SVEUČILIŠTE U ZAGREBU

FILOZOFSKI FAKULTET

ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI

(za diplomski rad) SMJER Informatika istraživački

Ak. god. 2024/2025

Bruno Križić

**Integracija Neuronskih Mreža u Personalizirano Savjetovanje o Prehrani: Razvoj i Primjena Web Aplikacije**

Diplomski rad

Mentor (ili Mentori): dr.sc. Sanja Seljan, red. prof.

Zagreb, lipanj, 2025

**Izjava o akademskoj čestitosti**

Izjavljujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

*Ovdje možete napisati kratku zahvalu ili stranicu možete ostaviti praznom.*

# Sadržaj

[Sadržaj ii](#_heading=h.c0ce5w3oh4f6)

[1.](#_heading=h.6qzbc5apqjo4) Uvod 1

[2.](#_heading=h.gm9mih6a8gi3) Teorijska podloga i analiza postojećih rješenja 2

[2.1.](#_heading=h.ogbwuoozl179) Tehnologije za razvoj web aplikacija 2

[2.1.1.](#_heading=h.m1e24r25czry) Python i radni okvir Flask 2

[2.1.2. 3](#_heading=h.ap3kh6ghjm6t)

[2.2. 4](#_heading=h.ozylktn7y6aq)

[2.2.1.](#_heading=h.hejd9hry0563) Ekspertni sustavi temeljeni na pravilima 4

[2.2.2.](#_heading=h.g980un9nek47) Učenje s potkrepljenjem kao napredni pristup 5

[2.3.](#_heading=h.d16jgancsztz) Veliki jezični modeli u konverzacijskoj umjetnoj inteligenciji 6

[2.3.1. 7](#_heading=h.sod3za8uok7b)

[2.4.](#_heading=h.warr4fs07q7o) Pregled i analiza srodnih rješenja 7

[3.](#_heading=h.np7gvdi2knwb) Dizajn i arhitektura sustava 9

[3.1.](#_heading=h.to2zo0oxnmml) Cjelokupna arhitektura aplikacije 9

[3.2.](#_heading=h.3ubt70o68xxg) Model podataka i dizajn baze podataka 11

[3.3.](#_heading=h.1ui02kegvcoz) Korišteni skupovi podataka 16

[3.4.](#_heading=h.dhbxnes2e01z) Proces pripreme i unosa podataka (ETL) 17

[4.](#_heading=h.jufjrtsivf7z) Implementacija ključnih funkcionalnosti 20

[4.1.](#_heading=h.fbhdb4s3pkzr) Korisničko sučelje (UI) 20

[4.1.1.](#_heading=h.br74zhswe4mm) Registracija korisnika (register.html) 20

[4.1.2.](#_heading=h.9tpl71rv8yxa) Nadzorna ploča (dashboard.html) 22

[4.1.3.](#_heading=h.tyrwjxcw5ksh) AI Virtualni Trener (smart\_input.html) 23

[4.1.4.](#_heading=h.d2836l5q3v4v) Tjedni izvještaj (weekly\_report.html) 24

[4.1.5.](#_heading=h.uh1ska8o4qos) Demografski uvidi (demographic\_insights.html) 25

[4.2.](#_heading=h.hdwe70ij1gp2) Sustav za upravljanje korisnicima i autentikaciju 26

[4.3.](#_heading=h.k1kjq8fvugrw) Implementacija sustava za generiranje preporuka 31

[4.4.](#_heading=h.uj2vwioq92pl) Implementacija inteligentnog asistenta i analize emocija 34

[4.4.1.](#_heading=h.l2ttbcjesh1f) Konverzacijski AI Virtualni Trener 34

[4.4.2.](#_heading=h.35diczy7a5t0) Analiza emocija 36

[5.](#_heading=h.ycbxoad6snex) Sigurnosni aspekti aplikacije 37

[5.1.](#_heading=h.nqabiwbxu182) Zaštita korisničkih podataka i autentikacija 37

[5.2.](#_heading=h.j0xjqr2a8n1b) Prevencija uobičajenih web napada 38

[5.3.](#_heading=h.3qyc6z7z1q23) Preporuke za sigurno produkcijsko okruženje 39

[6.](#_heading=h.o94wxvf8mymp) Rasprava i prijedlozi za budući rad 41

[6.1.](#_heading=h.3js3git4ky3i) Kritički osvrt na implementirano rješenje 41

[6.2.](#_heading=h.5pksuto8433w) Prijedlog unapređenja sustava preporuka primjenom učenja s potkrepljenjem (RL) 42

[6.3.](#_heading=h.vdzrzz22vsmv) Unapređenje povratne veze i integracija AI asistenta 43

[7.](#_heading=h.4ozkpkiaecyk) Zaključak 45

[8.](#_heading=h.f1xk0ckrdcji) Literatura 46

[Sažetak 49](#_heading=h.ny25j68j2zqm)

[Summary 50](#_heading=h.u2c7g5eqtf6g)

[Slika 4.1: Vizualni prikaz register.html 20](#_heading=h.mxbxjjh7xvre)

[Slika 4.2: Vizualni prikakaz dashboard.html 22](#_heading=h.k8layy3bwmwz)

[Slika 4.3: Vizualni prikaz smart\_input.html 24](#_heading=h.wuvjlgirijf4)

[Slika 4.4: Vizualni prikaz wee Weekly\_report.html 25](#_heading=h.rqntizfnn8i8)

[Slika 4.5: demographic\_insights.html 26](#_heading=h.vyrxas3x1nif)

[Tablica 3.1: Shema baze podataka aplikacije 13](#_heading=h.5k3yygx7fj8k)

[Tablica 4.1: Pregled ruta aplikacije 29](#_heading=h.8pqxf4quw79r)

[Tablica 4.2: Vizualni prikaz generiranja vježbi 31](#_heading=h.u5wqjaw640pe)

# Uvod

U suvremenom digitalnom dobu, koje karakterizira sveprisutnost tehnologije u svakodnevnom životu, personalizacija je postala ključni faktor u razvoju uspješnih tehnoloških rješenja, posebice u domeni zdravlja i wellnessa. Korisnici sve više traže alate koji se ne samo prilagođavaju njihovim individualnim potrebama, ciljevima i životnom stilu, već koji i uče te evoluiraju zajedno s njima. Tradicionalni, generički pristup praćenju fitnessa i prehrane, koji se često svodi na statične tablice i univerzalne planove, suočava se s fundamentalnim izazovima. Među njima se ističu niska stopa dugoročnog pridržavanja, preopterećenost općenitim i često kontradiktornim informacijama te poteškoće u primjeni planova koji ne uzimaju u obzir jedinstvene biometrijske, psihološke i kontekstualne karakteristike pojedinca. Kao odgovor na te izazove, javlja se potreba za integriranim, inteligentnim sustavima koji na jednom mjestu objedinjuju praćenje tjelesne aktivnosti, unosa nutrijenata i mentalnog stanja te pružaju dinamičnu i inteligentnu podršku koja korisnika aktivno vodi prema ostvarenju ciljeva. Motivacija za izradu ovog diplomskog rada proizlazi iz dubokog interesa za primjenu informacijskih i komunikacijskih znanosti u rješavanju konkretnih, svakodnevnih problema s ciljem poboljšanja kvalitete života. Želja je bila stvoriti alat koji korisnicima nudi holistički pristup brizi o zdravlju, kombinirajući strukturirano praćenje podataka s inteligentnom i personaliziranom podrškom koja nadilazi puko bilježenje aktivnosti. Razvoj ovakve aplikacije predstavlja izvanrednu priliku za istraživanje i primjenu suvremenih tehnologija i koncepata, uključujući agilni razvoj web aplikacija, napredne sustave za preporuke i konverzacijsku umjetnu inteligenciju, koja postaje sve dostupnija i moćnija.

# Teorijska podloga i analiza postojećih rješenja

Ovo poglavlje postavlja teorijski okvir neophodan za razumijevanje razvoja i implementacije web aplikacije za personalizirano praćenje fitnessa. Obuhvaća detaljan pregled ključnih tehnologija, dublju analizu paradigmi sustava za preporuke koje su relevantne za ovaj rad te pregled postojećih rješenja u domeni, s ciljem kontekstualizacije i opravdanja odabranih pristupa.

## Tehnologije za razvoj web aplikacija

Izbor odgovarajućih tehnologija temelj je za uspješan razvoj svake softverske aplikacije. Za ovaj projekt odabrane su tehnologije koje omogućuju brzu izradu prototipa, fleksibilnost i skalabilnost, s naglaskom na ekosustav programskog jezika Python, koji se nametnuo kao dominantna sila u području znanosti o podacima i umjetne inteligencije.

### Python i radni okvir Flask

Python se etablirao kao jedan od vodećih programskih jezika u području web razvoja, znanosti o podacima i umjetne inteligencije zbog svoje jednostavne i čitljive sintakse, bogate standardne knjižnice i golemog ekosustava dostupnih paketa. Njegova svestranost čini ga idealnim izborom za projekt koji integrira web tehnologije s obradom podataka i AI modelima.

