končnica .jsx, za TS pa .tsx.

```
export default function Domov() : JSX.Element {
    return(
        <main>
            <div>
                Pozdravljeni domov
            </div>
        </main>
    )
}
```

Slika 11: Primer React komponente

4.4.2 ZAGON NEXTJS APLIKACIJE

Preden začetkom programiranja z NextJS, moramo imeti naloženo Node.JS i s tem tudi npm. Ko je to naloženo, lahko v terminal napišemo **»npx create-next-app@latest«.** Ta ukaz bo začel prenos ustreznih odvisnosti za delovanje NextJS. Preden pa se začne prenos pa nam bo terminal postavil še nekaj vprašanj.



Slika 12: Vprašanja terminala

Po odgovoru na vsa vprašanja se bo začel prenos in aplikacija bo delovala. Zaženemo jo z ukazom **»npm run dev«.**

4.4.3 STREŽNIŠKE IN UPORABNIŠKE OPERACIJE V NEXTJS

V ogrodju NextJS, je že po osnovi vsaka komponenta strežniška. Torej ko ustavrimo novo datoteko »primer.tsx«, bo takoj postala strežniška komponenta. Tukaj pride do konfliktov, če NextJS poteka na strežniku, React pa pri uporabniku. Rešitev v NextJS je, da se komponente naložijo na strežniku in se pošljejo k uporabniku, kar pripomore k varnosti, ker se nič ne nalaga pri uporabniku, razen, če je to drugače označeno. Programer lahko označi datoteko z »use client«, kar pomeni, da se bo naložila pri uporabniku. Ko je datoteka označeno z »use client«, lahko dostopamo do vseh funkcij, ki jih ponuja React. Najbolj znani sta useState in useEffect, ki pa delujeta samo pri uporabniku. Z tema funkcijema je omogočen tudi Dynamic Data Rendering pri uporabniku (v nadaljevanju DDR), ker lahko nastavimo stanje z useState in ga nato v kodi primerjamo. Kodo, ki jo bomo naložili preko DDR pri uporabniku označimo z zavitimi oklepaji in vanje zapišemo kodo.

```
export default function Domov() : JSX.Element {
    const [jeOzadje : boolean , setJeOzadje : Dispatch<SetStateAction<boolea... ] = useState( initialState: true)
    return(
        <main>
            {
                jeOzadje && (
                    <div>
                        DA
                    </div>
                )} : {(
                <div>
                    NE
                </div>
                )
            }
        </main>
    )
}
```

Slika 13: Primer DDR

Na sliki 8 je prikazan DDR z pomočjo kljuke useState, ki je na začetku nastavljena na true. Glede na njeno vrednost, se bo pri uporabniku izpisalo DA ali NE. Nadgradnja je, da se podatki ddobijo preko Aplication Programing Interface (v nadaljevanju API) ali pa iz podatkovne baze in se potem dinamično naložijo na spletno stran.

4.4.4 DELOVANJE POTI V NEXTJS

NextJS deluje na principu map in datotek v mapah. Vsaka mapa v mapi »app« ali »src/app«, bo postala pot v URL naslovu. Na primer, če ustvarimo mapo **»prijava«**, bo potem URL bil **»/prijava«.** Potrebno pa je tudi ustvariti datoteko **»page.tsx«** znotraj mape, kjer prikažemo to kar želimo. Če nočemo da mapa postane pot v URL naslovu, jo predznačimo z »_«. Ta mapa in vse njene podmape ne bodo postale poti za URL naslov. Je pa tudi možnost poimenovanja mape, da služi samo organizacijskem namenu. Označimo jo z »()« in ta mapa ne bo postala pot, vendar vse njene podmape pa. Primer, če imamo map **»(stran)«** in v njej mapo »prijava«, bo URL naslov samo »/prijava«, ker »(stran)« služi samo organizacijskem namenu.

4.5 AUTH.JS

Auth.js je samostojna knjižnica ali modul, ki omogoča preverjanje identitete uporabnikov preko različnih metod, kot so e-poštni naslov in geslo, družabna omrežja, enkratne povezave itd. Implementacija avtentikacije vključuje preverjanje pravilnosti uporabniških podatkov in potrjevanje identitete uporabnika, preden mu je omogočen dostop do aplikacije.

Poleg avtentikacije Auth.js omogoča tudi upravljanje s pravicami dostopa uporabnikov. To vključuje določanje vlog in pravic, ki določajo, katere funkcije in vsebine lahko uporabnik dostopa, ko je prijavljen v aplikacijo.

Auth.js omogoča tudi dinamično dodeljevanje pravic na podlagi različnih dejavnikov, kot so uporabniške vloge, statusi in dovoljenja, kar vpliva na sam dostop posameznika do aplikacije.

Njen glavni namen je zagotoviti varnost in zaščito občutljivih podatkov ter omejiti dostop do določenih funkcij le na prijavljene uporabnike. To vključuje ustrezno šifriranje gesel in uporabo varnih metod za shranjevanje ter prenos podatkov med strežnikom in odjemalcem.

Implementacija Auth.js vključuje uporabo različnih metod in tehnik za preverjanje identitete uporabnikov ter upravljanje s sejami in dostopnimi pravicami za lahek ter varen dostop do same spletne aplikacije VirtualniProfesor.

4.5.1 ACTIONS

4.5.1.1 LOGIN.JS

Datoteka login.js vsebuje funkcijo login, ki je odgovorna za proces prijave uporabnika v aplikacijo. Preverja, ali so podatki za prijavo (e-pošta in geslo) veljavni. Če uporabnik že obstaja v podatkovni bazi, preveri, ali je njegov e-poštni naslov že potrjen. Če e-poštni naslov ni potrjen, funkcija pošlje e-poštno sporočilo za potrditev. Če so podatki za prijavo pravilni, uporabi funkcijo signIn za prijavo uporabnika. Če pride do napake, funkcija vrne ustrezno sporočilo o napaki. newVerification.js: Datoteka newVerification.js vsebuje funkcijo newVerification, ki je odgovorna za proces potrditve e-poštnega naslova. Preverja, ali obstaja veljaven potrditveni žeton s podanim tokenom. Preveri, ali je potrditveni žeton potekel. Če je potrditveni žeton veljaven, posodobi podatke o uporabniku v podatkovni bazi in označi e-poštni naslov kot potrjen. Nato izbriše uporabljen potrditveni žeton iz podatkovne baze.

