AI pomoćnik za personalizirano vježbanje

Zoričić, Dominik

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1620

AI POMOĆNIK ZA PERSONALIZIRANO VJEŽBANJE

Dominik Zoričić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1620

AI POMOĆNIK ZA PERSONALIZIRANO VJEŽBANJE

Dominik Zoričić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1620

Pristupnik:	**Dominik Zoričić (0036542400)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Tomislav Jagušt

Zadatak: **AI pomoćnik za personalizirano vježbanje**

Opis zadatka:

AI pomoćnici (eng. AI companions) su sustavi temeljeni na umjetnoj inteligenciji koji korisnicima pružaju personaliziranu podršku u različitima domenama. U sklopu aplikacija za organizaciju i praćenje tjelesnih treninga ovi sustavi mogu pružiti personalizirane trening planove, pratiti napredak korisnika, prilagoditi vježbe prema individualnim potrebama, davati upute za pravilno izvođenje vježbi, nuditi nutricionističke savjete, motivirati korisnike i organizirati njihov trening, podsjećati ih na neke aktivnosti i sl. U sklopu ovog završnog rada potrebno je detaljno proučiti područje AI pomoćnika i njihove primjene u aplikacijama za vođenje treninga i organizaciju vježbanja. Potrebno je istražiti mogućnosti postojećih aplikacija ovog tipa i na temelju njih osmisliti i ostvariti vlastitu aplikaciju za praćenje i vođenje treninga, koja će implementirati AI pomoćnika. Prijavom u sustav korisnik će moći odabrati unaprijed pripremljene treninge, ili, uz savjete AI pomoćnika osmisliti i sastaviti vlastite. AI pomoćnik će ga također podsjećati na treninge te redovito davati druge savjete, kao što su dodatni opisi vježbi, savjeti vezani uz prehranu i sl.

Rok za predaju rada: 14. lipnja 2024.

Sarraj

Uvod

Revolucionarno razvijanje tehnologije i tehnološkog napretka osim što unosi velike promjene u naše živote također igra i veliku ključnu ulogu u oblikovanju istog. Između brojnih tehnoloških inovacija, jedno od najznačajnijih i neizostavnih dijelova suvremenog svijeta su web aplikacije i umjetna inteligencija (engl. Ratificirala intelligence AI).

Integracije web aplikacija i umjetne inteligencije predstavlja sljedeći korak u oblikovanju sustava koji omogućuju bolje korisničko iskustvo, napredno i pametno korištenje sustava kao što je analiziranje podataka, prilagođavanje korisničkim zahtjevima u realnom vremenu i slično.

U današnjem suvremenom i ubrzanom načinu života, redovita tjelovježba i aktivnost u obliku nekog sporta od velike je važnosti. Prema podacima istraživanja čak 62% populacije Hrvatske starije od 15 godina ne bavi se nikakvim oblikom sporta ili drugim vidom tjelesne aktivnosti [1]. Dakle sportom ili nekim oblikom aktivnosti bavi se 38% građana Hrvatske, među kojima je samo 12% onih koji treniraju tri do četiri puta tjedno, dok 9% čine oni koji treniraju i do 5 ili više puta tjedno. Jedan od glavnih razloga ovih rezultata istraživanja je needuciranost ljudi o raznim pristupima treniranja i prehrane te otežanog vođenja trenažnog procesa. Također veliki problem predstavlja dostupnost informacija kao i sama potreba za motivacijom i održavanjem dosljednosti.

Iz ovih razloga neupitna je potreba za fleksibilnim rješenjem koje će nuditi stvaranje personaliziranih trening programa koji se dinamički adaptiraju korisniku i bilo kakav oblik pomoći u vođenju tog procesa. Jedno takvo rješenje je upravo web aplikacija s integracijom umjetne inteligencije gdje se iskorištava lako dostupnost weba te personalizacija i adaptivnost korištenjem pametnih metoda umjetne inteligencije koji predstavljaju interaktivne AI osobne trenere.

Cilj ovog završnog rada je istražiti i implementirati jedno ovakvo fleksibilno rješenje i učiniti trenažni proces lakšim i pristupačnijim. Uzimajući u obzir funkcionalne zahtjeve, aplikacija će korisnicima biti dostupna nakon njihove registracije i otvaranja računa. Otvaranjem računa, korisnici će imati nekoliko ključnih funkcionalnosti na raspolaganju. Samostalno kreiranje trening programa za iskusnije vježbače koji ne trebaju pomoć, ali i prilagođeno sastavljanje istih gdje će neiskusniji korisnici moći postavljati pitanja svom AI

osobnom treneru na koje će dobivati željene odgovore i u skladu s time slagati svoje treninge. AI osobni trener ne samo da će moći kreirati treninge, već će korisnici moći postavljati pitanja o pravilnom izvođenju vježbi, broju ponavljanja, okvirnom trajanju treninga, prehrani i slično. Nakon što se treninzi kreiraju, svi će postati dostupni za pregledavanje putem korisnikovog osobnog kalendara. Svaki odrađeni trening moguće je evidentirati kroz sučelje aplikacije koje će omogućiti praćenje napretka i pružati korisnicima uvid u njihov razvoj. Praćenjem i evidentiranjem treninga, profil korisnika nudit će i statističke podatke koji će biti vizualno prikazani grafovima kao što su: preostali broj pojedinih treninga za tekući mjesec, aktivnosti treniranja kroz protekle mjesece kao i promjene u njihovoj kilaži.

U idućim poglavljima predstavljena su postojeća slična rješenja kao i njihove prednosti i mane. Detaljno je opisana arhitektura ovog sustava, shema baze podataka i korištene tehnologije. Prije samog kraja prikazana je demonstracija korištenja aplikacije, od otvaranja računa pa sve do prilagodbe aplikacije vlastitim potrebama te je za sami kraj ostavljen zaključak i pregled napravljenog.

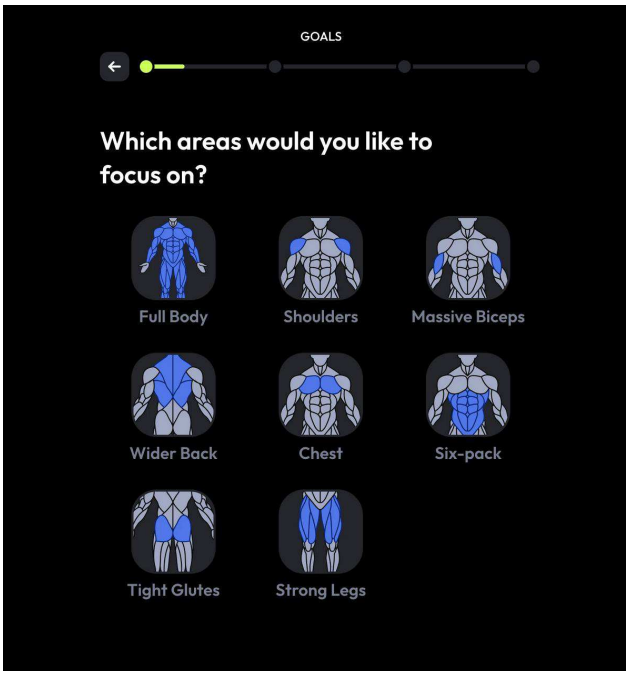
1. Slične aplikacije

Danas u svijetu fitnessa i treniranja postoje brojne aplikacije bazirane na ovoj temi s glavnim naglaskom olakšanog vođenja i praćenja trening procesa. U ovom poglavlju razmotrit ćemo trenutno neke od najpopularnijih dostupnih aplikacija i usporediti njihove funkcionalnosti.

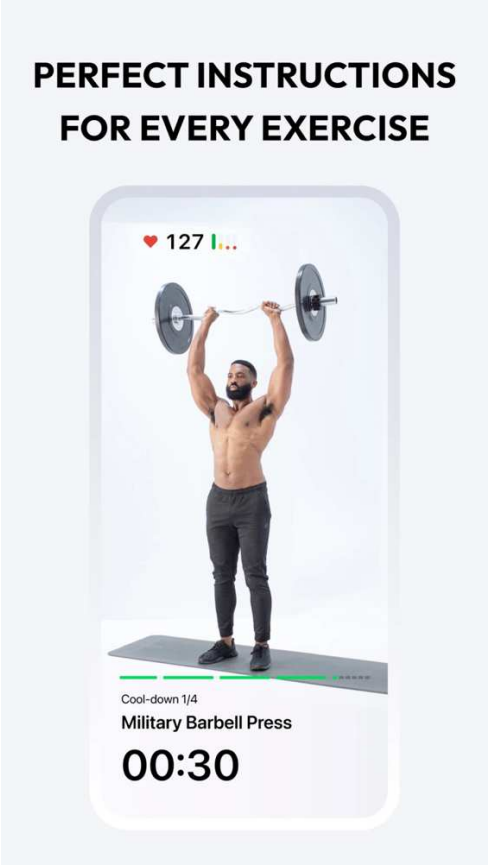
1.1. ZNG AI

zine AI jedna je od brojnih naprednih aplikacija koje imaju integriranu umjetnu inteligenciju za personalizaciju i prilagodbu različitim korisnicima i njihovim potrebama. Zing se ističe svojim brojnim funkcionalnostima koje pruža. Personalizirani planovi treninga stvaraju se i adaptiraju prilikom popunjavanja detaljnog formulara kod otvaranja računa prikazanog slikom (Sl. 1.1), kojem se predaju podaci uključujući dob, spol, fitness ciljeve, težinu, analizu tijela, razinu aktivnosti i slično. Također generirani treninzi i vježbe mogu biti prilagođeni ovisno o lokaciji korisnika, bilo da se radi o kućnom treningu, treningu na otvorenom ili u teretani, dostupne opreme, željenog intenziteta i posebnih dodatnih zahtjeva. Pritom svaka od vježbi ima svoj popratni demonstracijski video o pravilnom izvođenju vježbe prikazan slikom (Sl. 1.2). Osim integracije umjetne inteligencije Zing AI integrira se sa HealthKit-om i time omogućava laki uvoz i izvoz podataka o vježbanju što dodatno olakšava praćenje i evidenciju zdravstvenih metrika. Kako bi se održala visoka motivacija i dosljednost korisnika, aplikacija nudi prikazivanje motivacijskih ploča i izazova gdje se potiče natjecateljski duh i mogućnost pregledavanja rezultata povezanih korisnika [2].

Osim vođenja treninga aplikacija nudi i nutricionističke savjete kao dio svojih usluga. Aplikacija ima svoj Nutrition WorkBook koji pruža detaljan vodič i savjete o fitness prehrani kako bi svojim korisnicima omogućila lakše i bolje razumijevanje prehrane i zdravog načina života. Zing AI osim uz web sučelje pruža i mobilnu verziju aplikacije.



Sl. 1.1 Prikaz upita kod otvaranja računa u ZingAI aplikaciji [22]



Sl. 1.2 Video demonstracija izvođenja vježbi u ZingAI mobile aplikaciji [2]

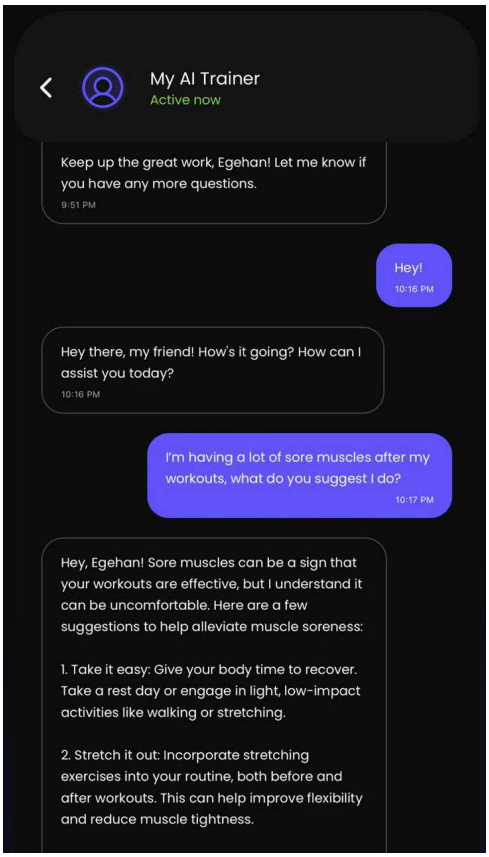
1.2. Coachify

Coachify je također web i mobilna aplikacija koja nudi usluge izgradnje personaliziranih treninga i praćenja napretka korištenjem umjetne inteligencije. Slično kao i prethodna aplikacija, za otvaranje računa popunjava se detaljni formular koji uzima sve potrebne informacije za izgradnju osobnog treninga. Također je moguća prilagodba vježbi ovisno o dostupnosti opreme i sprava za vježbanje. Budući da je pravilno izvođenje vježbi ključan faktor, aplikacija pruža animirane video demonstracije svake vježbe i detaljan opis generiran od strane umjetne inteligencije.