Flask je lagani radni okvir (engl. *micro-framework*) za razvoj web aplikacija u Pythonu[[1]](#footnote-1). Njegova "mikro" priroda ne znači da mu nedostaju funkcionalnosti, već da je njegova jezgra namjerno mala i proširiva. Za razliku od monolitnih okvira poput Djanga, Flask ne nameće specifičnu strukturu projekta ili ovisnosti, što programerima daje potpunu slobodu u izboru alata i biblioteka (npr. za rad s bazom podataka ili validaciju formi) [[2]](#footnote-2). Ova filozofija čini ga izuzetno fleksibilnim i pogodnim za projekte koji zahtijevaju visoku razinu prilagodbe, kao i za brzu izradu prototipova[[3]](#footnote-3). Ključni koncepti Flaska uključuju:

* **Usmjeravanje (engl. *Routing*):** Mehanizam koji povezuje URL adrese s Python funkcijama koje obrađuju zahtjeve za te adrese. U Flasku se to postiže korištenjem dekoratora, poput @app.route('/'), što kod čini deklarativnim i lakim za praćenje[[4]](#footnote-4).
* **Predlošci (engl. *Templates*):** Flask koristi moćan sustav predložaka Jinja2, koji omogućuje generiranje dinamičkog HTML sadržaja umetanjem podataka iz Python koda u HTML datoteke. Jinja2 automatski provodi sigurnosno escapiranje podataka, što pruža temeljnu zaštitu od XSS napada[[5]](#footnote-5).
* **Aplikacijski kontekst (engl. *Application Context*):** Mehanizam koji omogućuje pristup objektima specifičnim za aplikaciju, poput konfiguracije ili baze podataka, unutar funkcija koje obrađuju zahtjeve, bez potrebe za prosljeđivanjem tih objekata kao argumenata.

U ovom projektu koristi se obrazac "aplikacijske tvornice" (engl. *application factory*), što je preporučena praksa za strukturiranje Flask aplikacija koje prerastu jednostavnu skriptu[[6]](#footnote-6). Umjesto globalnog stvaranja instance aplikacije, definira se funkcija (npr. create\_app()) koja stvara i konfigurira aplikaciju. To omogućuje stvaranje više instanci aplikacije s različitim konfiguracijama, što je iznimno korisno za testiranje i modularnost, kao što je vidljivo iz datoteka run.py i app/\_\_init\_\_.py.

### Python i radni okvir Flask

Interakcija s relacijskim bazama podataka iz programskog koda tradicionalno zahtijeva pisanje sirovih SQL upita. Objektno-relacijsko preslikavanje (engl. *Object-Relational Mapping*, ORM) je tehnika koja omogućuje programerima da s bazom podataka komuniciraju koristeći objekte programskog jezika u kojem rade, apstrahirajući pritom specifičnosti SQL dijalekta[[7]](#footnote-7).

SQLAlchemy je najpopularnija i najmoćnija ORM biblioteka za Python. Ona implementira obrazac **Data Mapper[[8]](#footnote-8)**, koji razdvaja domenske objekte (poslovnu logiku) od logike za pristup bazi podataka. To znači da Python klase koje predstavljaju podatke (modeli) ne sadrže kod za vlastito spremanje ili dohvaćanje, već za to postoji zaseban sloj (Session) koji upravlja tim operacijama[[9]](#footnote-9). Korištenjem SQLAlchemyja, složeni upiti, umetanja i ažuriranja podataka mogu se izvesti pozivanjem metoda na Python objektima, što kod čini čišćim, lakšim za održavanje i, što je ključno, manje podložnim SQL injekcijskim napadima jer SQLAlchemy automatski parametrizira upite[[10]](#footnote-10). Ekstenzija Flask-SQLAlchemy dodatno pojednostavljuje integraciju SQLAlchemyja u Flask aplikaciju, automatski upravljajući životnim ciklusom sesija baze podataka unutar aplikacijskog konteksta[[11]](#footnote-11).

## Sustavi za preporuke

Sustavi za preporuke (engl. *recommender systems*) ključni su za personalizaciju korisničkog iskustva. U domeni zdravlja, njihov cilj je pružiti korisniku relevantne i pravovremene preporuke za vježbe, obroke ili druge aktivnosti koje će mu pomoći u ostvarenju ciljeva. U ovom radu analiziraju se dvije temeljne paradigme: ekspertni sustavi temeljeni na pravilima, koji su implementirani u aplikaciji, i učenje s potkrepljenjem, kao napredni pristup predložen za budući razvoj.

### Ekspertni sustavi temeljeni na pravilima

Ekspertni sustavi temeljeni na pravilima (engl. *Rule-Based Expert Systems*) predstavljaju jedan od klasičnih pristupa u umjetnoj inteligenciji, čiji korijeni sežu u 1970-e godine[[12]](#footnote-12). Njihov cilj je oponašati proces donošenja odluka ljudskog stručnjaka u usko definiranom području (domeni)[[13]](#footnote-13). Ovi sustavi koriste bazu znanja, predstavljenu kao skup IF-THEN pravila, kako bi donosili zaključke na temelju dostupnih činjenica[[14]](#footnote-14).

Ključne komponente ekspertnog sustava su[[15]](#footnote-15):

1. **Baza znanja (engl. *Knowledge Base*):** Sadrži ekspertno znanje o domeni u obliku produkcijskih pravila. Pravilo ima strukturu "IF [uvjet ili premisa] THEN [akcija ili zaključak]". Na primjer: "IF korisnikov cilj je 'gubitak težine' AND razina fitnessa je 'početnik' THEN preporuči kardio vježbe niskog intenziteta".
2. **Baza činjenica (engl. *Fact Base* ili *Working Memory*):** Sadrži podatke specifične za trenutni problem, odnosno činjenice na koje se pravila primjenjuju. U kontekstu ove aplikacije, to su podaci o trenutnom korisniku (npr. dob, cilj, razina fitnessa).
3. **Stroj za zaključivanje (engl. *Inference Engine*):** Komponenta koja primjenjuje pravila iz baze znanja na činjenice iz baze činjenica kako bi izvela nove zaključke. Stroj za zaključivanje koristi jednu od dvije glavne strategije[[16]](#footnote-16):

* **Zaključivanje unaprijed (engl. *Forward Chaining*):** Ovo je strategija vođena podacima (*data-driven*). Proces započinje s poznatim činjenicama i primjenjuje pravila redom kako bi se izvele sve moguće posljedice. Na primjer, polazeći od korisničkih podataka, sustav "pali" sva primjenjiva pravila dok ne generira konačni plan vježbanja. Ovo je pristup koji je implicitno korišten u implementiranoj aplikaciji, gdje Python interpreter djeluje kao stroj za zaključivanje.
* **Zaključivanje unazad (engl. *Backward Chaining*):** Ovo je strategija vođena ciljem (*goal-driven*). Proces započinje s hipotezom ili ciljem i pokušava pronaći pravila i činjenice koje ga podržavaju, radeći "unazad" od zaključka prema uvjetima.

Prednosti ovog pristupa su transparentnost (pravila su eksplicitna i lako razumljiva), predvidljivost i jednostavnost implementacije za probleme s dobro definiranim pravilima[[17]](#footnote-17). Međutim, ključni nedostaci su krutost i nedostatak sposobnosti učenja; sustav se ne može automatski prilagoditi ili poboljšati na temelju korisničkih interakcija, a održavanje velikog broja pravila može postati složeno[[18]](#footnote-18).

### Učenje s potkrepljenjem kao napredni pristup

Učenje s potkrepljenjem (engl. *Reinforcement Learning*, RL) je grana strojnog učenja u kojoj softverski agent uči kako se ponašati u određenom okruženju tako što izvodi akcije i promatra rezultate, odnosno nagrade[[19]](#footnote-19). Za razliku od sustava temeljenih na pravilima, RL sustavi mogu dinamički učiti i prilagođavati svoju strategiju (politiku) kako bi maksimizirali kumulativnu nagradu tijekom vremena, čime se rješava problem statičnosti[[20]](#footnote-20).

U kontekstu sustava za preporuke, RL se može formalizirati kroz koncepte **Markovljevog procesa odlučivanja (MDP)[[21]](#footnote-21)**:

* **Stanje (State):** Predstavlja trenutni kontekst korisnika. To može biti vektor koji uključuje demografske podatke, povijest interakcija (prethodno odabrane vježbe, ocijenjeni obroci), pa čak i kontekstualne informacije poput dana u tjednu ili doba dana.
* **Akcija (Action):** Predstavlja preporuku koju sustav daje korisniku u određenom stanju (npr. preporučiti određeni set vježbi ili obrok).
* **Nagrada (Reward):** Numerički signal koji agent dobiva nakon izvođenja akcije. Nagrada kvantificira uspješnost akcije. U sustavu za preporuke, nagrada može biti eksplicitna (npr. korisnikova ocjena preporuke) ili implicitna (npr. je li korisnik kliknuo na preporuku, odradio preporučeni trening). Cilj RL agenta je naučiti politiku (strategiju odabira akcija) koja maksimizira zbroj budućih nagrada[[22]](#footnote-22).