4.5.1.2 REGISTER.JS

Datoteka register.js vsebuje funkcijo register, ki je odgovorna za proces registracije novega uporabnika v aplikaciji. Preveri, ali so podatki za registracijo (e-pošta, geslo, ime) veljavni. Preveri, ali je e-poštni naslov že v uporabi. Če e-poštni naslov ni v uporabi, ustvari nov uporabniški račun v podatkovni bazi. Po uspešni registraciji pošlje e-poštno sporočilo za potrditev registracije.

4.5.1.3 RESET.JS

Datoteka reset.js vsebuje funkcijo reset, ki je odgovorna za proces ponastavitve gesla. Preveri, ali je podani e-poštni naslov veljaven. Če je e-poštni naslov veljaven, pošlje e-poštno sporočilo za ponastavitev gesla.

4.5.2 DATA

4.5.2.1 PASSWORD-RESET-TOKEN.TS

Datoteka password-reset-token.ts vsebuje funkcijo getPasswordResetTokenByToken, ki je odgovorna za pridobitev podatkov o ponastavitvenem žetonu gesla glede na podani žeton. Funkcija poskuša poiskati podatke o ponastavitvenem žetonu v podatkovni bazi in jih vrne. Če želeni žeton ne obstaja v podatkovni bazi, funkcija vrne null.

4.5.2.2 USER.TS

Datoteka user.ts vsebuje dve funkciji: »getUserByEmail« in »getUserById«. »getUserByEmail« je odgovorna za pridobitev podatkov o uporabniku glede na podani e-poštni naslov. »getUserById« je odgovorna za pridobitev podatkov o uporabniku glede na podani ID. Obe funkciji poskušata poiskati podatke o uporabniku v podatkovni bazi in jih vrneta. Če uporabnik z določenim e-poštnim naslovom ali ID-jem ne obstaja v podatkovni bazi, funkciji vrneta null.

4.5.2.3 VERIFICATION-TOKEN.TS

Datoteka verification-token.ts vsebuje dve funkciji: getVerificationTokenByEmail in getVerificationTokenByToken. getVerificationTokenByEmail je odgovorna za pridobitev podatkov o potrditvenem žetonu glede na podani e-poštni naslov. getVerificationTokenByToken je odgovorna za pridobitev podatkov o potrditvenem žetonu glede na podani žeton. Obe funkciji poskušata poiskati podatke o potrditvenem žetonu v podatkovni bazi in jih vrneta. Če potrditveni žeton za določeni e-poštni naslov ali žeton ne obstaja v podatkovni bazi, funkciji vrneta null.

4.5.3 PRISMA:

Prisma schema določa strukturo podatkovne baze MongoDB in relacije med podatkovnimi modeli. V tej shemi so definirani naslednji modeli:

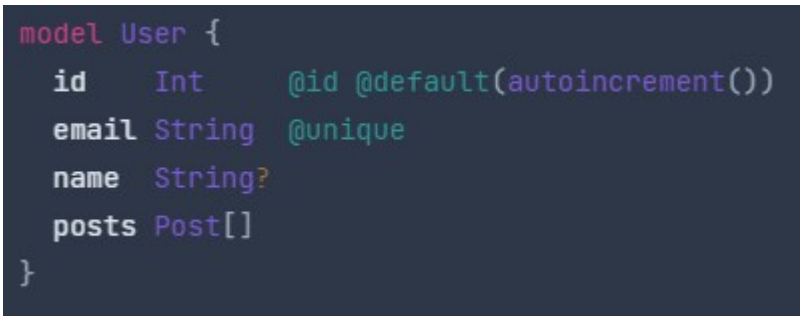
User: Vključuje podatke o uporabnikih, kot so njihov ID, ime, e-poštni naslov, čas potrditve e-pošte, slika, geslo in vloga. Vsak uporabnik ima lahko več povezanih računov.

Account: Predstavlja povezavo med uporabnikom in zunanjim računom (npr. Google, Facebook). Vsebuje podatke o ID-ju računa, uporabnikovem ID-ju, vrsti računa, ponudniku, ID-ju ponudnika in drugih atributih.

VerificationToken: Uporablja se za potrditev e-poštnega naslova. Vključuje podatke o ID-ju žetona, e-poštnem naslovu, samem žetonu in datumu poteka.

PasswordResetToken: Uporablja se za ponastavitev gesla. Vsebuje podatke o ID-ju žetona, e-poštnem naslovu, samem žetonu in datumu poteka.

History: Uporablja se za beleženje zgodovine dejanj ali aktivnosti v aplikaciji. Vključuje podatke o ID-ju, e-poštnem naslovu, sporočilu, odgovoru umetne inteligence in UUID-ju zvočne datoteke. Te definicije omogočajo strukturiranje in organiziranje podatkov v MongoDB bazi ter vzpostavljanje relacij med različnimi vrstami podatkov, kar omogoča učinkovito upravljanje in dostop do informacij v aplikaciji.



Slika 14: Primer modela (vir slike: https://www.prisma.io/docs/getting-started/quickstart, uporabljeno: 10.3.2024)

4.5.4 PUBLIC

V public mapi imamo podmape, ki shranjujejo različne dokumente za našo spletno stran. V podmapi animations so shranjene animacije našega avatarja. V podmapi images so shranjene slike ki se prikazujejo na spletni strani. Ena slika je za ozadje profeosrja, ena pa za ikono v zavihku.

4.5.5 SCHEMAS(INDEX.TS):

Ta datoteka vsebuje definicije shem za preverjanje veljavnosti podatkov v treh različnih kontekstih: ponastavitvi gesla, prijavi in registraciji. Spodaj jih bom opisal poimensko:

ResetSchema: ResetSchema je shema za preverjanje podatkov, ko uporabnik zahteva ponastavitev gesla. Vsebuje eno polje email, ki ga je treba preveriti. E-poštni naslov mora biti veljaven.