Iako je Coachify aplikacija primarno fokusirana na vođenje i evidentiranje treninga, nudi i praćenje prehrane gdje korisnici mogu pratiti unos svojih kalorija prikazano slikom (Sl. 1.3). Korištenjem umjetne inteligencije kreira se dnevni plan prehrane gdje se točno određuje koliko iznosi dnevni kalorijski unos te detaljni prikaz pojedinih makro nutrijenata kao što su proteini, masti i ugljikohidrati pojedinog obroka. Sve obroke moguće je zabilježiti u samoj aplikaciji [4].

Za bilo kakve druge oblike pomoći i pružanja informacija, aplikacija ima integrirani chat prema svom AI osobnom treneru prikazan slikom (Sl. 1.4) koji može odgovarati na bilo kakve upite i interese korisnika [3].

Sl. 1.3 Prikaz kalorija i makro nutrijenata u Coachify aplikaciji [4]



Sl. 1.4 Chat sučelje sa AI osobnim trenerom u Coachify aplikaciji [3]

2. Korištene tehnologije

U ovom poglavlju razmotrit ćemo sve tehnologije koje su korištene za izgradnju i implementaciju sustava.

Budući da se radi o web aplikaciji korištene su popularne tehnologije za lakše i brže razvijanje programske potpore. Arhitektura je temeljena na klijent-poslužitelj arhitekturi o kojoj će nešto više riječi biti u nadolazećim poglavljima.

2.1. Ruby on Rails

Poslužiteljska strana (engl. backend) aplikacije rađena je u tehnologiji Ruby on Rails ili poznatije samo kao Rails. Ruby je dinamički objektno orijentirani open source programski jezik koji ima fokus na jednostavnosti i produktivnosti s njegovom elegantnom i jednostavnom sintaksom koja je prirodna i laka za čitanje [5] . Rails je s druge strane web framework (Ruby gem) koji sadrži sve što je potrebno za izgradnju robusnih i skalabilnih web aplikacija podržanih bazama podataka temeljen na MVC arhitekturi. Rails je prilično poznat po svojoj filozofiji konvencije nad konfiguracijom (engl. convention over configuration) i načelu DRY (engl. „do not repeat yourself“), što pruža bržu implementaciju te jednostavniji i čitljiviji kod [6].

2.2. React

Klijentska strana (engl. frontend) aplikacije rađena je u tehnologiji React. React je jedna od najpopularnijih JavaScript biblioteka razvijen od strane Facebooka, koja se danas primarno koristi za izgradnju korisničkih sučelja. Olakšava izradu dinamičkih web aplikacija jer zahtijeva puno manje samog kodiranja i nudi puno više funkcionalnosti za razliku od pisanja samog JavaScript koda koji se u nekim situacijama može znatno zakomplicirati. React je poznat zbog svojih dobrih performansi a glavni razlog tome je što koristi virtualni DOM koji prati stanja komponenti i ažurira samo one komponente u stvarnom DOM-u koje su uistinu ažurirane umjesto ponovnog renderiranja svih komponenti [8]. Također valja naglasiti da je React baziran na komponentama koje predstavljaju gradivne blokove

aplikacije i omogućuje njihove ponovno korištenje na više mjesta. Osim web aplikacija, React se može koristiti i za izradu mobilnih aplikacija za što postoji okvir pod nazivom React native [7].

2.3. Ant Design

Ant Design poznata je React.js UI dizajnerska biblioteka koja omogućuje korištenje gotovih React komponenti prilikom izgradnje korisničkih sučelja web aplikacija [9].

Za njegovo korištenje potrebno ga je instalirati koristeći željene JavaScript package managere, npm ili yarn. Nakon instalacije uvoz/import željenih komponenti postaje moguć iz „antd“ paketa. Primjeri korištenja i importa Ant Design komponenti bit će prikazani u poglavljima koji slijede.

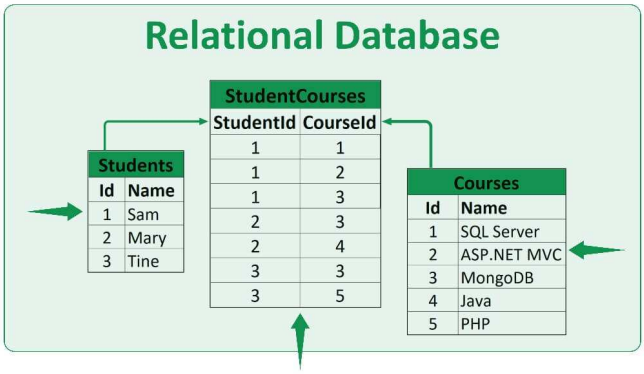
2.4. Git i GitHub

Git je specifični distribuirani sustav za verzioniranje (engl. version control) koji je nastao 2005. godine. Distribuirani sustav kontrole verzija zapravo znači da su sve verzije i baza promjena koda dostupne u računalu programera. Git je danas neizostavan alat u izgradnji softvera što govori i istraživanje Stack Overflowa koje je pokazalo da više od 87% programera koristi Git [10].

GitHub je web platforma koja pruža uslugu hostinga lokalnog Git repozitorija na oblaku. Omogućuje programerima na različitim lokacijama da istovremeno sudjeluju i rade na istom projektu koristeći u pozadini tehnologiju verzioniranja. GitHub pruža jednostavno korisničko sučelje koje olakšava korištenje Gita bez potrebe za znanjem svih Git naredbi koje se koriste u naredbenom retku. Ključna značajka GitHub-a je ta da programeri mogu međusobno pregledavati i komentirati izmjene koda putem Pull Requestova. Programer koji želi spojiti svoju granu s izvornom, commita i pusha svoje izmjene na remote repozitorij te potom kreira Pull Request. Ovime si programeri međusobno daju povratne informacije i odobre ili zatraže dodatne izmjene po potrebi prije samog spajanja u glavnu izvornu granu projekta [10].

2.5. PostgreSQL

Perzistencija podataka obavljena je koristeći relacijske baze podataka i PostgreSQL koji je moćan open source sustav za upravljanje bazama podataka podržavajući napredne SQL funkcionalnosti. Relacijske baze podataka su tip baza podataka koje omogućuju pohranjivanje i pristup podacima koji su međusobno na neki način povezani. Temeljene su na takozvanom relacijskom modelu koji je zapravo prikazivanje podataka u obliku tablica. Svaka tablica predstavlja jedan entitet s njegovim pripadnim atributima i pripadnom identifikacijskom oznakom ID. Svaki redak u pojedinoj tablici predstavlja jedinstveni entitet tako da ne postoje dva zapisa unutar jedna relacije/tablice koji imaju jednaki ID [12]. Relacijske baze jedne su od najpopularnijih jer omogućuju strukturirano i organizirano upravljanje podacima. Primjer jednostavne relacijske baze podataka prikazan je slikom (Sl. 2.1) [11].



Sl. 2.1 Primjer jednostavne relacijske baze podataka [11]

3. Dizajn i arhitektura sustava

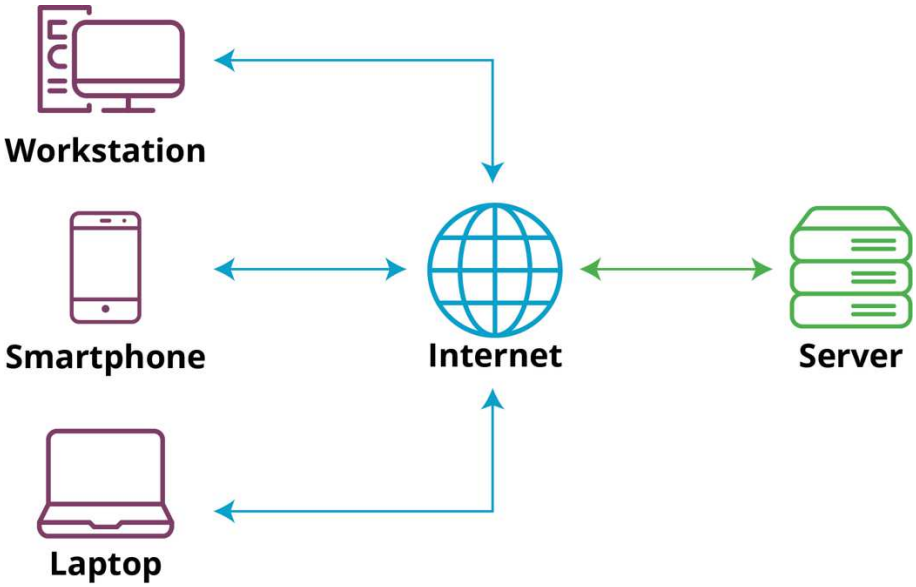
U ovom poglavlju razmotrit ćemo dizajn i arhitekturu aplikacije, koji su oblikovni obrasci korišteni, protok informacija od klijenta do poslužitelja i slično. Također prokomentirat će se i sigurnost aplikacije te metode autentifikacije i autorizacije koje su korištene za postizanje sigurnosti, privatnosti ali i ograničenja korisnika u korištenju aplikacije.

3.1. Klijent – Poslužitelj arhitektura

Klijent – Poslužitelj model je mrežne arhitekture u kojem se odgovornosti i poslovi dijele između pružatelja usluge (poslužitelj) i zahtjevatelja usluge (klijent). U ovakvoj arhitekturi poslužitelja predstavlja server na kojem se nalaze i obrađuju podaci dok klijenta predstavljaju obični korisnici (web preglednik) ili neka aplikacija koja traži uslugu za tim resursima [14].

U izgradnji ove web aplikacije korištena je arhitektura klijent-poslužitelj gdje je poslužiteljska strana razvijena kao REST API (engl. Representational State Transfer Application Programming Interface). REST API omogućuje klijentskoj strani (engl. frontend) i poslužiteljskoj strani (engl. backend) međusobnu komunikaciju i razmjenu podataka koristeći HTTP protokol i njegove standardne metode kao što su GET, POST, PUT, PATCH, DELETE.

Međusobna razmjena podataka između klijenta i poslužitelja odvija se slanjem i primanjem podataka u JSON formatu kao normu i dogovor komunikacije. Klijent tj. web preglednik korisnika šalje zahtjev za resursom koristeći HTTP GET metodu tražeći određeni resurs. Poslužitelj prima zahtjev korisnika i dostavlja tražene podatke reprezentirajući ih u JSON formatu. Pojednostavljeni prikaz toka informacija u klijent poslužitelj arhitekturi prikazan je slikom (Sl. 3.1) [15].



Sl. 3.1 Prikaz komunikacije klijenta i poslužitelja [15]

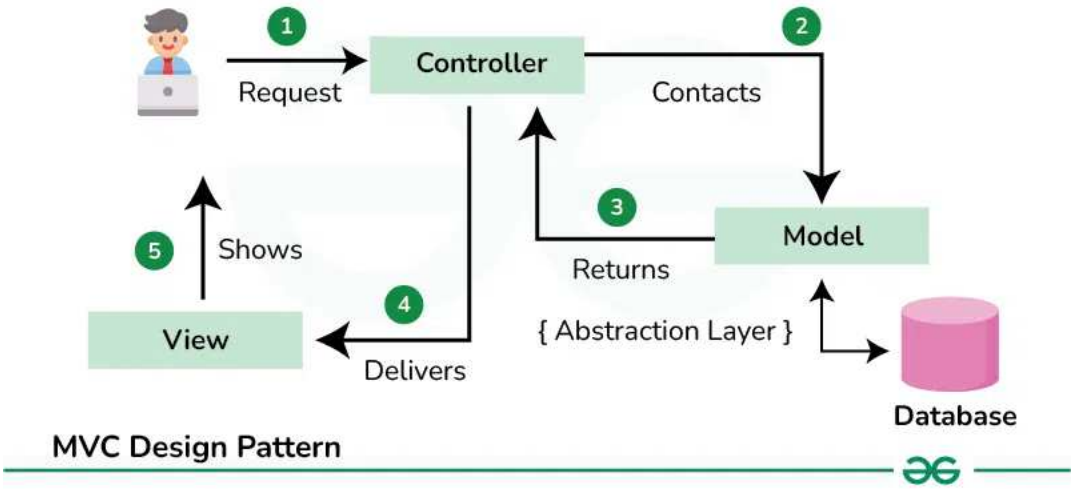
3.2. MVC – Model – View – Controller

Poslužiteljski dio aplikacije dizajniran je po jednom od najpopularnijih i najčešće korištenih obrazaca MVC. MVC je oblikovni arhitekturni obrazac koji propisuje da se aplikacija odvoji u 3 dijela tj. komponente a to su: M – model, V – view, C – controller [13]. Ovakav pristup u oblikovanju sustava omogućava bolju preglednost sustava, lakše održavanje i uvođenje novih promjena i funkcionalnosti. Osim toga MVC obrazac zahtjeva da se pojedini dijelovi koda koji pripadaju odgovarajućoj razini odvajaju u zasebne i različite komponente tj. razrede. Glavni razlog odvajanja razreda/komponenata u zasebne cjeline je taj što je svaki razred/komponenta nosi odgovornost za neki drugi aspekt funkcionalnosti aplikacije ili sustav. Koncept odvajanja odgovornosti u zasebne komponente također je ključan faktor za lakše održavanje i uvođenje novih promjena budući da se promjene nad jednom komponentom ne propagiraju na brojne druge, nego su lokalizirane i smanjene na minimum.