Znanstvena literatura pokazuje sve veću primjenu RL-a u domeni personalizirane prehrane i zdravlja. Istraživanja pokazuju da RL modeli, poput onih temeljenih na DDPG (engl. *Deep Deterministic Policy Gradient*) algoritmu, mogu poboljšati pridržavanje zdravih prehrambenih smjernica i smanjiti unos visokokalorične hrane dinamičkim prilagođavanjem preporuka[[23]](#footnote-23). Kombinacija RL-a s kolaborativnim filtriranjem (CF) može dodatno poboljšati preporuke učenjem iz skrivenih navika korisnika i povratnih informacija, što dovodi do veće prihvaćenosti i zadovoljstva korisnika[[24]](#footnote-24). Ovaj pristup predstavlja značajan napredak u odnosu na statične sustave i postavlja temelje za prijedloge budućeg razvoja u poglavlju 6.

## Veliki jezični modeli u konverzacijskoj umjetnoj inteligenciji

Konverzacijska umjetna inteligencija doživjela je revoluciju s pojavom velikih jezičnih modela (engl. *Large Language Models*, LLM). Ovi modeli, temeljeni na **Transformer arhitekturi[[25]](#footnote-25)**, trenirani su na ogromnim količinama tekstualnih podataka i sposobni su razumjeti i generirati ljudski jezik s visokom razinom koherentnosti i relevantnosti. Ključna inovacija Transformer arhitekture je mehanizam **samo-pažnje (engl. *self-attention*)**, koji omogućuje modelu da ponderira važnost različitih riječi u ulaznom nizu, neovisno o njihovoj udaljenosti, čime se efikasno rješava problem dugoročnih ovisnosti koji je bio prisutan u ranijim rekurentnim arhitekturama[[26]](#footnote-26).

### 

* **Arhitektura:** Zephyr-7B-beta je model sa 7.24 milijarde parametara temeljen na arhitekturi transformera. Njegova osnova je model mistralai/Mistral-7B-v0.1, koji je poznat po svojoj učinkovitosti[[27]](#footnote-27). Arhitektura uključuje napredne mehanizme poput *Sliding Window Attention* i *Grouped Query Attention*, koji mu omogućuju obradu dužih sekvenci teksta uz manju potrošnju resursa[[28]](#footnote-28).
* **Proces treniranja:** Ono što Zephyr-7B-beta čini posebno pogodnim za ulogu asistenta jest metoda finog podešavanja (engl. *fine-tuning*). Nakon početnog treniranja, model je dodatno usklađen s ljudskim preferencijama korištenjem tehnike **optimizacije izravnih preferencija (engl. *Direct Preference Optimization*, DPO)**[[29]](#footnote-29). DPO predstavlja stabilniju i računski efikasniju alternativu tradicionalnom učenju s potkrepljenjem iz ljudske povratne veze (RLHF). Umjesto treniranja zasebnog modela nagrade, DPO izravno optimizira jezični model na temelju preferencija, tretirajući proces kao jednostavan klasifikacijski problem[[30]](#footnote-30).

**Mogućnosti i ograničenja:** Zephyr-7B-beta pokazuje snažne performanse u zadacima poput odgovaranja na pitanja, generiranja teksta i općenite konverzacije[[31]](#footnote-31). Međutim, važno je biti svjestan njegovih ograničenja. Kao i većina LLM-ova, može generirati činjenično netočne ili pristrane informacije. Nije eksplicitno treniran za sigurnost na isti način kao komercijalni modeli poput ChatGPT-a, što znači da može proizvesti problematične odgovore ako ga se na to potakne[[32]](#footnote-32). Također, u svojoj osnovnoj formi, model nema pristup vanjskim, real-time podacima niti specifičnom kontekstu korisnika iz baze podataka aplikacije, što ograničava stupanj personalizacije njegovih odgovora.

## Pregled i analiza srodnih rješenja

Tržište mobilnih i web aplikacija za praćenje fitnessa i prehrane je zasićeno, no većina rješenja fokusira se na pojedinačne aspekte zdravlja. Aplikacije poput MyFitnessPal ili Cronometer izvrsne su za detaljno praćenje unosa kalorija i makronutrijenata, dok se aplikacije poput Strava ili Nike Training Club fokusiraju primarno na praćenje tjelesne aktivnosti.

Analiza postojećih akademskih i stručnih radova otkriva trend razvoja integriranih sustava. Na primjer, rad Ražova[[33]](#footnote-33) opisuje izradu Android aplikacije koja prati napredak putem pametnih narukvica, fokusirajući se na prikupljanje biometrijskih podataka za trenere. S druge strane, rad autora Babića[[34]](#footnote-34) opisuje web aplikaciju koja omogućuje trenerima unos podataka o klijentima i predlaže vježbe, što je slično inicijalnoj funkcionalnosti ove aplikacije. Neka rješenja, poput onog opisanog u radu Golje[[35]](#footnote-35), idu korak dalje i nude cjelovitu platformu za komunikaciju između trenera i klijenta, izradu planova prehrane i vježbanja te praćenje napretka, čime se naglašava važnost interaktivnosti.

Uspoređujući se s tim rješenjima, aplikacija koja je predmet ovog rada ističe se po nekoliko ključnih značajki. Prvo, nudi holistički pristup koji na jednom mjestu objedinjuje praćenje vježbanja, prehrane, unosa vode i mentalnog stanja (raspoloženja). Drugo, i najvažnije, integrira modernog konverzacijskog AI asistenta kao središnji element korisničke podrške, što predstavlja inovativan pristup u usporedbi s tradicionalnim, statičnim aplikacijama. Ipak, analiza postojećih rješenja također ukazuje na područja za poboljšanje, prvenstveno u domeni dinamičnosti i adaptivnosti sustava preporuka, što će biti detaljnije obrađeno u kasnijim poglavljima.

# Dizajn i arhitektura sustava

U ovom poglavlju detaljno će se opisati arhitektonska struktura razvijene web aplikacije, model podataka koji čini temelj njezine funkcionalnosti, kao i proces prikupljanja i pripreme podataka korištenih za inicijalizaciju sustava. Jasan i dobro strukturiran dizajn preduvjet je za razvoj održivog i skalabilnog softverskog rješenja.

## Cjelokupna arhitektura aplikacije

Aplikacija je dizajnirana prema provjerenom modelu troslojne arhitekture (engl. *three-tier architecture*), koja logički razdvaja komponente sustava prema njihovim zaduženjima. Ovakav pristup poboljšava modularnost, olakšava održavanje i omogućuje neovisan razvoj pojedinih slojeva. Slojevi arhitekture su:

1. **Klijentski sloj (engl. *Presentation Layer*):** Ovaj sloj je ono što korisnik vidi i s čime izravno interagira. Čine ga HTML predlošci koji se renderiraju u korisnikovom web pregledniku. Korištenjem Jinja2 sustava predložaka, ovaj sloj dinamički prikazuje podatke koje mu prosljeđuje poslužiteljski sloj i šalje korisničke unose natrag na obradu.
2. **Poslužiteljski sloj (engl. *Business Logic Layer*):** Srce aplikacije, implementirano pomoću radnog okvira Flask. Ovaj sloj je odgovoran za obradu HTTP zahtjeva s klijentskog sloja, implementaciju cjelokupne poslovne logike (npr. registracija korisnika, generiranje preporuka), upravljanje korisničkim sesijama i komunikaciju sa slojem podataka.
3. **Sloj podataka (engl. *Data Layer*):** Sastoji se od SQLite baze podataka, koja služi za trajno pohranjivanje svih podataka aplikacije, kao što su korisnički profili, zapisi o vježbama i obrocima. Interakcija s bazom podataka odvija se isključivo putem SQLAlchemy ORM-a, što apstrahira SQL upite i osigurava konzistentan pristup podacima iz poslužiteljskog sloja.

Struktura direktorija projekta organizirana je na način da podržava modularnost i primjenu obrasca "aplikacijske tvornice"[[36]](#footnote-36):

* run.py: Ulazna točka za pokretanje aplikacije. Njegova jedina svrha je importirati i pokrenuti aplikacijsku tvornicu.
* app/: Glavni paket koji sadrži svu logiku aplikacije.
* \_\_init\_\_.py: Sadrži aplikacijsku tvornicu create\_app(), gdje se inicijalizira Flask aplikacija, konfiguriraju ekstenzije (SQLAlchemy, Bcrypt, LoginManager) i registriraju blueprinti.
* routes.py: Definira sve URL rute aplikacije i povezuje ih s odgovarajućim funkcijama za obradu zahtjeva (kontrolerima).
* models.py: Sadrži definicije svih SQLAlchemy modela, koji predstavljaju tablice u bazi podataka.
* services.py: Sadrži odvojenu poslovnu logiku, poput funkcija za generiranje planova vježbanja i preporuka obroka. Ovo razdvajanje logike od ruta poboljšava čitljivost i održivost koda.
* templates/: Direktorij koji sadrži sve HTML predloške (korisničko sučelje).
* static/: Direktorij za statičke datoteke poput CSS stilova, JavaScript datoteka i slika.