LoginSchema: LoginSchema je shema za preverjanje podatkov, ko uporabnik poskuša prijaviti v aplikacijo. Vsebuje dve polji, email in password. E-poštni naslov mora biti veljaven, geslo pa mora vsebovati vsaj en znak.

RegisterSchema: RegisterSchema je shema za preverjanje podatkov, ko uporabnik poskuša registrirati nov račun. Vsebuje tri polja, email, password in name. E-poštni naslov mora biti veljaven, geslo pa mora biti dolgo najmanj 6 znakov, ime pa mora vsebovati vsaj en znak.

Vsaka shema uporablja modul zod, ki omogoča definiranje različnih pravil za vsako polje, kot so minimalna dolžina, obveznost, veljavnost e-poštnega naslova itd. Te sheme so namenjene zagotavljanju skladnosti in varnosti podatkov, ki jih uporabniki vnašajo v aplikacijo.

4.5.6 SRC/APP

Mapa app vsebuje več podmap in datotek. Skupaj te datoteke in mape tvorijo celotno funkcionalnost avtentikacije v aplikaciji. Uporabnikom omogočajo dostop do različnih funkcij avtentikacije, kot so prijava, registracija, ponastavitev gesla, in zagotavljajo ustrezno uporabniško izkušnjo med temi postopki. Opis datotek:

api/route.ts: Ta datoteka omogoča definiranje poti za API. To lahko vključuje poti za prijavo, registracijo, ponastavitev gesla, itd.

auth/error/page.tsx: Ta datoteka predstavlja stran za prikaz napak, ki se lahko pojavijo med postopkom avtentikacije. Uporabniku zagotavlja obvestilo o napaki in morda navodila za nadaljnje korake.

auth/login/page.tsx: Stran za prijavo uporabnika. Vsebuje obrazec za prijavo, kjer uporabnik vnese svoje poverilnice in se prijavi v aplikacijo.

auth/new-verification/page.tsx: Stran za novo potrditev e-poštnega naslova. Uporabniku omogoča, da zahteva novo potrditveno e-pošto, če prejšnja ni bila prejeta ali potrjena.

auth/register/page.tsx: Stran za registracijo novega uporabnika. Vsebuje obrazec za registracijo, kjer uporabnik vnese svoje podatke in ustvari nov račun.

auth/reset/page.tsx: Stran za ponastavitev gesla. Uporabniku omogoča, da zahteva ponastavitev gesla, če je pozabil svoje trenutno geslo.

layout.tsx: Definira osnovno postavitev (layout) avtentikacijskih strani. Določa, kako bodo komponente na teh straneh postavljene in stilizirane.

4.5.7 SRC/COMPONENTS

Datoteke v mapi Components vsebujejo več komponent, ki se uporabljajo za gradnjo različnih delov uporabniškega vmesnika, povezanih z avtentikacijo in drugimi elementi v aplikaciji. Opis datotek:

back-button.tsx: Komponenta za gumb za vračanje nazaj, ki omogoča prehod nazaj na prejšnjo stran.

Card-wrapper.tsx: Visoko nivojska komponenta, ki ovije vsebino v kartico s prikazom glave, noge in možnosti vračanja nazaj.

error-card.tsx: Komponenta za prikaz napak pri avtentikaciji, ki vsebuje sporočilo o napaki in možnost vračanja na prejšnjo stran.

header.tsx: Glava komponente, ki prikazuje ime aplikacije in določeno besedilo.

login-button.tsx: Gumb za preusmeritev na stran za prijavo.

login-form.tsx: Obrazec za prijavo uporabnika, vključno z e-poštnim naslovom in geslom.

new-verification-form.tsx: Obrazec za potrditev novega e-poštnega naslova.

register-form.tsx: Obrazec za registracijo novega uporabnika.

reset-form.tsx: Obrazec za ponastavitev gesla.

Social.tsx: Komponenta, ki omogoča prijavo preko družbenih omrežij.

avatar.jsx: Ta komponenta uporablja React in funkcionalne komponente. Renderira 3D avatar v WebGL okolju s pomočjo knjižnice @react-three/drei.

AvatarScene.tsx: Ta komponenta uporablja Suspense za zagotovitev, da se vsebina renderira, ko je pripravljena. Vsebuje <Canvas> komponento, ki predstavlja WebGL okolje za renderiranje 3D vsebine.

Chat.tsx: Ta komponenta uporablja React in funkcionalne komponente. Omogoča uporabniku pošiljanje sporočil in pretvorbo besedila v govor s pomočjo zunanjega API-ja.

navbar.tsx: Ta komponenta uporablja ikone iz knjižnice lucide-react in @radix-ui/react-icons. Omogoča uporabniku navigacijo po različnih straneh aplikacije prek padajočega menija.

button.tsx: To je generična komponenta gumba, ki se lahko prilagodi različnim variantam in velikostim. Uporablja knjižnico class-variance-authority za dinamično nastavljanje različnih stilov glede na podane variante.

card.tsx: To so generične komponente za prikazovanje kartic v uporabniškem vmesniku.

dropdown-menu.tsx: To so komponente, ki omogočajo prikaz padajočega menija v uporabniškem vmesniku. Vključujejo sprožilec, vsebino, elemente menija, ločilo, skupine elementov itd.

form.tsx: To so komponente za oblikovanje obrazca, ki olajšujejo uporabo skupaj z react-hook-form knjižnico.

input.tsx: Gre za generično komponento za vnos besedila v obrazce. Ponuja različne nastavitve, kot so velikost, obrobje, sence in druge.

label.tsx: Generična komponenta za oznake besedila v obrazcih. Omogoča prilagodljive stile glede na podane variante.

FormError.tsx in FormSuccess.tsx: To sta komponenti, ki prikazujeta sporočila o napaki ali uspehu v obrazcu. Uporabljata se za vizualno povratno informacijo o stanju pošiljanja obrazca.