Model komponenta u MVC obrascu predstavlja podatke i svu logiku koju je potrebno izvršiti nad odgovarajućim podacima (poslovna logika aplikacije). Osim toga komponenta Model ostvaruje konekciju prema bazi podataka iz koje dohvaća podatke i obrađuje ih prema zahtjevima drugih komponenti kao što je primarno Controller. Osim dohvaćanja podataka spremanje podataka također je dio odgovornosti koje preuzima Model komponenta [13].

View komponenta odgovorna je za prikazivanje podataka i prosljeđivanje korisničkih unosa prema kontroleru koji onda na temelju toga obavlja odgovarajuće operacije i dohvaćanje potrebnih podataka. Za razliku od modela, view komponenta u pravilu nema poslovne logike i nema direktne interakcije s model komponentom [13].

Controller komponenta odgovorna je za prosljeđivanje podataka i upita između komponenti viewa i modela gdje ima ulogu posrednika. Obrađuje korisničke unose i zahtjeve te na temelju njih zahtijeva odgovarajuće promjene na modelima i posljedično promjene na view komponenti. Controller također provjerava valjanost unosa, validacije i transformacije podatka. Prikaz interakcije komponenti MVC obrasca prikazan je slikom (Sl. 3.2) [13].



Sl. 3.2 Prikaz komponenti i interakcija MVC obrasca [13]

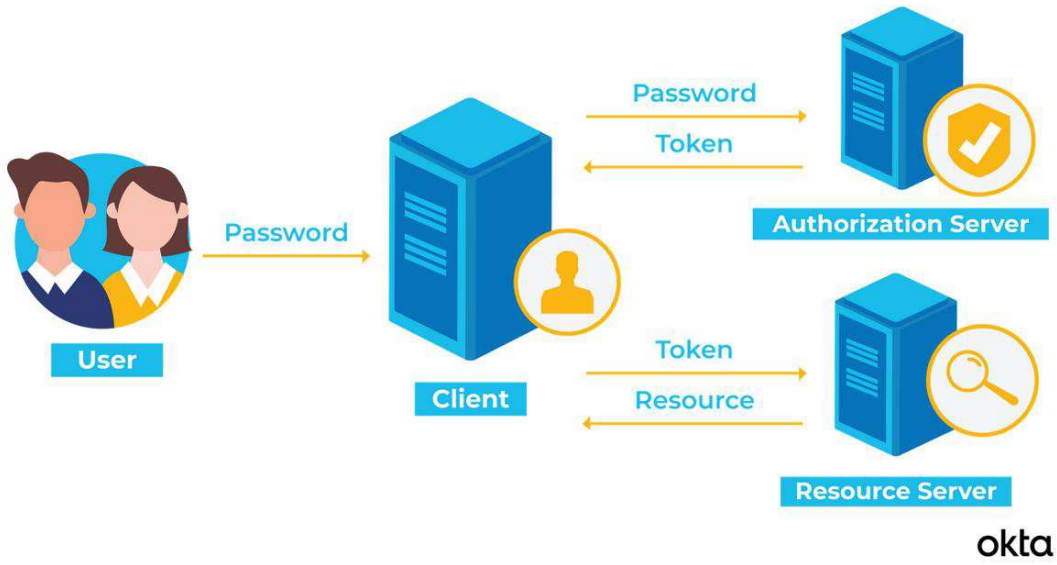
3.3. Sigurnost aplikacije

Sigurnost aplikacije ključna je komponenta svih aplikacija koje koriste klijent-poslužitelj arhitekturu. Osiguravanje sigurnosti web aplikacije uključuje niz mjera koje treba slijediti i kojih se treba pridržavati kako bi korisnički podaci bili zaštićeni od zlonamjernih korisnika i ostalih cyber napada.

Jedna od glavnih i ključnih mjera sigurnosti je autentifikacija i potvrda identiteta osobe koja se predstavlja da komunicira sa poslužiteljem. U izradi ove web aplikacije korištena je autentifikacija tokenima (engl. Token Based authentication). Autentifikacija tokenima protokol je koji omogućuje provjeru identiteta korisnika razmjenom i validiranjem tokena umjesto da korisnik pri svakom zahtjevu (engl. requestu) šalje svoje podatke kao što su

username i password kako bi potvrdio svoj identitet, korisnik zapravo šalje token koji je generiran od strane poslužitelja koji mu daje pristup resursima sve dok je token valjan. Nakon što se korisnik odjavi iz aplikacije token postaje nevažeći i ukinuta su prava pristupa sve do ponovne prijave u sustav i generiranja novog važećeg tokena. Protokol korištenja i razmjene tokena za autentifikaciju prikazan je slikom (Sl. 3.3) [16].

Još jedan ključan čimbenik u procesu ostvarenja sigurnosti aplikacije je autorizacija. Autorizacija je postupak određivanja radnji, akcija i dozvola koje korisnik smije obavljati na web aplikaciji temeljem svoje uloge ili role. Dobar primjer autorizacije je taj da administratori sustava smiju pregledavati podatke svih korisnika dok s druge strane obični korisnici smiju pregledavati isključivo vlastite podatke i podatke koju su u direktnoj poveznici s njima. Autorizacija i autentifikacija se može implementirati na razne načine a mi ćemo poslije objasniti i pokazati kako je to implementirano u ovoj web aplikaciji.

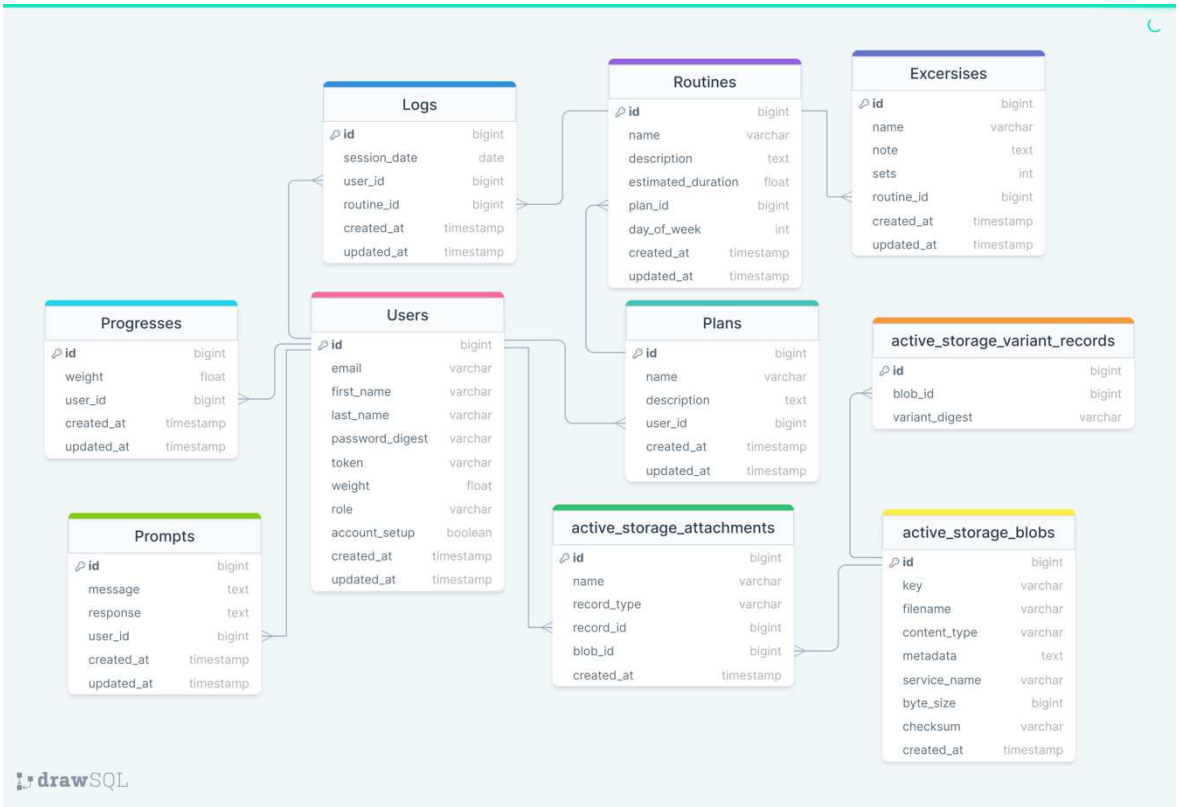


Sl. 3.3 Prikaz razmjene i korištenja tokena za autentifikaciju [16]

4. Implementacija

4.1. Model baze podataka

U nastavku ovog potpoglavlja bit će opisani svi entiteti i njihove relacije s ostalim entitetima kao i tipovi podataka. Na slici (Sl. 4.1) prikazan je relacijski model baze podataka razvijene aplikacije.



Sl. 4.1 relacijski model baze podataka

Users relacija predstavlja registrirane korisnike koji su se prijavili u sustav. Za svakog korisnika spremaju se sljedeći atributi:

- id – jedinstveno identificira svakog korisnika i generiran je automatski od baze podataka
- email – unosi se prilikom registracije i prijave u sustav
- first_name i last_name – ime i prezime korisnika
- password_digest – predstavlja hashirani password koji se pohranjuje nakon registracije

- token – služi za autentifikaciju što će detaljnije biti objašnjeno u narednim poglavljima
- weight – težina korisnika u kilogramima
- role – uloga korisnika za potrebe autorizacije i prava pristupa
- account_setup – boolean atribut koji je true ako je korisnik do kraja postavio sa svoj account a false ako još nije proveo postavljanje accounta nakon prve prijave,
- created_at i updated_at – atributi za evidentiranje izmjena nad accountom korisnika.

Progresses tablica koristi se za evidentiranje svih promjena kilaže korisnika. Za svaki zapis progress spremaju se sljedeći istaknuti atributi (atributi koji nisu navedeni vrijede analogno kao i na Users relaciji sa slike (Sl. 4.1)):

- user_id – jedinstveni identifikator korisnika za kojeg je zapis kreiran. One to many relacija prema Users tablici što zapravo znači da jedan progress zapis pripada točno jednom korisniku, dok s druge strane jedan korisnik (User) može imati više zapisa progress
- weight – atribut u koji se sprema težina. Naime svaki put kad korisnik ažurira svoju kilažu kroz sučelje aplikacije, u bazi će se pohraniti taj podatak kako bi se poslije mogao davati izvještaj o promjeni kilaže kroz vrijeme

Plans tablica koristi se pohranu korisničkih planova programa po kojima treniraju i vode svoje trenažne procese. Plans tablica zapravo je omotač jedne skupine treninga i vježbi koji pripadaju tom planu. Za svaki zapis u Plans tablici spremaju se sljedeći istaknuti atributi:

- name – ime plana programa treninga
- description – kratki opis o kakvom se planu programa treninga radi
- user_id – jedinstveni identifikator korisnika kojem konkretan plan programa pripada. One to many relacija prema relaciji Users isto kao i relacija Progresses.

Routines tablica koristi se za pohranu konkretnih treninga koje korisnici izvode za neki konkretan plan. Routines tablica u sebi bilježi atribute među kojima su najzanimljiviji:

- name – ime treninga
- description – kratki opis treninga i što se u treningu radi
- estimated_duration – predstavlja okvirno trajanje pojedinog treninga,

- plan_id – jedinstveni identifikator plana kojem trening pripada. One to many relacija prema tablici Plans što znači da jedan Plan može imati jedan ili više Routines zapisa tj. treninga dok s druge strane jedan Routine zapis pripada točno jednom planu.
- day_of_week - predstavlja dan u tjednu na sljedeći način: nedjelja – 0, ponedjeljak – 1, utorak – 2, srijeda – 3 , četvrtak – 4, petak – 5, subota – 6.

Exercises relacija koristi se za pohranu vježbi koje se izvode u pojedinom treningu. Exercises relacija sadrži sljedeće atribute:

- name – ime vježbe
- note – služi za pohranu nekih specifičnih detalja oko pojedine vježbe koje korisnik može zapisati
- sets – govori koliki broj setova te vježbe je potrebno izvoditi tijekom treninga
- routine_id – jedinstveni identifikator treninga kojem pripada. One to many relacija prema prema relaciji Routines analogno kako je objašnjeno na prethodnim primjerima.

Logs tablica koristi se za evidentiranje odrađenih treninga. Korisnik nakon kreacije plana pa zatim treninga i na kraju vježbi, može evidentirati svoj odrađeni trening za određeni datum. Evidencije odrađenih treninga poslije se koriste za prikazivanje frekvencije treniranja kroz protekli period. Atributi tablice Logs:

- session_date – datum odrađenog treninga
- user_id – jedinstveni identifikator korisnika za kojeg je zapis kreiran. One to many relacija prema Users tablici analogno kao na prethodnim primjerima
- routine_id – jedinstveni idenfitikator treninga kojeg je korisnik odradio i evidentirao. One to many relacija prema Routines tablici analogno prethodnim primjerima.