Primjer koda za run.py je:

from app import create\_app

app = create\_app()

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Ključni dio inicijalizacije nalazi se u funkciji create\_app unutar datoteke app/\_\_init\_\_.py:

def create\_app(config\_class=Config):

app = Flask(\_\_name\_\_)

app.config.from\_object(config\_class)

db.init\_app(app)

bcrypt.init\_app(app)

login\_manager.init\_app(app)

from app.routes import main\_bp

app.register\_blueprint(main\_bp)

with app.app\_context():

db.create\_all()

return app

## Model podataka i dizajn baze podataka

Model podataka definira strukturu, atribute i odnose između različitih entiteta unutar aplikacije. U ovom projektu, model podataka implementiran je kroz skup SQLAlchemy klasa u datoteci models.py. Svaka klasa preslikava se na jednu tablicu u SQLite bazi podataka.

U nastavku slijedi detaljan opis svake tablice i njezinih atributa.

* **User (tablica user):** Predstavlja korisnika aplikacije. Sadrži osnovne podatke za prijavu, kao i demografske i fitness podatke koji su ključni za personalizaciju.
* **Exercise (tablica exercise):** Sadrži katalog dostupnih vježbi s njihovim karakteristikama. Ovi podaci se koriste za generiranje planova vježbanja.
* **WorkoutLog (tablica workout\_log):** Zapis o pojedinačnom treningu koji je korisnik odradio. Povezan je s korisnikom preko vanjskog ključa user\_id.
* **MealLog (tablica meal\_log):** Zapis o obroku koji je korisnik unio. Također je povezan s korisnikom.
* **MoodLog (tablica mood\_log):** Zapis o korisnikovom raspoloženju za određeni dan.
* **WaterLog (tablica water\_log):** Zapis o dnevnom unosu vode.
* **ProgressReport (tablica progress\_report):** Pohranjuje generirane tjedne ili mjesečne izvještaje o napretku korisnika.
* **FoodItem (tablica food\_item):** Sadrži nutritivne informacije o različitim namirnicama, preuzete iz USDA baze podataka.

Primjer definicije User i WorkoutLog modela:

# app/models.py  
  
class User(UserMixin, db.Model):  
    id = db.Column(db.Integer, primary\_key=True)  
    username = db.Column(db.String(150), nullable=False, unique=True)  
    email = db.Column(db.String(150), nullable=False, unique=True)  
    password\_hash = db.Column(db.String(128), nullable=False)  
    age = db.Column(db.Integer)  
    gender = db.Column(db.String(10))  
    #... ostali atributi  
  
    def set\_password(self, password):  
        self.password\_hash = bcrypt.generate\_password\_hash(password).decode('utf-8')  
  
    def check\_password(self, password):  
        return bcrypt.check\_password\_hash(self.password\_hash, password)  
  
class WorkoutLog(db.Model):  
    id = db.Column(db.Integer, primary\_key=True)  
    user\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)  
    exercise = db.Column(db.String(150))  
    date = db.Column(db.Date, default=datetime.utcnow)  
    reps = db.Column(db.Integer)  
    sets = db.Column(db.Integer)  
    weight = db.Column(db.Float)  
    feeling = db.Column(db.String(100))

Primarni obrazac odnosa u bazi podataka je **jedan-prema-više (One-to-Many)**. Jedan korisnik (User) može imati više zapisa o treninzima (WorkoutLog), obrocima (MealLog), raspoloženjima (MoodLog), unosu vode (WaterLog) i izvještajima (ProgressReport). Ovi odnosi su implementirani pomoću vanjskih ključeva (user\_id) u tablicama na "više" strani, koji referenciraju primarni ključ (id) u tablici user.

Tablica 3.1: Shema baze podataka aplikacije

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Naziv tablice | Naziv atributa | Tip podatka | Ograničenja | Opis |
| user | id | Integer | Primarni ključ | Jedinstveni identifikator korisnika |
|  | username | String(150) | Not Null, Unique | Korisničko ime |
|  | email | String(150) | Not Null, Unique | E-mail adresa korisnika |
|  | password\_hash | String(128) | Not Null | Hashirana lozinka korisnika |
|  | age, gender, height, weight | Integer, String, Float, Float | - | Demografski i fizički podaci |
|  | goal, fitness\_level | String(50) | - | Fitness cilj i razina spremnosti |
| exercise | id | Integer | Primarni ključ | Jedinstveni identifikator vježbe |
|  | exercise\_name | String(150) | Not Null | Naziv vježbe |
|  | body\_part\_targeted | String(100) | - | Ciljana mišićna skupina |
|  | equipment\_needed | String(100) | - | Potrebna oprema |
|  | difficulty | String(50) | - | Težina izvođenja vježbe |
| workout\_log | id | Integer | Primarni ključ | Jedinstveni identifikator zapisa |
|  | user\_id | Integer | Foreign Key (user.id), Not Null | Poveznica na korisnika |
|  | exercise | String(150) | - | Naziv odrađene vježbe |
|  | date | Date | Default: now | Datum treninga |
|  | reps, sets, weight | Integer, Integer, Float | - | Detalji o izvedbi vježbe |
| meal\_log | id | Integer | Primarni ključ | Jedinstveni identifikator zapisa |
|  | user\_id | Integer | Foreign Key (user.id), Not Null | Poveznica na korisnika |
|  | food | String(200) | - | Naziv konzumirane hrane |
|  | calories | Float | - | Kalorijska vrijednost obroka |
| food\_item | id | Integer | Primarni ključ | Jedinstveni identifikator namirnice |
|  | name | String(200) | Not Null | Naziv namirnice |
|  | calories, protein, fat, carbs | Float, Float, Float, Float | - | Nutritivne vrijednosti |

## Korišteni skupovi podataka

Za inicijalno punjenje baze podataka i osiguravanje bogatog sadržaja za korisnike od samog početka, korišteno je nekoliko javno dostupnih skupova podataka:

* **USDA.csv:** Službeni skup podataka Ministarstva poljoprivrede Sjedinjenih Američkih Država (USDA) sadrži detaljne nutritivne informacije za tisuće namirnica. Iz ovog skupa podataka preuzeti su podaci o kalorijama, proteinima, mastima i ugljikohidratima za popunjavanje tablice FoodItem.[[37]](#footnote-37)
* **Gym\_exercise\_dataset.xlsx:** Ovaj skup podataka, prikupljen s platforme Kaggle, sadrži strukturirane informacije o različitim vježbama, uključujući njihov naziv, ciljanu mišićnu skupinu, potrebnu opremu i razinu težine. Poslužio je kao izvor za popunjavanje tablice Exercise.[[38]](#footnote-38)
* **recipes.csv:** Skup podataka s receptima, također s Kagglea. Skup podataka koji se koristi za preporuku recepata na osnovu korisničkih potreba.[[39]](#footnote-39)
* **emotions.csv:** Manji skup podataka koji mapira tekstualne opise osjećaja na određene kategorije emocija. Može se koristiti za standardizaciju unosa u tablicu MoodLog.[[40]](#footnote-40)

## Proces pripreme i unosa podataka (ETL)

Kako bi se podaci iz vanjskih izvora (CSV i Excel datoteka) unijeli u strukturiranu SQLite bazu podataka, implementiran je ETL (engl. *Extract, Transform, Load*) proces. Ovaj proces je realiziran kroz Python skripte koje koriste biblioteku pandas za obradu podataka.

Analiza skripte a\_11\_populate\_usda\_db.py otkriva jasan i robustan ETL cjevovod za nutritivne podatke:

1. **Extract (Ekstrakcija):** Skripta prvo učitava podatke iz USDA.csv datoteke u pandas DataFrame. Uključena je i obrada grešaka za slučaj da datoteka ne postoji.
2. **Transform (Transformacija):** Slijedi nekoliko koraka čišćenja i transformacije podataka kako bi odgovarali shemi FoodItem tablice:

* Uklanjaju se redovi s nedostajućim vrijednostima u ključnim stupcima (Description, Calories, Protein itd.).
* Stupci iz CSV datoteke preimenuju se kako bi odgovarali nazivima atributa u FoodItem modelu (npr. Description postaje name, TotalFat postaje fat).
* Uklanjaju se duplicirani unosi na temelju naziva namirnice kako bi se osigurala jedinstvenost.

1. **Load (Učitavanje):** Očišćeni i transformirani podaci se zatim učitavaju u food\_item tablicu baze podataka. Za ovaj korak koristi se visoko učinkovita metoda pandas.DataFrame.to\_sql, koja omogućuje masovni unos podataka (engl. *bulk insert*) u manjim serijama (*chunks*), što optimizira performanse. Prije unosa, skripta briše sve postojeće podatke iz tablice kako bi se izbjeglo dupliciranje prilikom ponovnog pokretanja.

Isječak koda iz a\_11\_populate\_usda\_db.py koji prikazuje ovaj proces:

# a\_11\_populate\_usda\_db.py

def populate\_food\_items():

try:

df = pd.read\_csv(CSV\_PATH)

required\_columns =

df.dropna(subset=required\_columns, inplace=True)

df.rename(columns={

'Description': 'name',

'Calories': 'calories',

'Protein': 'protein',

'TotalFat': 'fat',

'Carbohydrate': 'carbs'

}, inplace=True)

df\_to\_load = df[['name', 'calories', 'protein', 'fat', 'carbs']]

df\_to\_load.drop\_duplicates(subset=['name'], inplace=True)

df\_to\_load.to\_sql('food\_item', db.engine, if\_exists='append', index=False, chunksize=1000)

print(f"Successfully populated {len(df\_to\_load)} food items.")

except FileNotFoundError:

print(f"Error: {CSV\_PATH} not found.")

except Exception as e:

print(f"An error occurred: {e}")

Sličan proces primijenjen je i za podatke o vježbama iz Gym\_exercise\_dataset.xlsx datoteke, gdje se podaci čitaju, mapiraju na atribute Exercise modela i unose u odgovarajuću tablicu.