4.5.8 SRC/INTERFACE

Datoteka interfaces.ts vsebuje definicijo vmesnika (interface) ChatGPTProps, ki opisuje lastnosti (props) za komponento ali funkcijo, ki uporablja ta vmesnik. To omogoča boljšo berljivost, vzdrževanje in preprečuje napake med razvojem.

4.5.9 SRC/LIB

Mapa lib vsebuje več ključnih datotek, ki so bistvene za delovanje aplikacije. Vse te datoteke skupaj zagotavljajo ključne funkcionalnosti, kot so povezava s podatkovno bazo, pošiljanje e-pošte, generiranje in upravljanje žetonov ter upravljanje stilov v aplikaciji. Opis datotek:

db.ts: Ta datoteka inicializira povezavo s podatkovno bazo prek Prisma ORM. Uvozi PrismaClient in ustvari globalno spremenljivko prisma, ki omogoča dostop do podatkovne baze po vsej aplikaciji. Če je prisma že inicializiran, se uporabi obstoječa instanca, sicer se ustvari nova. Prav tako preveri, ali je aplikacija v produkcijskem okolju, in če ni, nastavi prisma kot globalno spremenljivko.

mail.ts: Ta datoteka vsebuje funkcijo za pošiljanje e-poštnih sporočil prek knjižnice emailjs/browser. Funkcija sendEmail pošlje e-pošto s povezavo za potrditev e-poštnega naslova.

token.ts: Ta datoteka vsebuje funkcijo za generiranje edinstvenih žetonov za potrditev e-poštnega naslova. Uporablja knjižnico uuid za generiranje žetonov in knjižnico PrismaClient za upravljanje podatkov v zvezi s temi žetoni.

utils.ts: Ta datoteka vsebuje funkcijo za upravljanje razredov za oblikovanje v React aplikaciji. Funkcija cn združuje razrede za stile z uporabo knjižnice clsx in tailwind-merge.

4.5.10 SRC/PROVIDERS

Datoteka **ChatGPT_api.tsx** v mapi providers vsebuje funkcije za uporabo storitve OpenAI za klepet. Opis datoteke:

Uvoz knjižnice OpenAI in definicija objekta openai, ki ga inicializiramo za uporabo storitve OpenAI. Funkcija CHAT omogoča klepet s storitvijo OpenAI. Uporablja openai.chat.completions.create za dokončanje sporočila na podlagi vnesenih podatkov, kot je vloga (npr. sistem ali uporabnik) in model, ki ga želimo uporabiti.

Datoteka **provider.tsx** v mapi providers vsebuje komponento Providers, ki zagotavlja osnovne ponudnike za aplikacijo. Opis datoteke:

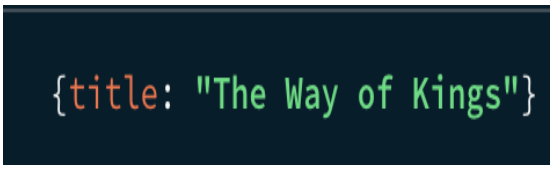
Uvoz NextUIProvider iz knjižnice @nextui-org/react. Definicija komponente Providers, ki sprejme children kot svoj edini parameter. Ta komponenta ovije aplikacijo z NextUIProvider, ki zagotavlja komponente za uporabniški vmesnik, kot so gumbi, obrazci, itd.

4.6 MONGODB

MongoDB je podatkovna baza, ki ne uporablja jezika SQL, torej ne shranjuje podatkov v tabele ampak v dokumente tipo BSON (angl. Binary JSON). Deluje na principu parov ključ in vrednost Na primer, da želimo shraniti uporabniško ime »Metka123« bi ga shranili na način **»uporabnisko_ime« : »Metka123«**. Za iskanje se uporabljajo filtri in ne ukazi kot v SQL. Na prejšnem primeru bi filter izgledal takole: »{uporabnisko_ime: »Metka123«}. Ta filter bi nam vrnil zgornji zapis uporabniškega imena v podatkovni bazi. Vsak dokument ima lahko poljubno število parov, vendar je omejitev velikosti dokumenta.

MongoDB ima omejitev 16 MB na dokument, vendar ima možnost nalaganja večjih datotek, z pomočjo GridFS. To je že vgrajena funkcija, ki datoteko razbije na več manjših kosov (angl. chunks) . Ti kosi so indeksirani po vrsti in shranjeni vsak posebej v dokument. Vsak kos ima velikost 255 kB, razen zadnji, ki je odvisen od količine podatkov.

Slika 15: Primer dokumenta v MongoDB



Slika 16: Primer filtra v MongoDB

4.7 THREE.JS

Three.JS je knjižnica datotek za prikaz 3D modelov in animacijo le-teh v spletnem brskalniku. Koristna knjižnica, ki poenostavi delovanje z WebGL in JS API.

4.7.1 NAŠA UPORABA

Three.JS je uporabljen za prikaz 3D modela profesorja in njegovih animacij.

4.8 NEST.JS

»Nest (NestJS) je ogrodje za gradnjo učinkovitih in razširljivih aplikacij na strani strežnika Node.js. Uporablja napredni JavaScript, zgrajeno je s pomočjo in v celoti podpira TypeScript (vendar razvijalcem še vedno omogoča kodiranje v čistem jeziku JavaScript) ter združuje elemente OOP (objektno usmerjeno programiranje), FP (funkcionalno programiranje) in FRP (funkcionalno reaktivno programiranje).« (NestJs, 2017) (Prevedeno z pomočjo DeepL, https://www.deepl.com/en/translator)

Za delovanje so potrebni 3 tipi datotek. Krmilnik, modul in pa storitev. Vsaka izmed teh treh datotek skrbi za svoj del API.