Prompts tablica služi za pohranu svih promptova tj. upita koje korisnik pošalje prema svom AI osobnom treneru. AI osobni trener implementiran je u obliku chata gdje se zapravo pitanje korisnika prosljeđuje na openAI API i od njega se bilježi odgovor. Dakle prompts tablica bilježi sljedeće atribute:

- message – poruka koju je korisnik postavio AI osobnom treneru
- response – odgovor od strane OpenAI API-a
- user_id – jedinstveni identifikator korisnika koji je postavio pitanje

Active_storage_blobs tablica koristi se za pohranu informacija o samim datotekama koje su spremljene u bazi. Svaka datoteka predstavljena je kao jedan blob i ima sljedeće atribute:

- key – jedinstveni ključ koji identificira datoteku
- filename – naziv pohranjene datoteke
- content_type – tip sadržaja (npr. image/jpeg)
- metadata – metapodaci poput dimenzija slik i slično
- byte_size – veličina datoteke u bajtovima
- checksum – kontrolni broj koji služi pri verificiranju integriteta datoteke

Active_storage_attachments tablica koristi se kao poveznica modela koji na sebi pohranjuju neke datoteke u aplikaciji. U ovom radu slike su se pohranjivale samo kao profilne fotografije korisnika pa je u ovom slučaju Active_storage_attachments tablica koja povezuje korisnka (User tablica) i Active_storage_blob tj. konkretnu datoteku. Neki od zanimljivih atributa tablice su:

- name – naziv pridružene datoteke na modelu s kojim ga povzuje (npr. Profile_picture, avatar, images i sl.)
- record_type – budući da je moguće pohranjivati datoteke i povezati ih s više model, ovaj atribut predstavlja tip modela na kojem je datoteka pridružena (u ovom radu to je user)
- record_id – jedinstveni identifikator modela na kojem je datoteka pridružena
- blob_id – jedinstveni identifikator datoteke

Active_storage_variants tablica koristi se za pohranu podataka o različitim varijantama pojedine datoteke. Najčešće se koristi upravo za slike koje mogu imati različite formate i dimenzije. Objašnjenja atributa ove tablice:

- blob_id - jedinstveni identifikator originalne datoteke
- variation_digest - hash vrijednost varijance koja identificira specifičnu varijantu

Tablice **plans**, **routines** i **exercises** u središtu su pozornosti i važno je objasniti odnose među njima. Da bi se kreirao zapis u tablici routine potrebno je prije imati kreiran plan treninga kojem će taj konkretni trening pripadati. Isto tako da bi se kreirala konkretna vježba (exercise), prije mora biti kreiran trening kojem ta vježba pripada a posljedično tome plan treninga mora postojati kako bi se trening mogao kreirati

4.2. RubyGems

Ruby kao i većina drugih programskih jezika sadrži veliki skup dodatnih biblioteka koje proširuju funkcionalnosti i olakšavaju implementaciju logike. Gotovo sve ruby biblioteke izdane su u obliku takozvanih Ruby gemova. RubyGems sustav je pakiranja programskog jezika Ruby kako bi olakšao i ubrzao instalaciju potrebnih paketa/gemova. Za pronalazak potrebnih gemova, najrasprostranjenije mjesto je javni repozitorij RubyGems.org [17].

U nastavku rada opisat će se kako su pojedini programski dijelovi implementirani i koji su ruby gemovi korišteni za ostvarenja tih funkcionalnosti.

4.3. OpenAI API integracija

Korištenje umjetne inteligencije u ovom radu postignuto je korištenjem gotovih i naprednih modela razvijenih od strane OpenAI-a, konkretno GPT-3 modela. OpenAI pruža svoje sučelje tj. API koji omogućava jednostavno pristupanje korištenjem HTTP protokola i njegovih standardnih metoda [18]. Da bi se ostvario pristup API-u potrebno je na stranici https://platform.openai.com/api-keys kreirati korisnički račun i generirati pristupni ključ (engl. key) s pomoću kojeg će se obavljati autentifikacija svakog zahtjeva, stoga je potrebno generirani ključ držati na sigurnom mjestu unutar projekta. Sigurnost i prikrivanje osjetljivih podataka kao što su lozinke, api ključevi i slično, u Ruby on Rails radnom okviru implementiran je koristeći credentials.yml.enc kriptiranu datoteku koju ćemo objasniti u odsječku sigurnosti aplikacije.

Nakon što smo generirali i na sigurno mjesto pohranili API ključ, potrebno je implementirati logiku za slanje HTTP zahtjeva. Slanje HTTP zahtjeva i oblikovanje samog zahtjeva tj. requesta implementiran je koristeći ruby-openai gem. Ovaj konkretni gem olakšava nam proces slanja zahtjeva i primanje odgovora koristeći HTTP protokol i olakšava implementaciju. Instalacija gema provodi se tako da se unutar Gemfile datoteke projekta napiše sljedeća linija:

```
gem 'ruby-openai'
```

te nakon dodane linije potrebno je u naredbenom retku instalirati novododani gem naredbom:

```
bundle install
```

Nakon izvršavanja prethodnih naredbi instalacija je gotova i gem je spreman za korištenje [19].

4.4. Implementacija ChatService klase

Logiku obrade korisničkih upita i zahtjeva implementirana je pomoću servisne klase `OpenAiApi::ChatService` koja kao argument za inicijalizaciju prima poruku odnosno zahtjev korisnika. Također navedena klasa pruža javnu metodu `call` koja radi pripremu i formatiranje podataka te otvaranje konekcije za slanje HTTP zahtjeva na openAI API. Izvorni kod `call` metode (Kod 4.1):

```
def call
      response = client.chat(
        parameters: {
          model: 'gpt-3.5-turbo',
          messages: [{ role: 'user' content: @message }],
          temperature: 0.7
        }
      )
      response.dig('choices', 0, 'message', 'content')
end
```

Kod 4.1 Izvorni kod metode call u klasi ChatService

Modul `OpenAiApi` je korišten iz razloga bolje organizacije i preglednosti te mogućnosti grupiranja srodnih metoda i konstanti te ponovno korištenje u različitim implementacijama razreda. Izvorni kod `OpenAiApi` modula:

```
module OpenAiApi
  OPEN_AI_API_KEY = Rails.application.credentials.open_ai_api_key
end
```

`OPEN_AI_API_KEY` konstanta služi za pohranjivanje API ključa koji je spremljen u kriptiranu credentials.yml.enc datoteku. Datoteka je kriptirana i moguće ju je otključati pomoću master ključa koji se nalazi u datoteci master.key unutar rails projekta.

Javna metoda `call` glavna je metoda ove servisne klase jer obavlja cijelu logiku slanja HTTP zahtjeva i primanje odgovora. Za generiranje HTTP zahtjeva potrebno je kreirati instancu klijenta za ostvarivanje konekcije s OpenAI API-jem. Prilikom instanciranja klijenta, koristi se tehnika memoizacije:

```
@client ||= OpenAI::Client.new…
```

kako bi se osigurala samo jedna instanca klijenta tijekom trajanja jednog korisničkog zahtjeva. Instanciranje se provodi predajom pristupnog ključa koji se nalazi definiran kao konstanta unutar modula `OpenAiApi` kako je prikazano na primjeru izvornog koda, te definiranjem atributa `log_errors` kojim se omogućava da se sve eventualne pogreške pri komunikaciji ispisuju kako bi se dobila točna i precizna informacija o pojedinostima potencijalnih grešaka.

Instanciranjem objekta klijenta, dostupna je javna metoda chat kojoj se šalju parametri za definiranje AI modela i sadržaja same poruke koja se prosljeđuje HTTP POST zahtjevom na OpenAI API i vraća odgovor na temelju zadanih parametara.

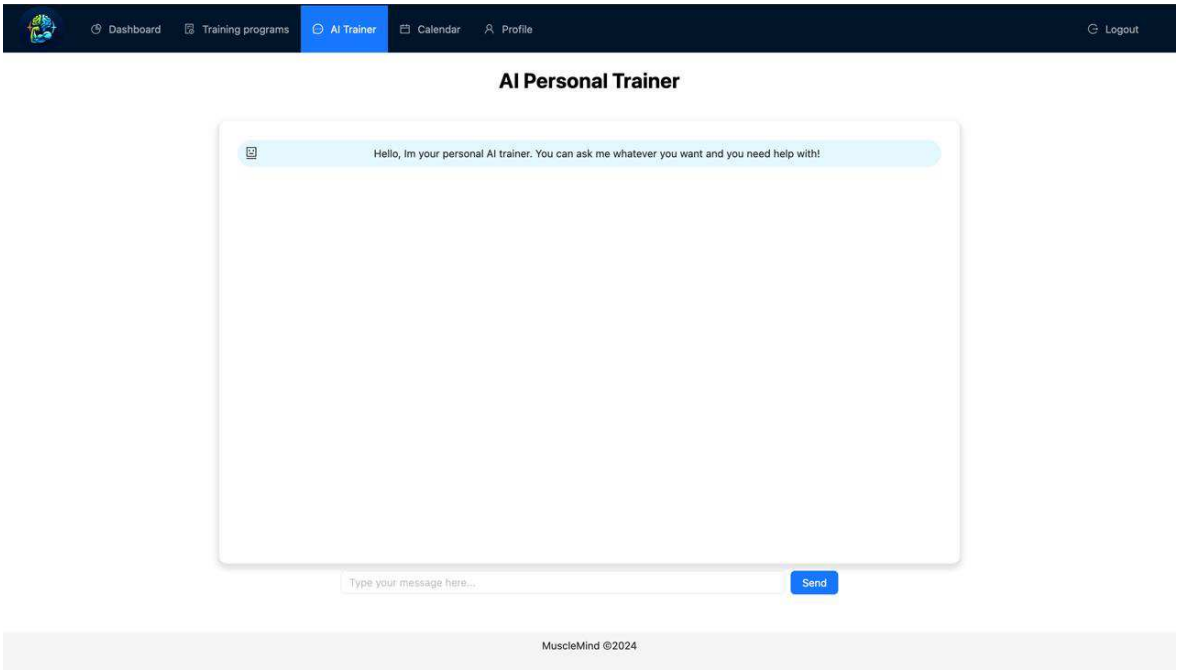
Objašnjenje parametara:

model – specificira model koji se koristi za generiranje odgovora. Za potrebe ovog rada koristio se „gpt-3.5-turbo“.

messages – lista JSON objekata poruka koji se šalju modelu. Svaki objekt poruke ima atribut `role` koji definira koja je uloga sudionika tj. osobe koja šalje zahtjev u razgovoru. Drugi atribut `content` sadrži stvarni izvorni tekst poruke koji se šalje. Atribut `temperature` definira mjeru neizvjesnosti i kontroliranosti odgovora, naime što je vrijednost `temperature` atributa veća to znači da će model davati kreativnije i nasumičnije odgovore dok niže vrijednosti atributa temperature predstavlja puno determinističnije odgovore [19]. Za potrebe ovog projekta koristi se: `role` user i `temperature` 0.7 budući da svi zahtjevi

koji idu prema OpenAI API-u potiču od korisnika i želi se postići veća kreativnost i raznovrsnost odgovora.

`OpenAiApi::ChatService` servisna klasa unutar rada aplikacije koristi se u dva načina primjene. Prva primjena je korisničko chat sučelje koje predstavlja direktnu komunikaciju sa svojim personalnim AI trenerom koji nudi bilo kakve oblike savjeta i pomoći prikazan slikom (Sl. 4.2). Druga primjena ove servise klase je slanje popunjenog formulara korisničkih podataka te željene strukture odgovora na temelju kojeg se onda radi personalizirani trening program prema željama korisnika i na temelju dobivenog odgovora formiraju se plan, treninzi i vježbe unutar aplikacije za korisnika.



Sl. 4.2 Prikaz chat sučelja prema AI osobnom treneru unutar aplikacije

4.5. Generiranje personaliziranih trening programa

Jedna od ključnih ideja ovog završnog rada bila je kreacija personaliziranih trening programa u ovisnosti o željama i karakteristikama svakog korisnika pomoću umjetne inteligencije. Osim generiranja treninga planova pomoću umjetne inteligencije korisnici bi isto tako mogli odabrati neki od unaprijed pripremljenih treninga. U nastavku objasnit će se cijeli postupak od prikupljanja korisničkih podataka pa sve do primanja informacija i generiranja sadržaja.