# Implementacija ključnih funkcionalnosti

Ovo poglavlje pruža detaljan uvid u tehničku implementaciju središnjih komponenti aplikacije. Fokus je na analizi programskog koda koji definira ponašanje sustava, od inicijalizacije i upravljanja korisnicima do realizacije korisničkog sučelja i inteligentnih funkcionalnosti.

## Korisničko sučelje (UI)

Korisničko sučelje (UI) realizirano je kroz skup HTML datoteka u templates direktoriju, koje se dinamički renderiraju pomoću Jinja2 sustava predložaka. Korišten je Bootstrap 5 radni okvir za responzivni dizajn i stiliziranje komponenti, što osigurava prilagodljivost sučelja različitim veličinama ekrana.

### Registracija korisnika (register.html)

Stranica za registraciju ključna je za prikupljanje početnih podataka potrebnih za personalizaciju. Forma je strukturirana tako da prikupi sve atribute definirane u User modelu.

* **Struktura:** Forma je podijeljena u logičke cjeline: osnovni podaci (korisničko ime, email, lozinka), demografski podaci (dob, spol, visina, težina) i fitness podaci (cilj, razina spremnosti, dostupna oprema).
* **Interaktivnost:** Korišteni su JavaScript i Bootstrap Tooltips kako bi se korisniku pružile dodatne informacije i objašnjenja za svako polje, npr. što znači koja razina fitnessa ili kakav plan može očekivati za odabrani cilj.
* **Unos podataka:** Koriste se standardni HTML elementi forme (<input>, <select>, <textarea>) s odgovarajućim tipovima (email, password, number) kako bi se osigurala osnovna validacija na strani klijenta.

Slika 4.1: Vizualni prikaz register.html

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Sadržaj generiran uz AI možda nije točan.

### Nadzorna ploča (dashboard.html)

Nadzorna ploča je središnja točka interakcije za prijavljenog korisnika. Dizajnirana je kao dinamičan i interaktivan prostor koji pruža pregled ključnih informacija i brz pristup glavnim funkcionalnostima.

* **Personalizirani pozdrav:** Na vrhu stranice nalazi se pozdravna poruka koja dinamički prikazuje korisničko ime ({{ user.username }}) i njegov odabrani cilj.
* **AI Trener Chat:** Istaknuti dio sučelja je poziv na akciju za korištenje "AI Trener" chata, koji korisnika vodi na /smart\_input stranicu.
* **Prikaz planova i preporuka:** Ako su u korisničkoj sesiji pohranjeni generirani plan vježbanja ili preporuke za obroke, oni se dinamički prikazuju u obliku kartica. Plan vježbanja je organiziran po danima, a svaka vježba je poveznica na video s uputama.
* **Brzi unos:** U desnom stupcu nalaze se forme za brzi unos treninga i obroka, omogućujući korisniku da bez napuštanja nadzorne ploče zabilježi svoje aktivnosti.
* **Pregled aktivnosti:** Na dnu desnog stupca prikazuju se liste nedavno unesenih treninga i obroka, dohvaćene iz baze podataka.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, web-stranica

Sadržaj generiran uz AI možda nije točan.

Slika 4.2: Vizualni prikakaz dashboard.html

### AI Virtualni Trener (smart\_input.html)

Ovo sučelje je dizajnirano kao klasična aplikacija za chat, pružajući intuitivno i prirodno iskustvo komunikacije s AI asistentom.

* **Struktura:** Sučelje se sastoji od zaglavlja koje objašnjava funkcionalnost, glavnog prozora za prikaz poruka i polja za unos na dnu.
* **Prikaz povijesti razgovora:** Povijest razgovora se dinamički renderira pomoću Jinja2 for petlje koja iterira kroz listu poruka pohranjenu u sesiji. Poruke korisnika (role == 'user') i asistenta (role == 'assistant') imaju različite stilove kako bi se vizualno razlikovale.
* **Interakcija:** Korisnik unosi tekst u polje za unos i šalje ga POST metodom na istu /smart\_input rutu. Stranica se zatim ponovno učitava s ažuriranom poviješću razgovora.
* **Upravljanje chatom:** Postoje gumbi za povratak na nadzornu ploču i za brisanje cjelokupne povijesti razgovora, što korisniku daje potpunu kontrolu nad interakcijom.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, web-stranica

Sadržaj generiran uz AI možda nije točan.

Slika 4.3: Vizualni prikaz smart\_input.html

### Tjedni izvještaj (weekly\_report.html)

Stranica tjednog izvještaja služi za vizualizaciju i sumiranje korisnikovog napretka tijekom proteklih sedam dana. Cilj je pružiti korisniku jasan i sažet pregled njegovih postignuća i trendova.

Struktura: Stranica je podijeljena na tri glavna dijela: ključne metrike, personalizirani uvidi i status napretka prema cilju.

Ključne metrike: Na vrhu stranice, u obliku istaknutih kartica (stat-card), prikazuju se agregirani podaci za protekli tjedan: ukupan broj odrađenih treninga, prosječan dnevni unos kalorija, prosječna ocjena raspoloženja i prosječan dnevni unos vode. Ovi podaci se dohvaćaju iz generate\_weekly\_report servisa.

Personalizirani uvidi: Ispod ključnih metrika nalazi se sekcija koja prikazuje tekstualne uvide (insight-card) generirane na temelju analize tjednih podataka. Ovi uvidi mogu sadržavati pohvale, sugestije za poboljšanje ili uočene korelacije (npr. "Primijetili smo da je vaše raspoloženje bolje u danima kada trenirate.").

Napredak prema cilju: Na dnu stranice nalazi se dinamički blok koji pruža sažetak napretka u kontekstu korisnikovog glavnog cilja (gubitak težine, gradnja mišića, održavanje). Prikazuje se poruka koja ovisi o tome jesu li korisnikove aktivnosti u skladu s preporukama za taj cilj.

Slika 4.4: Vizualni prikaz wee Weekly\_report.html

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Sadržaj generiran uz AI možda nije točan.

### Demografski uvidi (demographic\_insights.html)

Ova stranica pruža korisniku kontekstualne informacije uspoređujući njegove navike s prosječnim vrijednostima za njegovu demografsku skupinu (npr. ista dobna skupina i spol).

Struktura: Stranica se sastoji od niza kartica (card), od kojih svaka predstavlja jedan specifičan uvid.

Prikaz uvida: Svaka kartica sadrži naslov uvida, poruku koja objašnjava usporedbu (npr. "Vaš prosječni tjedni broj treninga je veći od prosjeka za vašu dobnu skupinu.") i konkretnu preporuku koja iz toga proizlazi (npr. "Sjajan posao! Razmislite o uvođenju naprednijih vježbi kako biste dodatno izazvali svoje tijelo.").

Dinamika: Sadržaj ovih uvida generira se u get\_demographic\_insights servisu, koji vjerojatno izvodi agregatne upite nad cjelokupnom bazom korisnika kako bi izračunao prosjeke za različite demografske segmente.

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.5: demographic\_insights.html

## Sustav za upravljanje korisnicima i autentikaciju

Upravljanje korisnicima je temeljna funkcionalnost svake personalizirane aplikacije. Logika je implementirana kroz nekoliko ruta u datoteci routes.py, uz podršku ekstenzija Flask-Login i Bcrypt.

* **Registracija (/register):**
* Kada korisnik pristupi ovoj ruti GET metodom, prikazuje mu se registracijska forma (register.html).
* Nakon slanja forme (POST metoda), funkcija provjerava postoje li korisničko ime i e-mail adresa već u bazi podataka kako bi se osigurala jedinstvenost.
* Ako su podaci jedinstveni, stvara se nova instanca User modela. Lozinka koju je korisnik unio ne sprema se izravno; umjesto toga, poziva se metoda set\_password() na User objektu, koja koristi Bcrypt za generiranje sigurnog, "slanog" hasha lozinke.
* Novi korisnik se dodaje u bazu podataka, a korisnik se preusmjerava na stranicu za prijavu.