4.8.1 KRMILNIK

Krmilnik sprejema https zahteve in izvede kodo v njih. Vsak razred v tej datoteki lahko postane krmilnik, samo potrebno je dodati okrasitev **»@Controller()«.**

```
1+ usages
@Controller()
export class AppController {
  no usages
  constructor(private readonly appService: AppService) {}

  1+ usages
  @Get()
  getHello(): string {
    return this.appService.getHello();
  }
}
```

Slika 17: Primer Krmilnika v Nest.JS

Na sliki je opazna nova okrasitev **»@Get()«.** Z to okrasitvijo povemo krmilniku, da naj pričakuje GET zahtevo. V oklepaje lahko podamo posebej pot in tako naredimo svoje API poti v URL naslovu. Če so oklepaji prazni ne bo posebej poti in se bo zahteva izvedla takoj ko pridemo na URL naslov, kjer teče naš API. Obstajajo pa še tudi druge okrasitve, kot so »**POST, PATCH, DELETE, …«**

```
@Get( path: "/hello")
```

Slika 18: Primer posebne poti API

4.8.2 MODULI

Moduli so organizacijske strukture v Nest.JS, ki vračajo meta podatke.

```
@Module( metadata: {
  imports: [OpenAiApiModule, ConfigModule.forRoot(), ElevenlabsApiModule],
  controllers: [AppController, FilesController],
  providers: [AppService],
})
export class AppModule {}
```

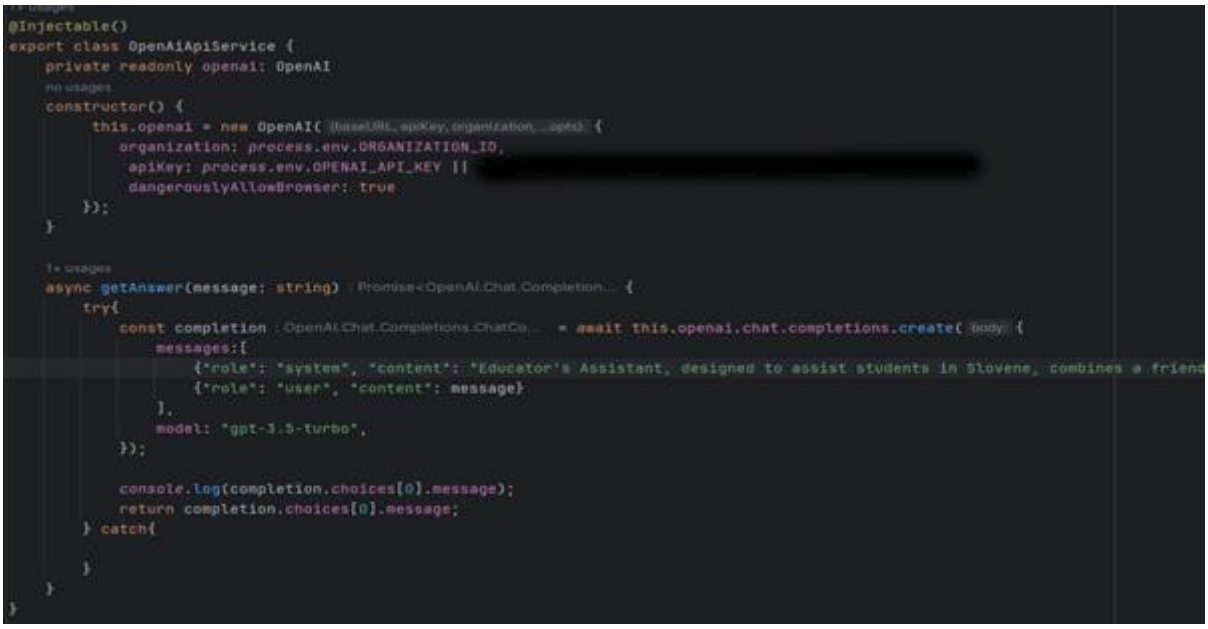
Slika 19: Primer modula

4.8.3 STORITEV

V datoteki storitev se nahaja večina kode, ki jo uporablja API. V njo zapišemo vse funkcije in razrede, ki jih bomo potem uporabili v krmilniku.

Slika 20: Primer storitve

»@Injectable« okrasitev se uporablja, da lahko »vbrizgamo« kodo v druge razrede. Predvsem uporabno je za preverjanje, če je uporabnik prijavljen ali ne.



Slika 21: OpenAi Storitev

4.8.4 NAŠA UPORABA

Nest.JS smo uporabili za povezavo uporabnikove strani aplikacije z strežnikom, kjer se izvedejo klici na OpenAI API in na ElevenLabs. Ko uporabnik vpiše svoje vprašanje se preko v naprej definiranega URL-ja in poti v URL naslovu. Uporabili smo **»/open-ai-api/getAnswer«**, **»/text-to-speech/create«** in **»/text-to-speech/files/:uuid«.** Prva pot skrbi za podajanje in pridobivanje odgovora od ChatGPT-ja in vračanje oddgovora uporabniku na spletni strani. Ta odgovor se nato izpiše uporabniku in se pošlje na drugo pot kjer se ustvari še zvočna datoteka. Ta se predvaja simultano z odgovorom na spletni strani. Zadnja pot se uporablja za pridobivanje datoteke iz podatkovne baze.

```
@Controller( prefix: 'text-to-speech')
export class ElevenlabsApiController {
    no usages
    constructor(private readonly textToSpeechService: ElevenlabsApiService) {}

    no usages
    @Post( path: 'create')
async convertTextToSpeech(@Body() data: getMessageObject): Promise<any> {
        try {
            const {uuid : string , message : string } = await this.textToSpeechService.convertTextToSpeech(data.message);
            return {uuid, message}
        } catch (error) {
            throw new Error('Failed to convert text to speech');
        }
    }
}
```

Slika 22: Krmilnik ElevenLabs

4.9 SHADCN-UI

ShadCn-ui je namenjen lažjem sestavljanju uporabniškega vmesnika, ker ponuja že v naprej narejene komponente, ki jih lahko potem oblikuješ po svojem okusu. Ima tudi že vgrajeno menjavo tem iz svetlega v temni način