4.5.1. Implementacija PlanImporter klase

Odabir unaprijed pripremljenih treninga zahtijeva specificiranje formata podataka i definiranje klase koja će obavljati odgovornost funkcionalnosti kreiranja i upisivanja podataka umjesto samih korisnika. Prije definiranja klase objasnit ćemo potrebni format u kojem se podaci moraju nalaziti kako bi klasa uspješno obavila uvoz podataka. Primjer korištene strukture podataka (Kod 4.2):

```
{
  name: 'Example name',
  description: 'Example description',
  routines: [
    {
      name: 'Example name',
      description: 'Example description',
      estimated_duration: 75,
      day_of_week: 1,
      exercises: [
        {
          name: 'Example name',
          note: 'Example note',
          sets: 4
        }
      ]
    },
    ……
  ]
}
```

Kod 4.2 Struktura podataka zapisivanja trening programa

Na prikazanom izvornom kodu, definirana je točna struktura podataka koja se koristi kod klase za uvoz podataka, bilo da se radi o generiranim podacima od strane umjetne inteligencije ili se radi unaprijed definirani uvoz spremnih podataka.

Konkretna implementacija klase za uvoz podataka napisana je unutar `DataImporter::PlanImporter` razreda. `PlanImporter` slično kao i `ChatService` razred smješten je u modul `DataImporter` zbog istih razloga kao što je to objašnjeno na primjeru `ChatService` klase. `PlanImporter` razred za svoju inicijalizaciju prima sljedeće argumente. Korisnika koji je zatražio pripadnu akciju uvoza podataka, naziv trening programa ako se radi o odabiru već pripremljenih planova. Ako je to slučaj tada se na temelju naziva programa korištenjem dinamičkog polimorfizma dohvaća ime pripadne klase koja u sebi sadrži tražene podatke. Ako je slučaj da korisnik želi nove generirane personalne treninge, tada postoji i treći argument custom_data. Atribut custom_data zapravo je specificirana i generirana struktura podataka popunjena pravim podacima. Razred ima jednu javnu metodu import_data koja interno poziva jednu privatnu metodu perfom unutar koje se nalazi cijela logika uvoza podataka.

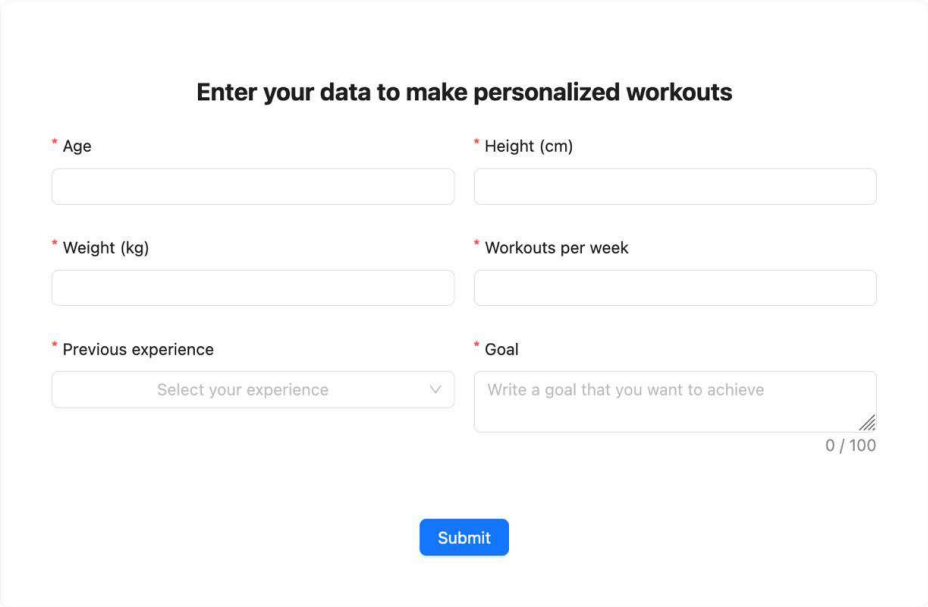
Metoda perform iterira kroz cijeli dobiveni objekt i redom kreira pripadne objekte na način da korektno postavlja i kreira prvo plan treninga a tek onda pojedine treninge i vježbe unutar tih treninga.

4.5.2. Generiranje prompta za OpenAI API

Generiranje personaliziranih trening programa zahtjeva prikupljanje podataka korisnika i njihovih karakteristika te formatiranja tih podataka, slanje na OpenAI API te formatiranje odgovora i obavljanje uvoza dobivenog odgovora s pomoću prethodno opisanog `DataImporter::PlanImporter` razreda. U ovom poglavlju objasnit će se klasa `Builder::PromptBuilder` koja se koristi upravo za generiranje pravilno oblikovane poruke/prompta za slanje na API.

Klasa `Builder::PromptBuilder` prima sljedeće podatke korisnika kao argumente za inicijalizaciju: `weight` (težina korisnika u kilogramima), `age` (starost korisnika), `height` (visina), `workouts_per_week` (broj željenih treninga po tjednu), `experience` (prethodno iskustvo), `goal` (cilj koji se želi ostvariti treningom). Upisivanje

podataka provodi se kroz sučelje u kojem korisnik specificira sve potrebne argumente prikazan slikom (Sl. 4.3):



Sl. 4.3 Sučelje za unos podataka pri izradi personaliziranih treninga

Nakon pritiska tipke submit podaci se šalju s klijenta na poslužitelj, gdje odgovarajući kontroler poslužitelja poduzima potrebne akcije. Konkretno instancira se objekt klase `Builder::PromptBuilder` koji onda nudi javnu metodu `build`. Metoda `build` koristi argumente koji su predani kod inicijalizacije i na temelju njih slaže odgovarajuću poruku. Izvorni kod metode `build` (Kod 4.3):

```
def build
  "Please create a detailed training plan for a person with the following characteristics: Weight: #{weight} kg, Age: #{age} years, Height: #{height} cm, Training Experience: #{experience}, Training Frequency: #{workouts_per_week} days per week, Fitness Goal: #{goal}. The response format should be in this form:
      " + example.to_s
end
```

Kod 4.3 Izvorni kod metode build

Privatna metoda `example` vraća primjerni objekt koji je opisan u prethodnim poglavljima kako bi podaci u odgovoru trebali biti formatirani, te se nad tim objektom poziva ruby metoda `to_s` koja isti objekt pretvara u `string`. Tako generirana poruka koristi se u prethodno opisanoj i definiranoj klasi `OpenAiApi::ChatService` kao `messages` atribut i poziva se metoda `call`. Nakon primitka odgovora od strane OpenAI API-a, podaci se predaju klasi `DataImporter::PlanImporter` koja dalje obavlja opisanu logiku uvoza i kreiranja podataka za korisnika.

4.6. Implementacija sigurnosti web aplikacije

Danas veliki problem brojnih web aplikacija predstavlja sigurnost, stoga ćemo ovo poglavlje posvetiti načinu kako su pojedine sigurnosne tehnike implementirane i kako se koriste. Kao što je već navedeno, izrađena aplikacija za proces autentifikacije koristi token based authentication tehniku. U ovom poglavlju objasnit ćemo implementacijske detalje, način komunikacije i slanje tokena između klijenta i poslužitelja.

Poslužiteljska strana pisana u Ruby on Rails radnom okviru nudi svoje snažne zaštitne mehanizme za generiranje sigurnih tokena i sigurno pohranjivanje kriptiranih vrijednosti. Dvije ključne metode koje pružaju ove funkcionalnosti su `has_secure_token` koja pruža generiranje sigurnih i jedinstvenih tokena [21] koje smo u ovoj aplikaciji koristili u svrhe autentifikacije i `has_secure_password` koja omogućava jednostavnu implementaciju sigurnog pohranjivanja hashiranih lozinki koristeći popularni algoritam bcrypt [20]. Budući da smo u aplikaciji autentificirali korisnika, pripadne metode koje nudi Rails okvir definirali smo na User modelu. User tablica iz tog razloga kao što je prije bilo pokazano ima atribute password_digest i token. Primjer jednostavnog user modela u kodu s navedenim metodama prikazan izvornim kodom (Kod 4.4):

```
class User < ApplicationRecord
  has_secure_password
  has_secure_token
  ……
end
```

Kod 4.4 Izvorni kod primjera klase User sa metodama za sigurnost

Da bi korisnik otvorio sesiju, potrebno je na poslužitelj poslati svoje podatke: email i lozinku, koji se onda na poslužitelju verificiraju i ako odgovaraju pohranjenim vrijednostima (lozinka se hashira i onda provjerava) korisnik kao odgovor poslužitelja dobije autentifikacijski token koji se onda pohranjuje u lokalni spremnik preglednika.

Svaki sljedeći zahtjev koji korisnik šalje na poslužitelj, potrebno je da se unutar tog zahtjeva uključi i autentifikacijski token umjesto da korisnik svaki put unosi svoje podatke: email i lozinku kako bi se verificirao što također predstavlja i sigurnosni problem. Na poslužitelju nakon što dođe zahtjev koji sadrži autentifikacijski token, u bazi podataka traži se korisnik koji ima pridružen dobiveni token te ako takav korisnik postoji radi se provjera samog tokena te ako je i token valjan, korisnik ima pravo pristupa poslužitelju. Za uništavanje sesije, korisnik šalje zahtjev koji se također verificira tokenom i ako je poslani token valjan, obavlja se ponovno generiranje tokena u bazi za tog korisnika i samim time prethodni token postaje nevažeći. Primjer izvornog koda za otvaranje sesije i spremanje tokena u lokalni spremnik klijenta (Kod 4.5):

```
const onFinish = (values) => {
    fetch("http://localhost:3000/api/session", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ session: values }),
    }).then((response) =>
        ……
      ).then(({ status, statusText, body }) => {
        if (status >= 200 && status < 300) {
          localStorage.setItem("user_id", body.session.user.id);
          localStorage.setItem("token", body.session.token);
           ……
      }}
```

Kod 4.5 Izvorni kod spremanja tokena u local storage preglednika

Primjer slanja novog zahtjeva na poslužitelj koristeći generirani token prikazan izvornim kodom (Kod 4.6):

```
try {
  const token = localStorage.getItem("token");
  const userId = localStorage.getItem("user_id");
  const response = await fetch(
    `http://localhost:3000/api/users/${userId}`,
    {
      method: "GET",
      headers: {
        Authorization: `${token}`,
        "Content-Type": "application/json",
      },
    }
  );
```

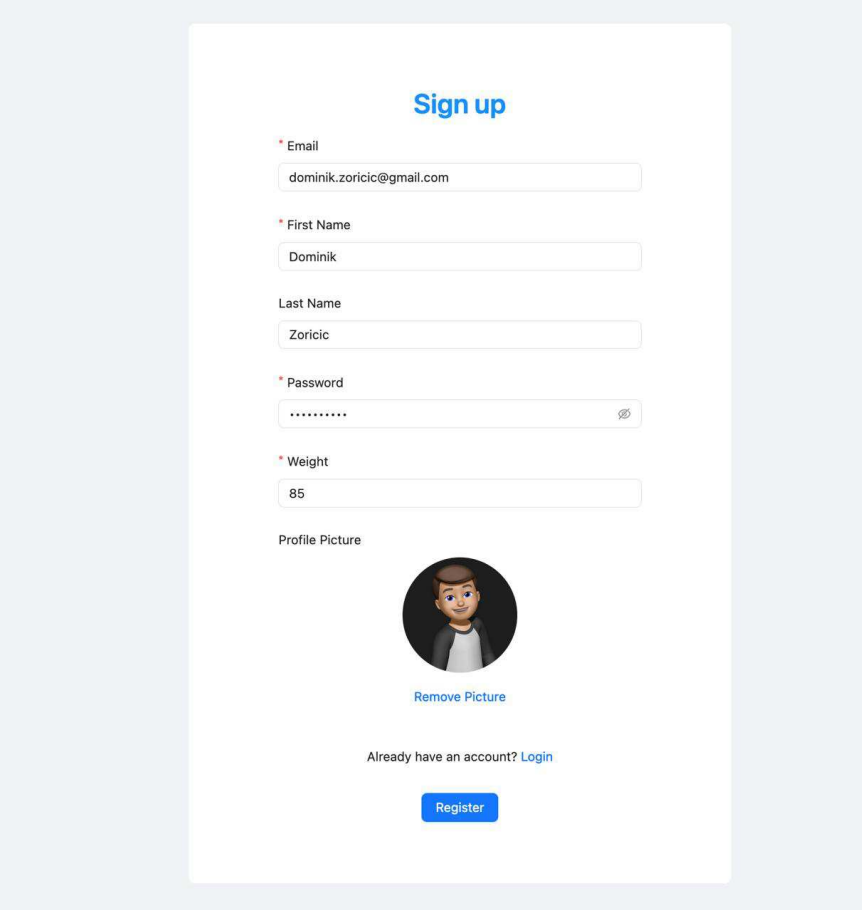
Kod 4.6 Izvorni kod slanja zahtjeva na poslužitelj koristeći token

5. Vodič za korištenje aplikacije

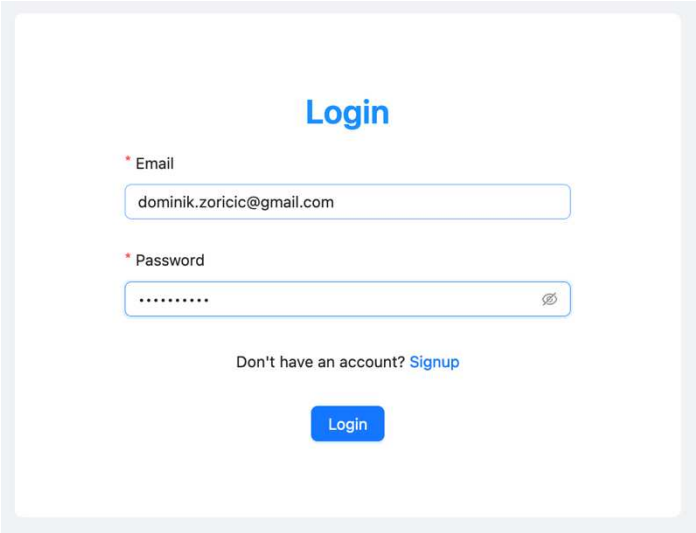
U ovom poglavlju pokazat ćemo na koji način se aplikacija koristi i kako se mogu iskoristiti sve funkcionalnosti koje nudi.