# app/routes.py

@main\_bp.route('/register', methods=)

def register():

if request.method == 'POST':

#... (dohvaćanje podataka iz forme)

existing\_user = User.query.filter\_by(email=email).first()

if existing\_user:

flash('Email adresa već postoji.', 'danger')

return redirect(url\_for('main.register'))

new\_user = User(...)

new\_user.set\_password(password)

db.session.add(new\_user)

db.session.commit()

flash('Uspješno ste se registrirali! Možete se prijaviti.', 'success')

return redirect(url\_for('main.login'))

return render\_template('register.html')

* **Prijava (/login):**
* GET zahtjev prikazuje formu za prijavu (login.html).
* Nakon POST zahtjeva, funkcija pronalazi korisnika u bazi podataka prema unesenoj e-mail adresi.
* Ako korisnik postoji, poziva se metoda check\_password() koja uspoređuje unesenu lozinku s pohranjenim hashem.
* Ako je lozinka ispravna, poziva se funkcija login\_user(user) iz ekstenzije Flask-Login. Ova funkcija uspostavlja korisničku sesiju spremanjem korisničkog ID-a u kriptografski potpisan kolačić. Korisnik se zatim preusmjerava na nadzornu ploču (/dashboard).

Implementacija rute za prijavu:  
# app/routes.py  
  
@main\_bp.route('/login', methods=)  
def login():  
    if request.method == 'POST':  
        email = request.form.get('email')  
        password = request.form.get('password')  
        user = User.query.filter\_by(email=email).first()  
          
        if user and user.check\_password(password):  
            login\_user(user)  
            return redirect(url\_for('main.dashboard'))  
        else:  
            flash('Prijava neuspješna. Provjerite email i lozinku.', 'danger')  
    return render\_template('login.html')

* **Odjava (/logout):**
* Ova ruta zahtijeva da korisnik bude prijavljen (zaštićena dekoratorom @login\_required).
* Poziva funkciju logout\_user() koja briše korisničku sesiju i preusmjerava korisnika na početnu stranicu.
* Tablica 2 daje sustavan pregled svih definiranih URL ruta i njihovih funkcionalnosti.

Tablica 4.1: Pregled ruta aplikacije

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL ruta | HTTP metode | Naziv funkcije | Opis funkcionalnosti | Potrebna autentikacija |
| / | GET | home() | Prikazuje početnu stranicu aplikacije. | Ne |
| /register | GET, POST | register() | Prikazuje i obrađuje formu za registraciju novih korisnika. | Ne |
| /login | GET, POST | login() | Prikazuje i obrađuje formu za prijavu korisnika. | Ne |
| /logout | GET | logout() | Odjavljuje trenutno prijavljenog korisnika. | Da |
| /dashboard | GET | dashboard() | Prikazuje glavnu nadzornu ploču za prijavljenog korisnika. | Da |
| /generate\_plan | POST | generate\_plan() | Generira novi plan vježbanja za korisnika. | Da |
| /get\_meals | POST | get\_meals() | Generira preporuke za obroke za korisnika. | Da |
| /smart\_input | GET, POST | smart\_input() | Prikazuje i obrađuje interakciju s AI asistentom. | Da |
| /clear\_smart\_chat | POST | clear\_smart\_chat() | Briše povijest razgovora s AI asistentom. | Da |
| /reports/weekly | GET | weekly\_report() | Generira i prikazuje tjedni izvještaj o napretku. | Da |
| /insights/demographics | GET | demographic\_insights() | Prikazuje demografske uvide i usporedbe. | Da |
| /log\_workout | POST | log\_workout() | Obrađuje unos podataka o odrađenom treningu. | Da |
| /log\_meal | POST | log\_meal() | Obrađuje unos podataka o konzumiranom obroku. | Da |

## Implementacija sustava za generiranje preporuka

Logika za generiranje preporuka smještena je u datoteku services.py kako bi bila odvojena od logike ruta. Kao što je objašnjeno u teorijskom dijelu, ova implementacija funkcionira kao **ekspertni sustav temeljen na pravilima**.

Pretpostavljeni algoritam funkcije generate\_workout\_plan je sljedeći:

1. **Dohvaćanje činjenica:** Funkcija prima objekt prijavljenog korisnika i iz njega dohvaća ključne atribute (činjenice): goal, fitness\_level i equipment.
2. **Primjena pravila (filtriranje):** Funkcija izvršava upit nad Exercise tablicom u bazi podataka. Upit sadrži niz if/elif/else uvjeta koji djeluju kao pravila:

* difficulty vježbe mora odgovarati korisnikovom fitness\_level.
* equipment\_needed vježbe mora biti u skladu s opremom koju korisnik posjeduje.
* Ciljana mišićna skupina (body\_part\_targeted) odabire se na temelju korisnikovog cilja (goal) i/ili dana u tjednu (npr. ponedjeljak za prsa, srijeda za noge).

1. **Generiranje zaključka:** Iz filtriranog skupa vježbi, sustav nasumično odabire određeni broj vježbi i sastavlja strukturirani plan (npr. u obliku rječnika ili liste), koji se zatim vraća ruti /dashboard za prikaz.

Tablica 4.2: Vizualni prikaz generiranja vježbi

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Sadržaj generiran uz AI možda nije točan.

Primjer implementacije u services.py:

# app/services.py

def generate\_workout\_plan(user):

#... (logika za definiranje dana i mišićnih skupina)

plan = {}

for day, body\_parts in schedule.items():

if body\_parts == ["Odmor"]:

plan[day] = ["Odmor"]

continue

query = Exercise.query.filter(Exercise.body\_part\_targeted.in\_(body\_parts))

# Pravila na temelju opreme

if user.equipment == 'gym':

# Sva oprema je dostupna

pass

elif user.equipment == 'home\_dumbbells':

query = query.filter(Exercise.equipment\_needed.in\_())

else: # bodyweight

query = query.filter\_by(equipment\_needed='Bodyweight')

# Pravila na temelju razine fitnessa

if user.fitness\_level == 'beginner':

query = query.filter\_by(difficulty='Beginner')

#... (ostali uvjeti)

exercises = query.limit(5).all()

plan[day] = [{'name': e.exercise\_name, 'link': e.link} for e in exercises]

return plan

Slično tome, funkcija get\_meal\_recommendations slijedi sličan obrazac:

1. **Dohvaćanje činjenica:** Funkcija prima korisnikov ciljani dnevni unos kalorija i dosad unesene kalorije.
2. **Primjena pravila:** Izračunava se preostali kalorijski budžet za dan.
3. **Generiranje zaključka:** Sustav pretražuje FoodItem tablicu za namirnice ili obroke čija kalorijska vrijednost odgovara preostalom budžetu, uzimajući u obzir eventualne dodatne preferencije. Vraća se lista preporučenih obroka.

## Implementacija inteligentnog asistenta i analize emocija

Središnja inovativna komponenta aplikacije je inteligentni asistent. Njegova implementacija ne obuhvaća samo generiranje odgovora, već i potencijalnu analizu korisnikovih emocija.

### Konverzacijski AI Virtualni Trener

Interakcija s AI asistentom odvija se kroz rutu /smart\_input. Logika upravljanja konverzacijom implementirana je na sljedeći način:

1. **Upravljanje poviješću:** Povijest razgovora čuva se u korisničkoj sesiji (session['chat\_history']). Pri prvom posjetu stranici, sesija se inicijalizira s početnom sistemskom porukom koja definira personu AI asistenta (npr. "Ti si koristan fitness asistent.").
2. **Obrada korisničkog upita:** Kada korisnik pošalje poruku (POST zahtjev), njegov upit se dodaje u listu chat\_history.
3. **Komunikacija s modelom:** Cjelokupna chat\_history lista šalje se Hugging Face InferenceClient-u, koji prosljeđuje podatke modelu HuggingFaceH4/zephyr-7b-beta. Slanje cijele povijesti omogućuje modelu da zadrži kontekst razgovora.
4. **Prikaz odgovora:** Odgovor koji model generira dodaje se u chat\_history i sesija se ažurira. Korisnik se preusmjerava natrag na /smart\_input stranicu (GET zahtjev), gdje se renderira ažurirana povijest razgovora.
5. **Održavanje konteksta:** Kako bi se spriječilo preopterećenje i prekoračenje limita konteksta modela, implementirana je logika koja u sesiji čuva samo zadnjih 10 interakcija (10 korisničkih poruka i 10 odgovora modela), uz početnu sistemsku poruku.

# app/routes.py

@main\_bp.route('/smart\_input', methods=)

@login\_required

def smart\_input():

if 'chat\_history' not in session:

session['chat\_history'] = [{"role": "system", "content": "You are a helpful fitness assistant."}]

if request.method == 'POST':

user\_input = request.form.get('description')

session['chat\_history'].append({"role": "user", "content": user\_input})

try:

response = hf\_client.chat\_completion(

model="HuggingFaceH4/zephyr-7b-beta",

messages=session['chat\_history'],

max\_tokens=500,

)

ai\_response = response.choices.message.content

session['chat\_history'].append({"role": "assistant", "content": ai\_response})

# Ograničavanje povijesti na zadnjih 10 interakcija

if len(session['chat\_history']) > 21: # 1 sistemska + 10\*2 korisnik/asistent

session['chat\_history'] = [session['chat\_history']] + session['chat\_history'][-20:]

except Exception as e:

flash(f"Greška u komunikaciji s AI: {e}", "danger")

session.modified = True

return redirect(url\_for('main.smart\_input'))

return render\_template('smart\_input.html', chat\_history=session.get('chat\_history',))

### Analiza emocija

Funkcionalnost se implementira na sljedeći način:

1. **Učitavanje modela za klasifikaciju emocija:** Prilikom pokretanja aplikacije, mogao bi se učitati prethodno trenirani model za klasifikaciju teksta (npr. temeljen na BERT-u ili sličnoj arhitekturi), finetuniran na emotions.csv skupu podataka.
2. **Analiza unosa:** Kada korisnik unese tekstualni opis osjećaja u WorkoutLog ili unutar chata, taj tekst bi se proslijedio modelu za klasifikaciju.
3. **Strukturirani zapis:** Model bi vratio standardiziranu kategoriju emocije (npr. "sretan", "umoran", "motiviran"), koja bi se zatim mogla pohraniti u bazu podataka.
4. **Korištenje u preporukama:** Ovi strukturirani podaci o emocijama mogli bi postati dio **stanja** u naprednijem sustavu preporuka (kao što je opisano u poglavlju 6), omogućujući sustavu da prilagodi preporuke na temelju korisnikovog emocionalnog stanja (npr. preporučiti lakši trening ako se korisnik osjeća umorno).