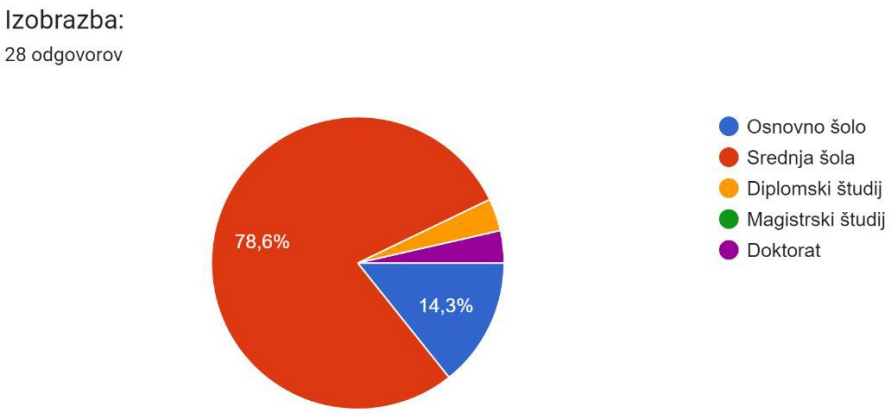
5 ANALIZA ANKETE

Ustvarili smo anketo, s katero bi lahko pridobili o učenju med dijaki ter učenci pridobili smo 28 odzivov na našo anketo. Spodaj so vprašanja ki si sledijo po vrstnem redu ter imajo zapisano zakaj smo te podatke probali pridobiti.

1. **Izobrazba anketirancev**

   Želeli smo izvedat kako ima izobrazba anketirancev povezavo z njihovo sposobnostjo z uporabo različnih učnih načinov.

   Iz vseh odgovorov ko smo dobili lahko vidimo da smo dobili največ odgovorov iz srednje šole.

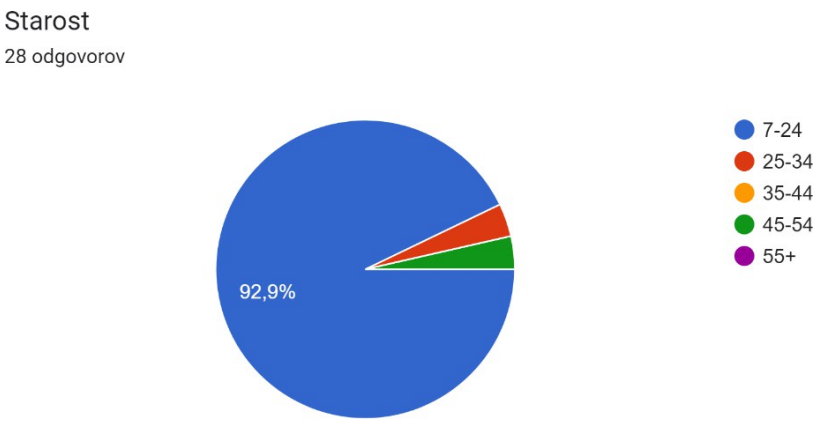


Slika 23: Vprašanje 1

2. Starost anketirancev

S tem vprašanjem smo pa želeli izvedati kako ima starost naših anketirancev vpliv na uporabo različih sredstev saj so lahko mlajše generacije bolj naklonjene k učenjem s AI tehnologijo.

Spodaj vidimo da smo dobili največ odgovorov is starosti 7-24 let

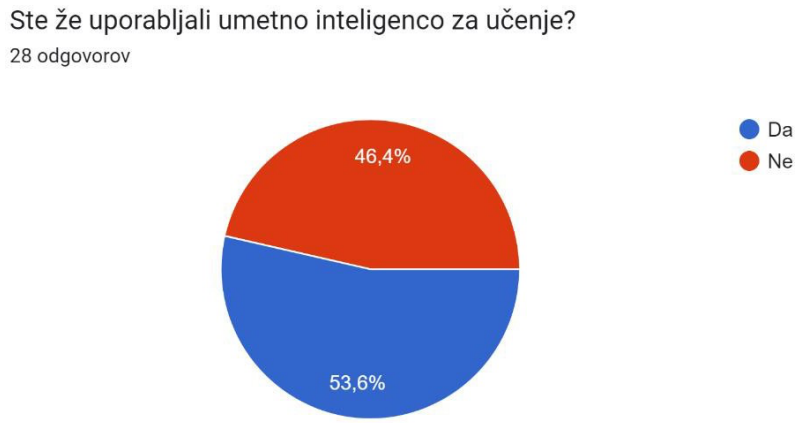


Slika 24: Vprašanje 2

3. Ste že uporabljali umetno inteligenco za učenje?

To vprašanje omogoča razumevanje obstoječe izkušnje anketirancev z uporabo AI pri učenju in kako to vpliva na njihovo mnenje o tej metodi.

Spodaj lahko vidimo, da je približno 54% odgovorov nakazovalo, da se uporablja umetna inteligenca pri učenju.

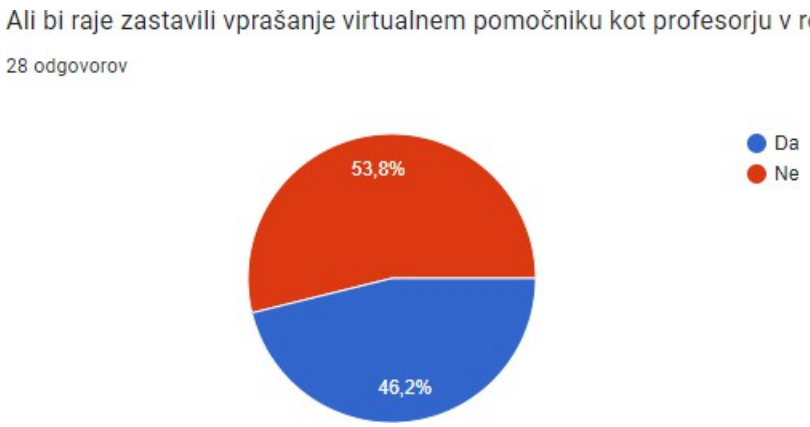


Slika 25: Vprašanje 3

4. **Ali bi raje zastavili vprašanje virtualnem pomočniku kot profesorju v realnem času?**

   To vprašanje nam je pomagalo ugotoviti ali lahko potrdimo prvo hipotezo ali ne.

   Po odzivih lahko vidimo da je več osebkov ki bi želeli uporabiti virtualnega pomočnika kot ne.