5.1. Registracija i prijava

Da bi korisnik mogao koristiti aplikaciju potrebno je prvo otvoriti korisnički račun registracijom i nakon toga prijaviti se u sustav. Na slikama (Sl. 5.1) i (Sl. 5.2) prikazana su sučelja za otvaranje računa i prijavu u sustav.



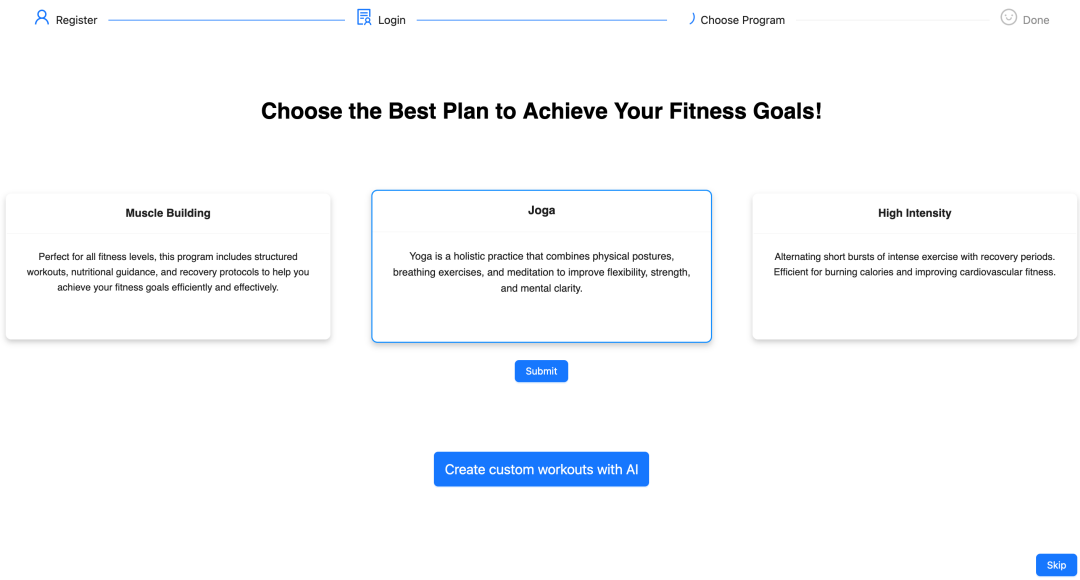
Sl. 5.1 Sučelje za registraciju



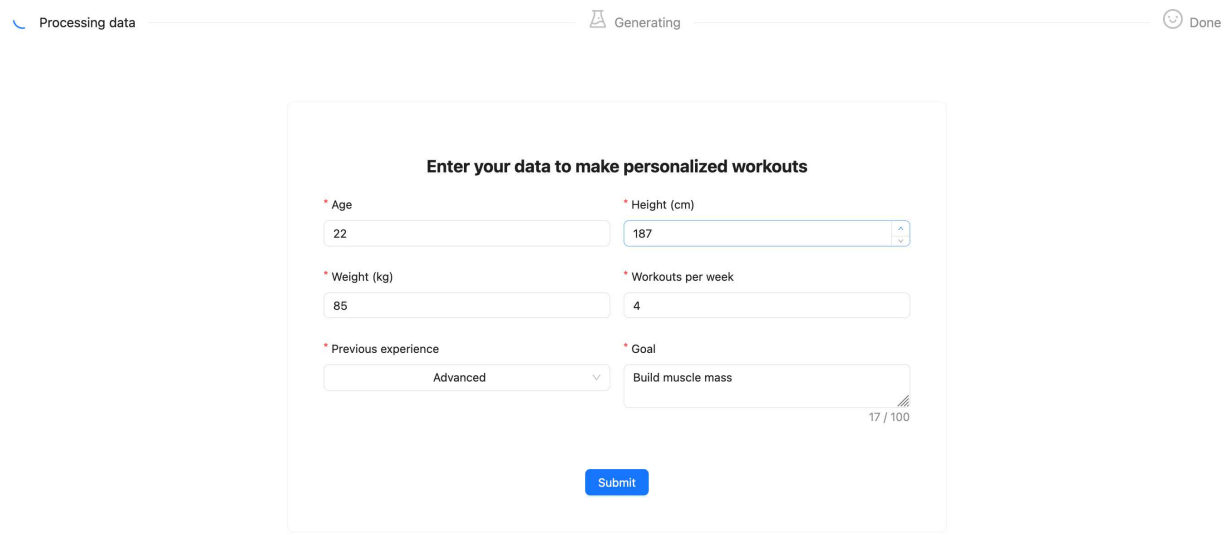
Sl. 5.2 Sučelje za prijavu u sustav

5.2. Kreiranje personalnih trening programa

Nakon uspješne registracije i prijave, korisnik može odabrati treninge koji se najčešće koriste i koji su unaprijed pripremljeni (Sl. 5.3). Druga mogućnost je da ako korisnik nema prethodnog iskustva u vježbanju, može iskoristiti integriranu umjetnu inteligenciju koja će mu na temelju njegovih želja konstruirati i posložiti personalizirani trening program unutar aplikacije (Sl. 5.4). Također ako se korisniku ne sviđa niti jedna od navedenih opcija, može preskočiti ove korake pritiskom na skip i sve kreirati samostalno.



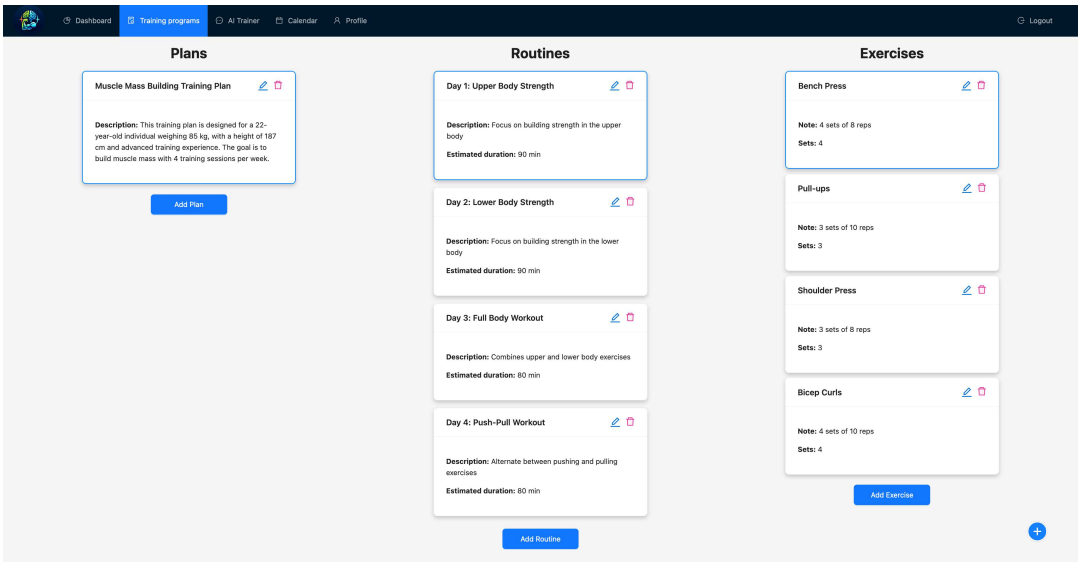
Sl. 5.3 Sučelje za odabir gotovih trening programa



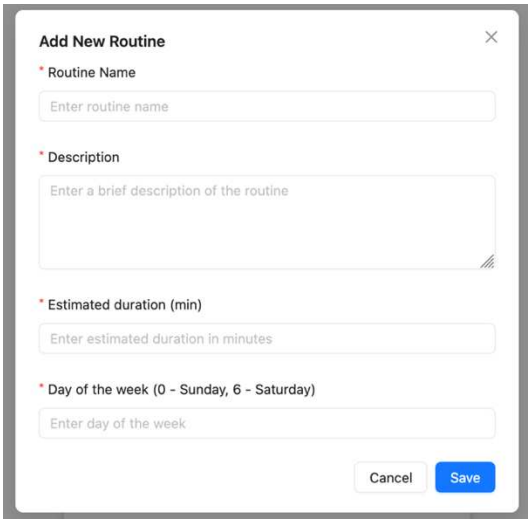
Sl. 5.4 Sučelje za kreiranje personaliziranih trening programa upisivanjem podataka

Pretpostavimo da korisnik u ovom trenutku želi da mu aplikacija kreira personalizirane treninge. Za kreaciju personaliziranih treninga potrebno ispuniti formular sa podacima koji rade procjenu samog korisnika ali i potpuno prilagođavaju trening program uzimajući u obzir njegove tjelesne metrike kao i njegove želje i ciljeve koje želi ostvariti. Nakon pritiska na submit, aplikacija obrađuje korisničke podatke i spaja se na openAI API gdje se šalju karakteristike korisnika kao i specifikacija željenog formata odgovora. Nakon obavljene kreacije treninga od strane umjetne inteligencije i uvoza tih podatka u aplikaciju, korisnik se preusmjerava na zaslon prikazan slikom (Sl. 5.5), gdje može pregledati svoje kreirane treninge. Pritiskom na pojedini plan otvaraju se svi treninzi za taj konkretni plan te također pritiskom na pojedini trening/rutinu prikazuju se sve vježbe za taj konkretan trening/rutinu.

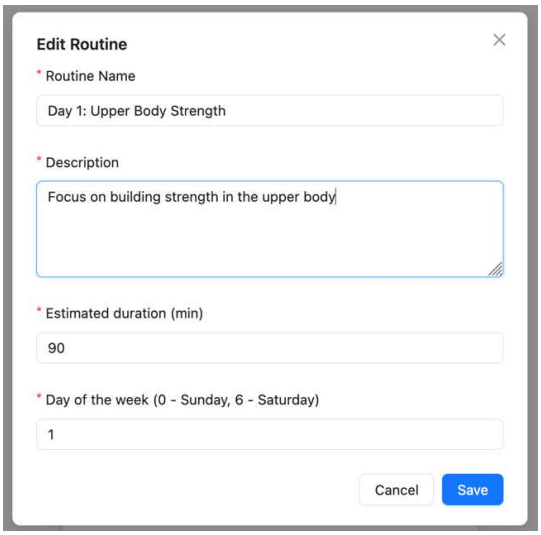
Također ukoliko se korisniku neki dijelovi generiranog plana treninga ne sviđaju, korisnik podatke može ažurirati, obrisati te dodavati nove otvaranjem modala prikazanim slikama (Sl. 5.6) i (Sl. 5.7). Također u donjem desnom uglu prozora nalazi se gumb oznake + koji korisnika vodi na prethodno pokazani zaslon kreacije personaliziranih treninga korištenjem umjetne inteligencije.