# Sigurnosni aspekti aplikacije

Sigurnost web aplikacija je kritičan aspekt razvoja koji se ne smije zanemariti. Iako Flask kao mikro-okvir ne nameće mnoge sigurnosne mehanizme po zadanome, pruža temelje i ekosustav ekstenzija za implementaciju robusne zaštite[[41]](#footnote-41). U ovom poglavlju analiziraju se implementirane sigurnosne mjere, identificiraju potencijalni rizici i daju preporuke za sigurno produkcijsko okruženje, s osvrtom na OWASP Top 10 preporuke[[42]](#footnote-42).

## Zaštita korisničkih podataka i autentikacija

Zaštita korisničkih računa i osjetljivih podataka počinje sigurnom pohranom lozinki i pouzdanim upravljanjem sesijama, što direktno adresira OWASP kategoriju **A07:2021-Identification and Authentication Failures**.

* **Hashiranje lozinki:** Aplikacija ne pohranjuje korisničke lozinke u čistom tekstu. Umjesto toga, koristi se biblioteka Bcrypt, koja je inicijalizirana u app/\_\_init\_\_.py. Bcrypt je adaptivna kriptografska hash funkcija dizajnirana specifično za hashiranje lozinki. Ona je namjerno spora, što otežava *brute-force* napade na bazu podataka lozinki ako bi ona ikada bila kompromitirana. Također, automatski koristi "sol" (engl. *salt*), jedinstveni nasumični niz znakova za svaku lozinku prije hashiranja, što sprječava korištenje unaprijed izračunatih tablica (*rainbow tables*) za probijanje lozinki[[43]](#footnote-43).

Metode za upravljanje lozinkom u User modelu:

# app/models.py  
  
class User(UserMixin, db.Model):  
    #...  
    def set\_password(self, password):  
        self.password\_hash = bcrypt.generate\_password\_hash(password).decode('utf-8')  
  
    def check\_password(self, password):  
        return bcrypt.check\_password\_hash(self.password\_hash, password)

* **Upravljanje sesijama:** Za upravljanje korisničkim sesijama nakon prijave koristi se ekstenzija Flask-Login. Ova ekstenzija pohranjuje identifikator prijavljenog korisnika u kolačić na strani klijenta. Taj kolačić je kriptografski potpisan korištenjem aplikacijskog SECRET\_KEY-a. To osigurava da korisnik ne može neovlašteno mijenjati sadržaj kolačića (npr. promijeniti svoj ID u ID drugog korisnika) bez da potpis postane nevažeći. Ovo pruža snažnu zaštitu od neovlaštenog pristupa tuđim računima[[44]](#footnote-44).

## Prevencija uobičajenih web napada

Aplikacija je dizajnirana s ciljem prevencije najčešćih vektora napada na web aplikacije.

* **Cross-Site Scripting (XSS):** XSS napadi, dio OWASP kategorije **A03:2021-Injection**, događaju se kada napadač uspije ubaciti zlonamjerni skriptni kod (obično JavaScript) u web stranicu, koji se zatim izvršava u pregledniku drugih korisnika. Primarna linija obrane u ovoj aplikaciji je automatsko escapiranje podataka od strane Jinja2 sustava predložaka. Svaka varijabla koja se ispisuje u HTML-u pomoću sintakse {{... }} automatski se obrađuje tako da se specijalni znakovi poput <, > i & pretvaraju u njihove sigurne HTML entitete (npr. <, >, &). To sprječava da preglednik interpretira korisnički unos kao HTML ili JavaScript kod[[45]](#footnote-45).
* **Cross-Site Request Forgery (CSRF):** CSRF je napad kojim se korisnika koji je prijavljen na neku stranicu navodi da, bez svog znanja, izvrši neželjenu akciju na toj stranici. Analizom koda utvrđeno je da forme za registraciju i prijavu, koje su vjerojatno implementirane uz pomoć ekstenzije Flask-WTF, sadrže CSRF zaštitu. Međutim, forme za unos treninga (/log\_workout) i obroka (/log\_meal) su jednostavne HTML forme koje šalju POST zahtjeve. Ako te forme nisu eksplicitno zaštićene CSRF tokenom, predstavljaju značajan sigurnosni rizik. Ključno sigurnosno unapređenje bilo bi implementirati globalnu CSRF zaštitu, idealno korištenjem ekstenzije poput Flask-CSRFProtect ili refaktoriranjem svih formi da koriste Flask-WTF. To bi osiguralo da svaki zahtjev koji mijenja stanje aplikacije mora sadržavati jedinstveni, tajni token, čime se potvrđuje da je zahtjev potekao s legitimne stranice aplikacije[[46]](#footnote-46).

**SQL Injection:** Napadi SQL injekcijom, također dio **A03:2021-Injection**, događaju se kada napadač kroz polja za unos uspije ubaciti maliciozne SQL naredbe koje se zatim izvršavaju na bazi podataka. Korištenje SQLAlchemy ORM-a u ovoj aplikaciji pruža inherentnu zaštitu od ovog tipa napada. SQLAlchemy ne gradi SQL upite jednostavnim spajanjem stringova. Umjesto toga, koristi parametrizirane upite (engl. *parameterized queries*), gdje se SQL naredba i podaci šalju bazi podataka odvojeno. Baza podataka sama sigurno umeće podatke u upit, tretirajući ih uvijek kao vrijednosti, a nikada kao izvršni kod. Time se eliminira mogućnost SQL injekcije[[47]](#footnote-47).

## Preporuke za sigurno produkcijsko okruženje

Za postavljanje aplikacije u produkcijsko okruženje, potrebno je poduzeti dodatne korake kako bi se osigurala maksimalna sigurnost i performanse, adresirajući OWASP kategoriju **A05:2021-Security Misconfiguration**.

* **Isključivanje debug moda:** Apsolutno je nužno postaviti debug=False u produkciji. Ostavljanje debug moda uključenim izlaže aplikaciju ozbiljnim rizicima, uključujući mogućnost izvršavanja proizvoljnog koda putem interaktivnog debuggera[[48]](#footnote-48).
* **Upravljanje tajnama:** Osjetljivi podaci poput SECRET\_KEY i HF\_TOKEN nikada ne smiju biti tvrdo kodirani u izvornom kodu. Najbolja praksa je učitavati ih iz varijabli okruženja (engl. *environment variables*). To sprječava njihovo slučajno curenje putem sustava za kontrolu verzija (npr. Git) i olakšava upravljanje konfiguracijom u različitim okruženjima[[49]](#footnote-49).
* **Implementacija sigurnosnih HTTP zaglavlja:** Moderni web preglednici podržavaju niz HTTP zaglavlja koja mogu značajno poboljšati sigurnost. Preporučuje se korištenje ekstenzije poput Flask-Talisman za jednostavno postavljanje ovih zaglavlja[[50]](#footnote-50):
* **HTTP Strict Transport Security (HSTS):** Nalaže pregledniku da sa stranicom komunicira isključivo putem sigurne HTTPS veze.
* **Content Security Policy (CSP):** Definira iz kojih izvora je dopušteno učitavanje sadržaja (skripti, stilova, slika), što je dodatna, snažna linija obrane protiv XSS napada.
* **X-Frame-Options:** Sprječava da se stranica učita unutar <iframe> na drugim domenama, čime se štiti od *clickjacking* napada[[51]](#footnote-51).
* **X-Content-Type-Options:** Postavljanjem na nosniff sprječava se da preglednik pokuša "pogoditi" tip sadržaja, što može spriječiti neke vrste napada.

**Korištenje produkcijskog WSGI poslužitelja:** Ugrađeni razvojni poslužitelj Flaska nije dizajniran za produkcijsku upotrebu. U produkciji je potrebno koristiti robustan WSGI poslužitelj poput Gunicorna ili uWSGI-a, najčešće iza reverznog proksija kao što je Nginx.

# Rasprava i prijedlozi za budući rad

Nakon detaljne analize dizajna i implementacije, ovo poglavlje nudi kritički osvrt na postignuto rješenje, identificirajući njegove snage i slabosti. Na temelju te analize i teorijske podloge iz drugog poglavlja, predlažu se konkretna i značajna unapređenja koja bi aplikaciju mogla transformirati iz statičnog alata u dinamičan i adaptivan sustav..