Slika 26: Vprašanje 4

5. **Ali imate motnje pri sledenju pouka ali učenja?**

   To vprašanje nam je pa pomagalo pri ugotovitvi druge hipoteze.

   Sklepajoči po odzivih lahko ugotovimo da večina anketirancev nima učnih motenj ali motenj pri sledenju pouka.

Slika 27: Vprašanje 5

6. **Ali se pogosto srečujete z vprašanji, na katera ne znate odgovoriti med učenjem?**

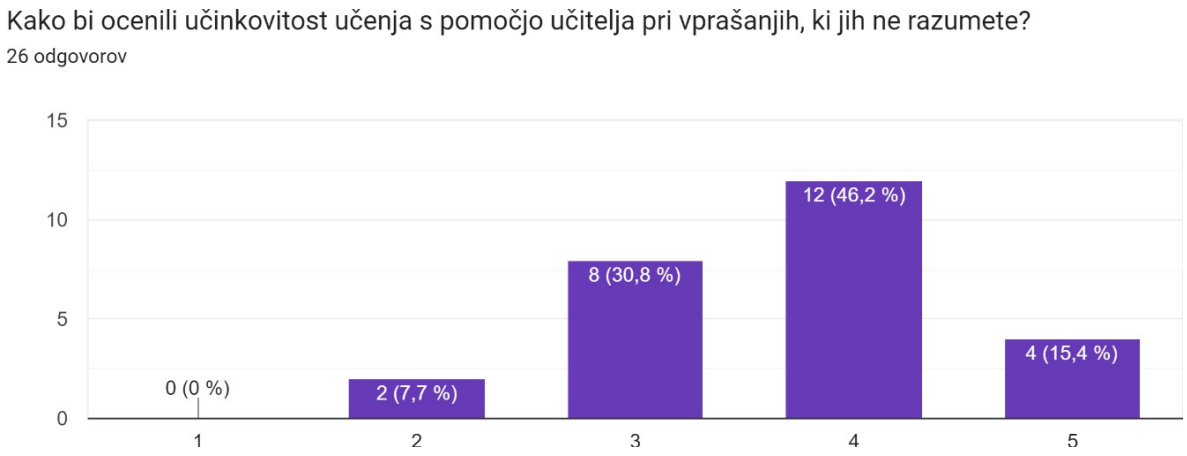
   To vprašanje pomaga ugotoviti, kako pogosto se anketiranci srečujejo z vprašanji ki jih ne razumejo pri učenju

   Kot vidite po odzivih, se v večini primerov pojavljajo vprašanja, na katera ne znajo odgovoriti pri vsakem učencu do določene mere.

Slika 28: Vprašanje 6

7. **Kako bi ocenili učinkovitost učenja s pomočjo učitelja pri vprašanjih, ki jih ne razumete?**

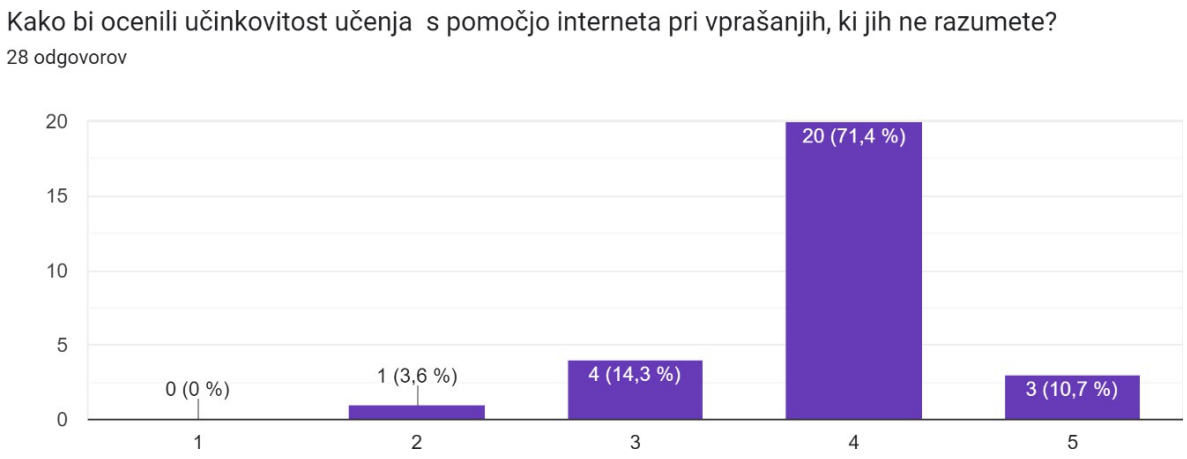
   Prikaže učinkovitost učenja s pomočjo tradicionalnih učiteljev da lahko potem s nasljednjemi vprašanji ugotovimo katera je najbolj priljubljena metoda učenja.



Slika 29: Vprašanje 7

8. **Kako bi ocenili učinkovitost učenja s pomočjo interneta pri vprašanjih, ki jih ne razumete?**

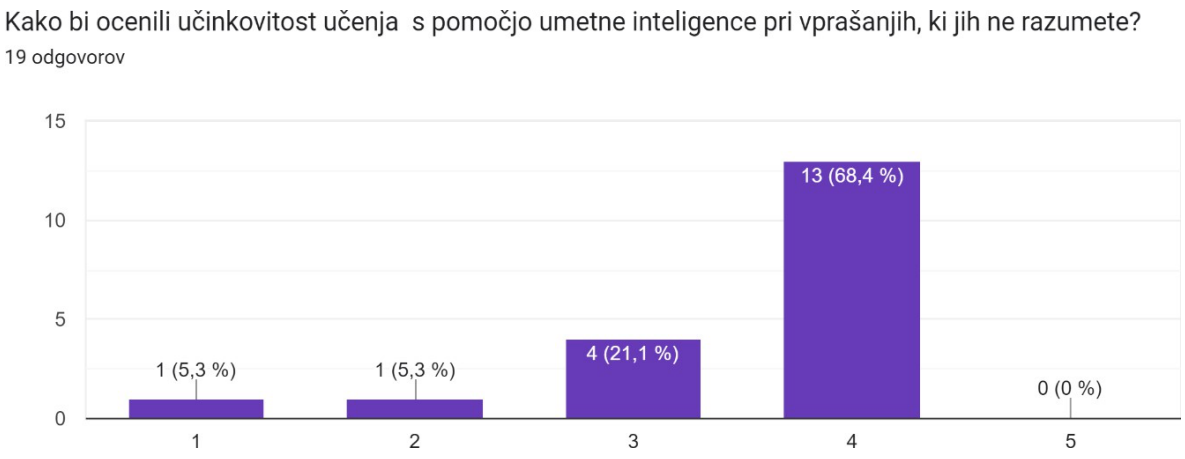
   Prikaže učinkovitost učenja s pomočjo internetnih sredstev da lahko potem ugotovimo najboljšo metodo učenja.