Sl. 5.5 Prikaz svih trenutnih planova, treninga i vježbi korisnika



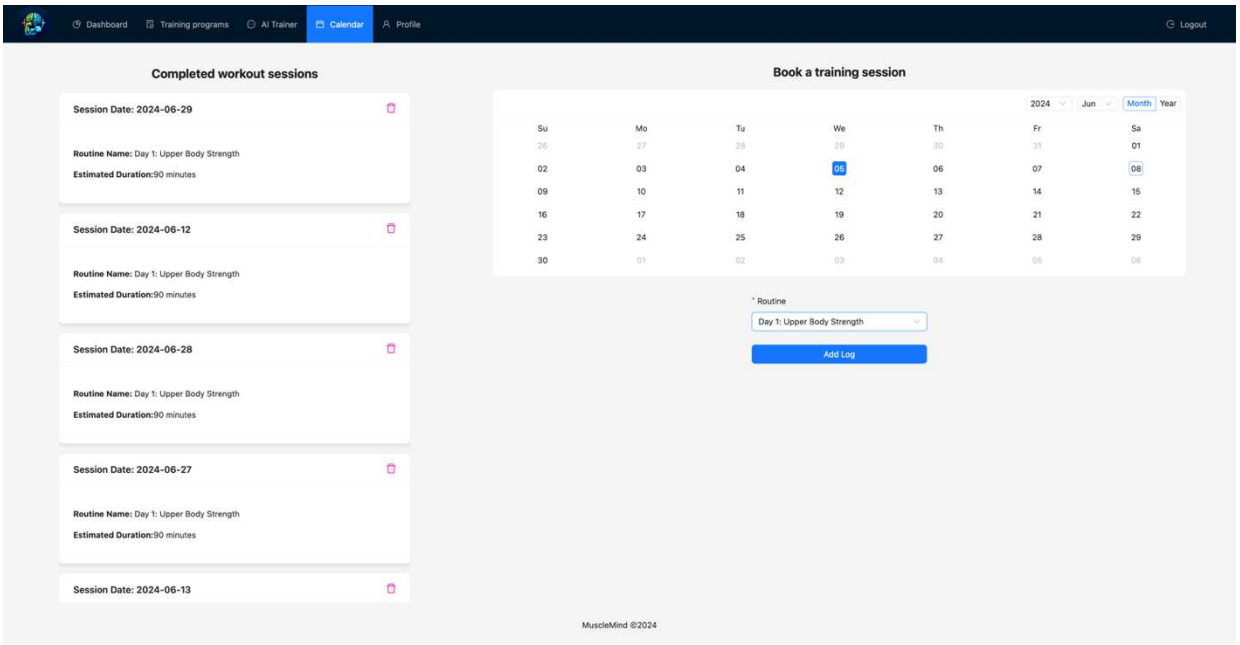
Sl. 5.6 Modal za kreiranje novog treninga



Sl. 5.7 Modal za editiranje postojećih treninga

5.3. Evidencija odrađenih i zakazanih treninga

Nakon generiranja ili samostalnog kreiranja trening programa, korisnik kroz aplikaciju može evidentirati odrađene treninge. Evidentiranje treninga odrađuje se putem sučelja prikazanog na slici (Sl. 5.8) gdje se na kalendaru odabire datum odrađivanja treninga, te na padajućem izborniku korisnik ima pravo odabrat jedan od svojih kreiranih treninga. Također na lijevoj strani zaslona, korisnik može pogledati popis svojih evidentiranih treninga te ih po potrebi obrisati.

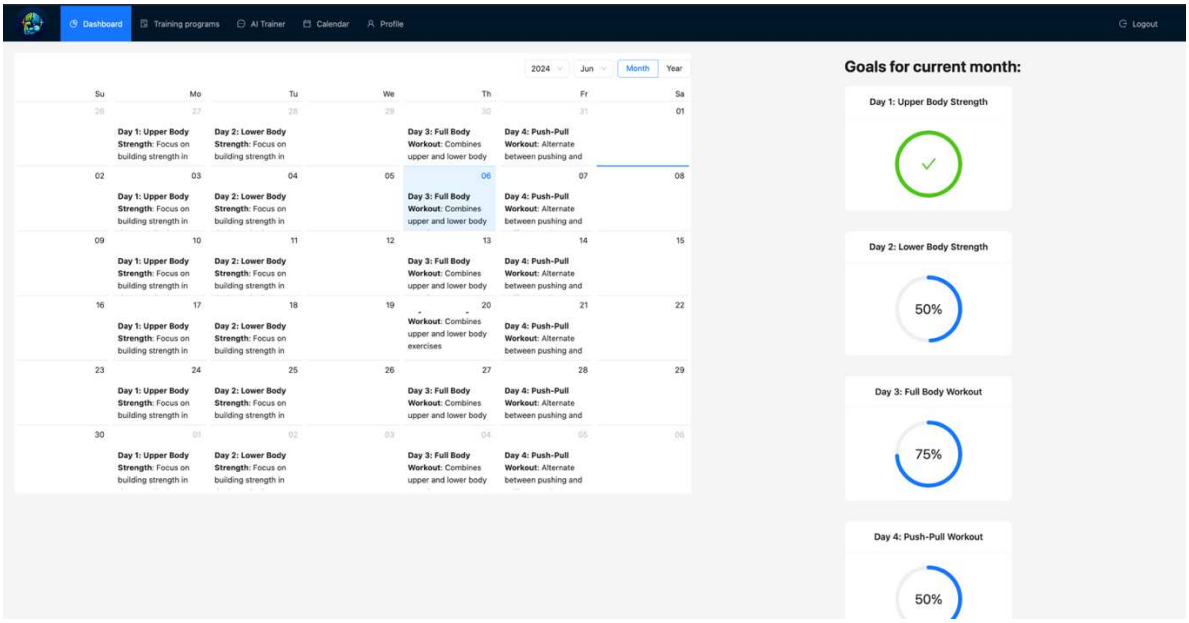


Sl. 5.8 Prikaz kalendara za evidenciju odrađenih treninga

Kao aplikacija koja prati i upravlja rasporedom treninga korisnika, korisnik ima na raspolaganju i zaslon koji mu prikazuje njegov raspored za tekući mjesec. Cilj ove funkcionalnosti je korisnike držati maksimalno organizirane i dosljedne. Osim rasporeda, korisnici također imaju uvid u postotak određenosti pojedinih treninga za tekući mjesec te im pružiti jasan i intuitivan pregled odrađenog.

Pregled rasporeda korisnik vidi kroz komponentu kalendara koji se puni s podacima ovisno o tome kako ih je korisnik kreirao. Dakle budući da za svaki trening treba definirati za koji je dan u tjednu predodređen, komponenta kalendara na temelju tih podataka unaprijed slaže raspored za bolje planiranje i jasan pregled. Postotak izvršenosti svojih pojedinih treninga gleda se na temelju broja zakazanih treninga te broja evidentiranih treninga koje je

korisnik odradio. Tako korisnici mogu brzo vidjeti koliko su napredovali prateći raspored i koliko im još preostaje za tekući mjesec. Prikaz navedenog zaslona prikazan je slikom (Sl. 5.9).

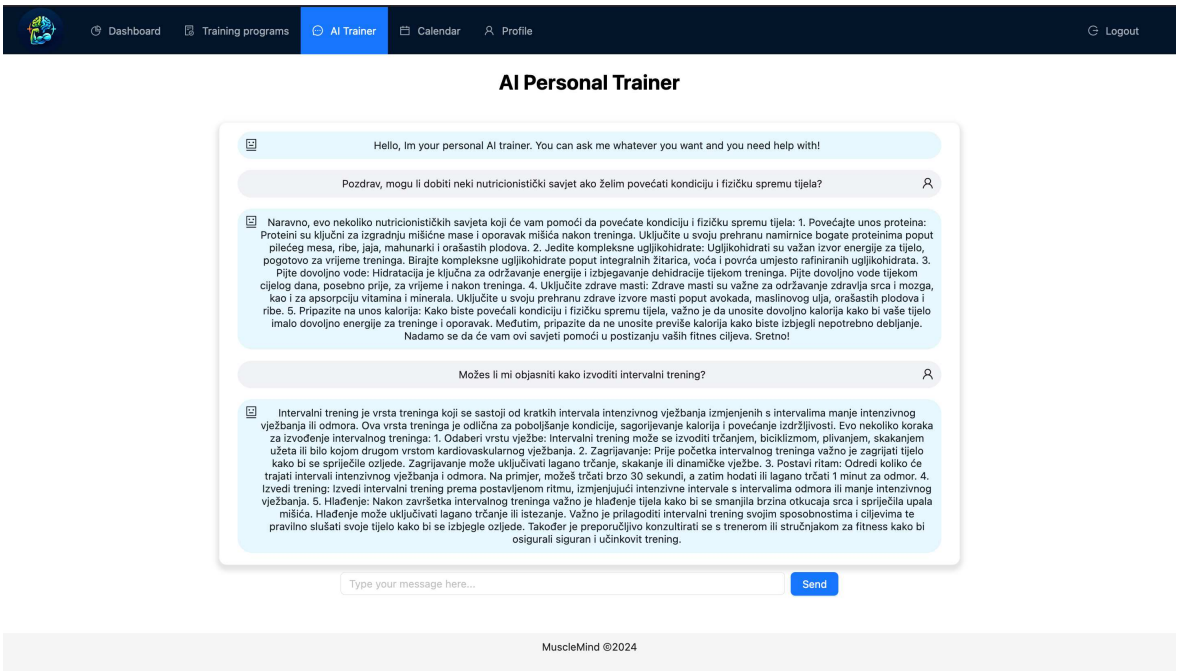


Sl. 5.9 Prikaz kalendara sa zakazanim treninzima i postotcima odrađenosti istih

5.4. Chat sučelje

Budući da korisnici nerijetko mogu imati pitanja i nejasnoće, neupitna je potreba za integracijom i implementacijom stalno dostupnog AI osobnog trenera za komunikaciju. Ova napredna funkcionalnost implementirana je u okviru chat sučelja koje omogućava korisnicima postavljanje bilo kakvih pitanja i primanja dodatnih savjeta pružajući im kompletnu podršku u postizanju ciljeva. Prikaz chat sučelja prikazan je slikom (Sl. 5.10)

Korisnici mogu postavljati bilo kakva pitanja vezana za trening, prehranu ili općenito fitness savjete. Nakon što korisnik postavi neko pitanje, pitanje se prosljeđuje na OpenAI API koji generira odgovor i koji se zatim prikazuje korisniku. Ova funkcionalnost omogućuje korisnicima da odgovore na svoja pitanja dobe u trenutnom vremenu te da odgovori budu dovoljno informativni i edukativni.



Sl. 5.10 Chat sučelje prema AI osobnom treneru

5.5. Profile page

Osim svim navedenih funkcionalnosti, korisnici imaju mogućnost pregledavanja i ažuriranja svojih podataka na svojoj profilnoj stranici.

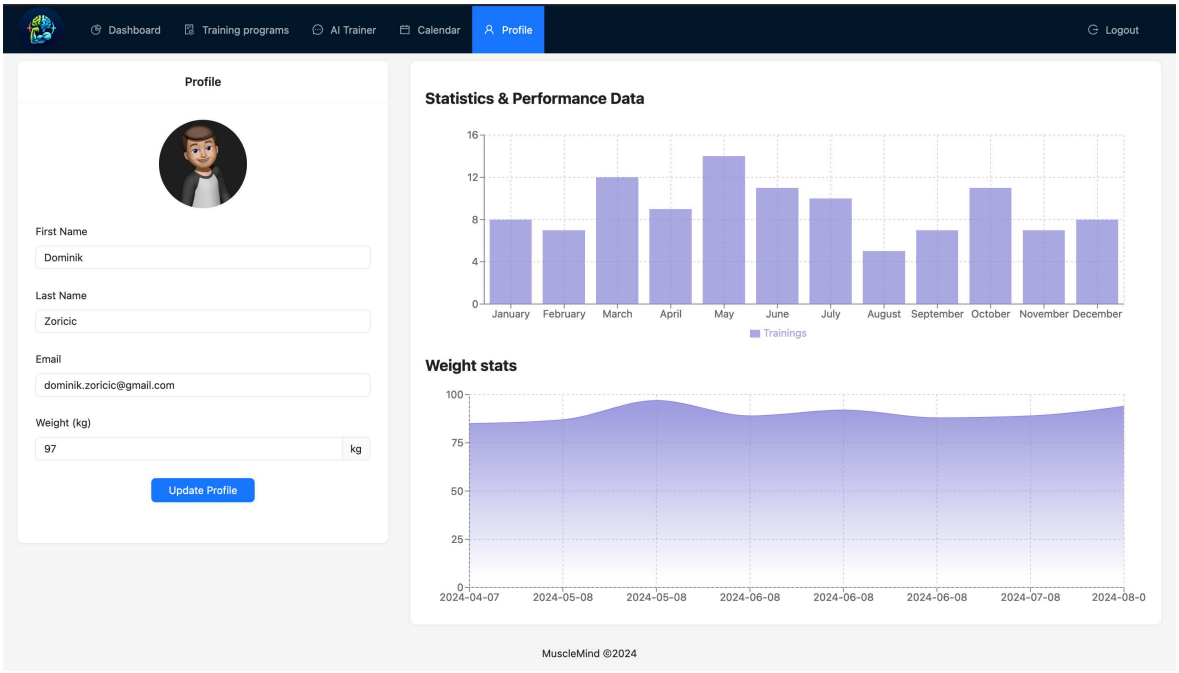
Profilna stranica predstavlja korisnika i njegove opće podatke kao što su: ime, prezime, email adresa, težina u kilogramima i slika profila. Osim općih podataka korisniku su predstavljena dva grafička prikaza koristeći grafikone aktivnosti.

Klijentska strana implementirana je koristeći JavaScript biblioteku React, koja nudi još mnoge druge biblioteke za olakšavanje implementacije novih funkcionalnosti. Grafikoni na profilnoj stranici implementirani su koristeći biblioteku ReCharts, popularnu React biblioteku koja omogućuje jednostavnu i intuitivnu grafičku prezentaciju podataka.