Slika 30: Vprašanje 8

9. **Kako bi ocenili učinkovitost učenja s pomočjo umetne inteligence pri vprašanjih, ki jih ne razumete?**

   Prikaže učinkovitost učenja s pomočjo internetnih sredstev da lahko potem ugotovimo najboljšo metodo učenja.



Slika 31: Vprašanje 9

10. **Kakršni koli dodatni komentarji ali povratne informacije, ki jih želite dodati?**

    To odprto vprašanje omogoča anketirancem, da delijo svoje dodatne poglede, izkušnje in mnenja v zvezi z uporabo umetne inteligence pri učenju

6 UGOTOVITVE

Prva hipoteza, ki je predvidevala, da bodo dijaki verjetneje uporabljali virtualnega pomočnika kot pa se odločili za neposredno vprašanje profesorja v realnem času, je bila ovrgla. Naša raziskava ni podprla te hipoteze, saj rezultati niso pokazali statistično pomembne razlike v pogostosti uporabe med virtualnim pomočnikom in neposrednim vprašanjem profesorju.

Druga hipoteza, ki je predvidevala, da se bo uporaba našega orodja izkazala kot potencialna rešitev za dijake z učnimi težavami, ni bila zadostno podprta zaradi omejenega števila anketirancev. Kljub temu so rezultati nakazali na možnost učinkovite rešitve za dijake s težavami, vendar je potrebno večje število anketirancev za trdnejše zaključke.

Tretja hipoteza, ki je predvidevala, da bo virtualni profesor verjetno naredil napako v približno 10% primerov, je bila potrjena na podlagi ugotovitev analize podatkov. Virtualnem profesorju smo zadali dvajset nalog. Te naloge je opravil z 80 odstotnim pravilnim odgovorom kar potrjuje našo hipotezo.

Rezultati so pokazali, da je bila stopnja napak virtualnega profesorja v skladu z napovedmi, kar kaže na zanesljivost sistema v večini primerov.

	A	B	C	D	E	F	G	H
1	Vprašanje - 1		Vprašanje - 6		Vprašanje - 11		Vprašanje - 16	
2	Vprašanje - 2		Vprašanje - 7		Vprašanje - 12		Vprašanje - 17	
3	Vprašanje - 3		Vprašanje - 8		Vprašanje - 13		Vprašanje - 18	
4	Vprašanje - 4		Vprašanje - 9		Vprašanje - 14		Vprašanje - 19	
5	Vprašanje - 5		Vprašanje - 10		Vprašanje - 15		Vprašanje - 20	

Slika 32: Analiza druge hipoteze

7 ZAKLJUČEK

Na koncu je raziskovalni projekt izpostavil hipoteze o uporabi virtualnih asistentov v izobraževalnih okoljih, ki so si prizadevale razumeti uporabniške preference in potencialne koristi za učence z učnimi težavami. Metodologija je vključevala celovit pregled obstoječe literature, osebne izkušnje in mnenja, da bi raziskala posledice umetne inteligence in avatarjev pri izboljšanju učnih procesov.

Naključno gledano ponuja ta raziskava dragocene vpoglede v vlogo asistentov, ki jih poganja umetna inteligenca, v izobraževanju, ponuja pa tudi temelj za prihodnje raziskovanje.

8 BIBLIOGRAFIJA

AWS. (brez datuma). *what-is-java*. Pridobljeno 12. Marec 2024 iz AWS: https://aws.amazon.com/what-is/java/#:~:text=Java%20is%20a%20multi-platform,applications%20and%20server-side%20technologies.

NestJs. (2017). *Documentation*. Pridobljeno 10. Marec 2024 iz NestJS: https://docs.nestjs.com/first-steps

NPM. (2024). Pridobljeno 11. Mesec 2024 iz NPM: https://www.npmjs.com

Pocock, M. (5. December 2022). *TypeScript: Should you use Types or Interfaces?* Pridobljeno 11. Marec 2024 iz Youtube: https://www.youtube.com/watch?v=zM9UPcIyyhQ&t=127s&ab_channel=MattPocock

Prisma. (2024). *ORM*. Pridobljeno 11. Marec 2024 iz Prisma: https://www.prisma.io/orm

React. (2024). *Learn*. Pridobljeno 11. Marec 2024 iz React: https://react.dev/learn/start-a-new-react-project

StackOverflow. (2023). *2023*. Pridobljeno 11. Marec 2024 iz StackOverflow: https://survey.stackoverflow.co/2023/#learning-to-code-learn-code-learn

union, E. (19. December 2023). Pridobljeno iz consillium europa: https://www.consilium.europa.eu/sl/policies/artificial-intelligence/

Vercel. (2024). *Docs*. Pridobljeno 10. Marec 2024 iz NextJS: https://nextjs.org/docs

Vite. (2024). *Guide*. Pridobljeno 11. Marec 2024 iz Vite: https://vitejs.dev/guide/

Wikipedia. (19. Marec 2021). *JavaScript*. (Wikipedia) Pridobljeno 10. Marec 2024 iz https://sl.wikipedia.org/wiki/JavaScript

Wikipedia. (12. Marec 2024). Pridobljeno iz Umetna inteligenca: https://en.wikipedia.org/wiki/Artificial_intelligence#Philosophy

Wikipedia. (8. Marec 2024). Pridobljeno iz Technological singularity: https://en.wikipedia.org/wiki/Technological_singularity