Grafikon pod imenom Statistics & Performance Data prikazuje aktivnost korisnika u tekućoj godini za svaki pojedini mjesec. Prislanjanjem miša na određeni stupac grafa korisniku se prikazuje broj odrađenih treninga za taj konkretan mjesec, pružajući detaljan uvid u postignuti napredak.

Grafikon pod imenom Weight stats prikazuje promjene u kilaži korisnika ovisno o tome kako je korisnik ažurirao svoju težinu kroz vrijeme. Prislanjanjem miša na određenu poziciju

grafa, prikazuje se precizna informacija o datumu zapisa i iznosu težine. Zaslon profilne stranice prikazan je slikom (Sl. 5.11).



Sl. 5.11 Zaslon profilne stranice i pripadnih grafova

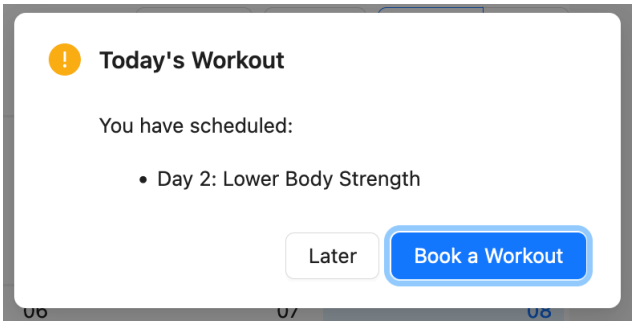
5.6. Obavijesti i podsjetnici

Aplikacija također kao jednu od funkcionalnosti nudi sustav notifikacija i podsjetnika koji će redovito podsjećati korisnike aplikacije na njihove potrebne aktivnosti za taj dan nakon njihove prijave u sustav.

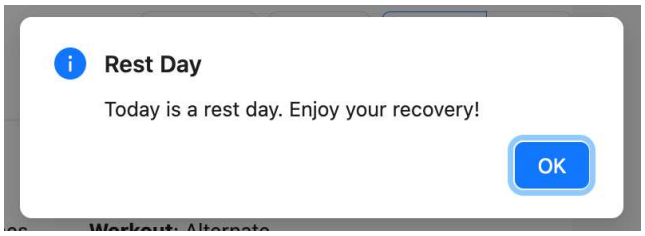
Dnevni podsjetnici za treninge uključuju obavijesti o zakazanim treninzima i danima na koje je zakazan odmor tj. dan bez treninga prikazani slikama (Sl. 5.12) i (Sl. 5.13). Ako korisnik za određeni dan ima zakazani trening, odmah nakon prijave u sustav korisniku će se otvoriti mali popup prozor u kojem će pisati detalji o zakazanom treningu tog dana. Popup prozor s prikazom obavijesti nudi mogućnost korisnika da odabere opciju „Later“ kako bi odgodio obavijest za poslije. U slučaju pritiska „Later“ obavijest će se ponovo pokazati nakon što se korisnik opet prijavi u sustav tog dana. Druga mogućnost je da korisnik odabere opciju „Book a workout“, čijim pritiskom će se preusmjeriti korisnika na stranicu za evidentiranje obavljenih treninga. Nakon što korisnik evidentira bilo koji trening za taj dan, obavijesti mu se tog dana više neće prikazivati. S druge strane ako korisnik za taj dan nema

zakazanih treninga, nakon prijave dobit će obavijest kao informaciju kako nema zakazanih treninga.

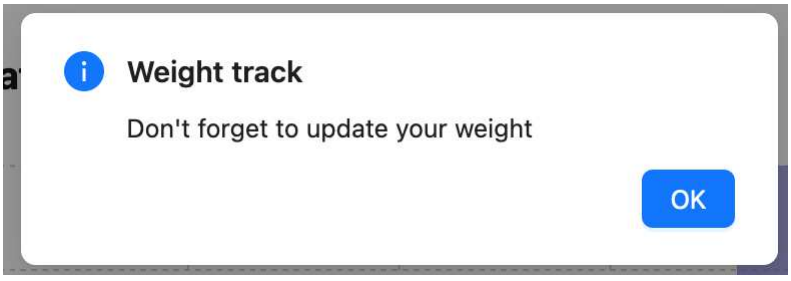
Tjedni podsjetnici redovito podsjećaju korisnika da ažurira svoju težinu barem jednom tjedno prikazan slikom (Sl. 5.14). Ako korisnik u tekućem tjednu nije ažurirao svoju kilažu, ulaskom na profilnu stranicu, korisnik će primiti obavijest koja će ga podsjetiti na tu radnju. Nakon što korisnik ažurira svoju težinu u tekućem tjednu, obavijesti o ažuriranju težine za taj tjedan više neće dobivati. Korisnici unatoč tim obavijestima imaju na pravo ažuriranja težine i više puta tjedno.



Sl. 5.12 Obavijest o zakazanim treninzima



Sl. 5.13 Obavijest da nema zakazanih treninga



Sl. 5.14 Obavijest o ažuriranju kilaže

Zaključak

Umjetna inteligencija kao jedan od najvećih tehnoloških valova našeg vremena ima jako veliki utisak na gotovo svaki aspekt našeg društva. Od jednostavnijih primjena pa sve do onih složenijih, AI mijenja način rada ali i navike svakodnevnog života. Međutim, uz konstantni napredak i evoluciju AI modela, većina populacije nije svjesna mogućnosti i dostupnosti ovih alata. Također jedan od velikih problema je nedostatak tjelesne aktivnosti u svakodnevnom životu ljudi uglavnom zbog nedostatka znanja o pravilnom pristupu, vremena i motivacije.

Cilj i ideja rada bila je implementacija sustava tj. web aplikacije koja će omogućiti korisnicima kreiranje personaliziranih trening planova te omogućiti pružanje bilo kakve pomoći i savjeta korištenjem umjetne inteligencije, te redovno podsjećati na pojedine aktivnosti. Osim toga razvijena aplikacija omogućuje potpuno praćenje napretka kroz vrijeme, uključujući grafički prikaz podataka kao što su napravljeni treninzi, promjena kilaže kroz vrijeme, raspored zakazanih aktivnosti i slično. Svi funkcionalni zahtjevi koji su bili zamišljeni i planirani, uspješno su implementirani.

Aplikacija ima prostora za daljnji napredak te ima potencijal napretka paralelno s razvojem novih i moćnijih AI modela. Napredak samih AI modela omogućit će još veću personalizaciju i prilagodbu sustava svakom pojedinom korisniku sa sve manje determinizma čime će se dodatno povećati učinkovitost ali i zadovoljstvo korisnika. Također prostor za napredak postoji u prostoru implementacije novih integracija s pametnim satovima, mobilnim uređajima (mobilna verzija ove aplikacije) i drugim nosivim uređajima te omogućavanja automatskog praćenja fizičkih aktivnosti i drugih relevantnih podataka.

Literatura

[1] Ministarstvo turizma i sporta Republike Hrvatske, Rezultati istraživanja o sportskoj i rekreacijskoj aktivnosti. Poveznica: https://mint.gov.hr/UserDocsImages//dokumenti-sdus/analize//Rezultati%20istra%C5%BEivanja%20o%20sportskoj%20i%20rekreacijskoj%20aktivnosti.pdf; pristupljeno 2. lipnja 2024.

[2] Softonic, Zing: AI Personal Trainer for iPhone. Poveznica: https://zing-ai-personal-trainer.en.softonic.com/iphone; pristupljeno 2. lipnja 2024.

[3] Coachify.ai, Coachify. Poveznica: https://coachify.ai/; pristupljeno 2. lipnja 2024.

[4] APKPure, Coachify AI. Poveznica: https://apkpure.net/coachify-ai/ai.coachify.coachify#ai.coachify.coachify-6; pristupljeno 2. lipnja 2024.

[5] Ruby, Ruby Programming langauge. Poveznica: https://www.ruby-lang.org/en/; pristupljeno 4. lipnja 2024.

[6] Rails, Ruby on Rails GitHub. Poveznica: https://github.com/rails/rails; pristupljeno: 4. lipnja 2024.

[7] React, React Documentation. Poveznica: https://react.dev/; pristupljeno: 4. lipnja 2024.

[8] Simplilearn, What is ReactJs. Poveznica: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs; pristupljeno: 4. lipnja 2024.

[9] GeeksforGeeks, Ant Design. Poveznica: https://www.geeksforgeeks.org/ant-design/; pristupljeno 5.lipnja 2024.

[10] Kinsta, What is GitHub? Poveznica: https://kinsta.com/knowledgebase/what-is-github/; pristupljeno 5. lipnja 2024.

[11] PragimTech, Relational and Non-Relational Databases. Poveznica: https://www.pragimtech.com/blog/mongodb-tutorial/relational-and-non-relational-databases/; pristupljeno 5. lipnja 2024.

[12] Oracle, What is a Relational Database? Poveznica: https://www.oracle.com/database/what-is-a-relational-database/; pristupljeno 5. lipnja 2024

[13] GeeksforGeeks, MVC Design Pattern. Poveznica: https://www.geeksforgeeks.org/mvc-design-pattern/; pristupljeno 5. lipnja 2024.

[14] GeeksforGeeks, Client-Server Model. Poveznica: https://www.geeksforgeeks.org/client-server-model/; pristupljeno 5.lipnja 2024.

[15] Molloy, Joseph, „What is Client-Server Architecture?“. Povenzica: https://www.liquidweb.com/blog/client-server-architecture/; pristupljeno 5. lipnja 2024.

[16] Okta, What is Token-Based Authentication? Poveznica: https://www.okta.com/identity-101/what-is-token-based-authentication/; pristupljeno: 5. lipnja 2024.

[17] Ruby, Libraries. Poveznica: https://www.ruby-lang.org/en/libraries/; pristupljeno 8. lipnja 2024.

[18] OpenAI, OpenAI API. Poveznica: https://openai.com/index/openai-api/; pristupljeno 8. lipnja 2024.

[19] Rudell, Alex, Ruby OpenAI GitHub. Poveznica: https://github.com/alexrudall/ruby-openai; pristupljeno: 8. lipnja 2024.

[20] Ruby on Rails, ActiveModel::SecurePassword::ClassMethods. Poveznica: https://api.rubyonrails.org/v7.1.3/classes/ActiveModel/SecurePassword/ClassMethods.html; pristupljeno 8. lipnja 2024.

[21] Ruby on Rails, ActiveModel::SecureToken::ClassMethods. Poveznica: https://api.rubyonrails.org/classes/ActiveRecord/SecureToken/ClassMethods.html; pritupljeno: 8. lipnja 2024.

[22] Zing Coach. Poveznica: https://zing-gym.coach/both/male?utm_source=Google&utm_medium=cpc&utm_campaign=4036_WEB_YD_GG_WW_FMFLOW_Brand_MaxConv_ALL_1502&adgroupid=164106952692&utm_term=zing%20ai&adpos=&gc_id=21013874802&gad_source=1&gclid=Cj0KCQjwsaqzBhDdARIsAK2gqneITHXk9SRGu4J8mke3R0hl5ifcxv8GNBKGIqlm4TtNmQ0ZLbIgY80aAmHkEALw_wcB; pristupljeno: 8. lipnja 2024.

Sažetak

Naslov:

AI pomoćnik za personalizirano vježbanje

Sažetak:

U sklopu ovog završnog rada razvijena je aplikacija za izradu personaliziranih trening programa te pružanje raznovrsne pomoći u vođenju trenažnog procesa korištenjem umjetne inteligencije. Implementacija i integracija umjetne inteligencije u web aplikaciju provedena je korištenjem popularnog OpenAI API-a, koju nudi moćne gotove AI modele i ima ključnu ulogu u radu aplikacije. Kroz rad je detaljno opisan postupak povezivanja i komunikacije implementiranog poslužitelja i API-a.

Cilj rada bio je iskoristiti napredne i inteligentne metode umjetne inteligencije kako bi se korisnicima bez prethodnog iskustva i znanja, omogućilo jednostavno kreiranje i praćenje treninga ili drugih tjelesnih aktivnosti. U radu je opisana arhitektura sustava, implementacija programskih rješenja te korištenje aplikacije i prikaz svih njenih funkcionalnosti.

Ključne riječi: React.js, Ruby on Rails, PostgreSQL, Umjetna inteligencija , AI, OpenAI API, fitness, personalizirani treninzi

Summary

Title:

AI assistant for personalized exercise

Summary:

As part of this final work, an application was developed for creating personalized training programs and providing a variety of assistance in managing the training process using artificial intelligence. The implementation and integration of artificial intelligence in the web application was carried out using the popular OpenAI API, which offers powerful ready-made AI models and plays a key role in the operation of the application. The paper describes in detail the connection and communication procedure of the implemented server and API.

The goal of the work was to use advanced and intelligent methods of artificial intelligence to enable users without previous experience and knowledge to easily create and monitor training or other physical activities. The paper describes the architecture of the system, the implementation of software solutions and the use of the application and the presentation of all its functionalities.

Keywords: React.js, Ruby on Rails, PostgreSQL, Artificial Intelligence, AI, OpenAI API, fitness, personalized trainings