## Prijedlog unapređenja sustava preporuka primjenom učenja s potkrepljenjem (RL)

Kako bi se prevladalo ograničenje statičnosti, predlaže se potpuna reimplementacija sustava preporuka korištenjem paradigme učenja s potkrepljenjem (RL). Aplikacija već prikuplja idealne podatke za treniranje takvog sustava: zapise o vježbama, obrocima, raspoloženju i, što je najvažnije, eksplicitnu povratnu informaciju kroz atribut liked\_recommendation u MealLog tablici.

Tranzicija na RL sustav zahtijevala bi formalizaciju problema kao **Markovljevog procesa odlučivanja (engl. *Markov Decision Process*, MDP)**, gdje bi komponente bile definirane na sljedeći način[[52]](#footnote-52):

* **Stanje (State):** Stanje bi predstavljalo sveobuhvatan "snimak" korisnika u trenutku donošenja preporuke. Bio bi to vektor sastavljen od:
* **Statičkih korisničkih podataka:** Atributi iz User modela (age, gender, goal, fitness\_level).
* **Dinamičke povijesti interakcija:** Agregirani podaci iz zadnjih N dana, npr. prosječan unos kalorija (MealLog), ukupni volumen treninga (WorkoutLog), dominantno raspoloženje (MoodLog).
* **Kontekstualnih informacija:** Dodatni podaci poput dana u tjednu ili doba dana.
* **Akcija (Action):** Akcija više ne bi bila samo jedna preporuka, već odabir jedne od mogućih preporuka iz skupa kandidata. Na primjer, akcija bi mogla biti "preporuči plan A", "preporuči plan B" ili "preporuči plan C", gdje svaki plan predstavlja različitu kombinaciju vježbi ili obroka.
* **Nagrada (Reward):** Ključ uspjeha RL sustava leži u pažljivom dizajniranju funkcije nagrade. Ona mora kvantificirati koliko je preporuka bila "dobra". Funkcija nagrade mogla bi biti kompozitna[[53]](#footnote-53):
* **Eksplicitna povratna veza:** Visoka pozitivna nagrada (npr. +1) ako korisnik označi liked\_recommendation=True ili ocijeni preporuku s 5 zvjezdica.
* **Implicitna pozitivna veza:** Manja pozitivna nagrada (npr. +0.5) ako korisnik izvrši preporučeni trening i zabilježi pozitivan feeling (npr. "snažno", "energično").
* **Implicitna negativna veza:** Negativna nagrada (npr. -1) ako korisnik ignorira preporuku (npr. ne unese nikakav trening ili obrok unutar 24 sata od preporuke).
* **Dugoročna nagrada:** Značajna pozitivna nagrada (npr. +5) kada se korisnikovi podaci (npr. weight) približe definiranoj vrijednosti u goal.

Za implementaciju bi se mogao koristiti neki od *off-policy* algoritama, poput **Deep Q-Networks (DQN)**, koji su pogodni za učenje iz pohranjenih podataka o interakcijama (iskustava)[[54]](#footnote-54). Treniranje bi se moglo provoditi periodički (npr. jednom tjedno) na svim prikupljenim podacima, čime bi se model kontinuirano poboljšavao i prilagođavao svakom pojedinom korisniku.

## Unapređenje povratne veze i integracija AI asistenta

Paralelno s razvojem RL sustava, mogu se implementirati i druga značajna unapređenja.

* **Kontekstualno svjestan AI asistent:** Funkcionalnost AI asistenta može se drastično poboljšati obogaćivanjem prompta koji se šalje modelu. Prije slanja korisnikovog pitanja, aplikacija bi mogla automatski dodati sažetak relevantnih podataka iz baze u prompt. Na primjer, prompt bi mogao izgledati ovako: "Kontekst: Korisnikov cilj je 'gubitak težine'. Njegova trenutna težina je 85 kg. Jučer je odradio trening za noge i osjećao se 'umorno'. Danas je do sada unio 1200 kalorija. Korisnik pita: 'Što da jedem za večeru i trebam li danas vježbati?'". Ovako obogaćen prompt omogućio bi Zephyr-7B-beta modelu da pruži daleko relevantniji i personaliziraniji odgovor, pretvarajući ga iz općenitog sugovornika u pravog osobnog asistenta[[55]](#footnote-55).
* **Poboljšanje mehanizama povratne veze:** Trenutni mehanizam povratne veze je binaran (liked\_recommendation). Uvođenje granularnijeg sustava, poput ocjenjivanja preporuka na skali od 1 do 5, pružilo bi bogatiji i informativniji signal za učenje RL agenta. Korisnici bi također mogli ostaviti kratke tekstualne komentare, čijom bi se analizom sentimenta mogao dobiti dodatni ulaz za funkciju nagrade[[56]](#footnote-56).

Ovi prijedlozi, posebice tranzicija na RL, predstavljaju značajan istraživački i razvojni korak. Njihovom implementacijom, aplikacija bi se transformirala u inteligentan sustav koji uistinu uči od svojih korisnika i s vremenom postaje sve bolji partner u ostvarenju njihovih zdravstvenih i fitness ciljeva.

# Zaključak

Ovaj diplomski rad detaljno je obradio proces razvoja i implementacije inteligentne web aplikacije za personalizirano praćenje fitnessa i prehrane. Polazeći od definiranih ciljeva, rad je sustavno pokrio sve faze projekta, od teorijske analize i dizajna arhitekture do tehničke implementacije i kritičke evaluacije.

Kroz rad je uspješno demonstrirana primjena suvremenih tehnologija i metodologija. Razvijena je potpuno funkcionalna web aplikacija korištenjem radnog okvira Flask, koja korisnicima omogućuje sveobuhvatno praćenje ključnih aspekata zdravog načina života. Implementiran je sustav za preporuke koji, iako temeljen na pravilima, pruža personalizirane planove vježbanja i prehrane. Posebna vrijednost aplikacije leži u inovativnoj integraciji velikog jezičnog modela Zephyr-7B-beta kao konverzacijskog AI asistenta, što predstavlja korak naprijed u odnosu na standardna rješenja u ovoj domeni. Svi postavljeni ciljevi rada su time ostvareni: aplikacija je dizajnirana i implementirana, sustav preporuka i AI asistent su integrirani, a cjelokupni proces je detaljno dokumentiran i analiziran.

Kritičkim osvrtom na implementirano rješenje identificirane su njegove snage, poput modularne arhitekture i holističkog pristupa, ali i ključne slabosti, prvenstveno statična priroda sustava preporuka i ograničena kontekstualna svijest AI asistenta. Na temelju tih spoznaja, u radu su izneseni detaljni i utemeljeni prijedlozi za budući rad. Središnji prijedlog je tranzicija sustava preporuka na paradigmu učenja s potkrepljenjem (RL). Definiranjem Markovljevog procesa odlučivanja, gdje su stanja, akcije i nagrade prilagođene domeni fitnessa, otvara se put ka stvaranju dinamičnog i adaptivnog sustava koji uči iz korisničkih interakcija i kontinuirano optimizira svoje preporuke.

Zaključno, ovaj rad ne predstavlja samo dokumentaciju o izrađenom softverskom proizvodu, već pruža i jasan putokaz za njegov daljnji razvoj. Predložena unapređenja, posebice primjena učenja s potkrepljenjem, imaju potencijal transformirati aplikaciju u istinski inteligentnog partnera za zdravlje, potvrđujući time ogroman potencijal primjene naprednih tehnika umjetne inteligencije u rješavanju stvarnih problema i poboljšanju kvalitete života.

# Literatura

1. Applying Machine Learning. (n.d.). *Reinforcement Learning for Recommender Systems*. Preuzeto s <https://applyingml.com/resources/rl-for-recsys/>
2. Babić, M. (2021). *Aplikacija za praćenje fitness treninga* (Završni rad). Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek. Preuzeto s <https://repozitorij.etfos.hr/islandora/object/etfos:3411>
3. Boost Commerce. (2023). *Rule-Based vs. AI-Powered Product Recommendation: Which One Is for You?*. Preuzeto s <https://blog.boostcommerce.net/posts/rule-based-ai-powered-recommendation>
4. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025>
5. Dataloop AI. (n.d.). *HuggingFaceH4/zephyr-7b-beta Model Card*. Preuzeto s <https://dataloop.ai/library/model/huggingfaceh4_zephyr-7b-beta/>
6. Flask Pallets. (2024). *Larger Applications as Packages*. Preuzeto s <https://flask.palletsprojects.com/en/stable/patterns/packages/>
7. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.
8. GeeksforGeeks. (n.d.). *Rule-based system in AI*. Preuzeto s <https://www.geeksforgeeks.org/artificial-intelligence/rule-based-system-in-ai/>
9. Golja, D. (2024). *Izrada aplikacije za fitness i zdravlje primjenom razvojnog okvira Quasar i Firebase* (Specijalistički diplomski stručni rad). Veleučilište u Rijeci, Rijeka. Preuzeto s <https://urn.nsk.hr/urn:nbn:hr:125:980358>
10. Greengeeks. (2023). *Django vs Flask: Which Python Framework Should You Choose?*. Preuzeto s <https://www.greengeeks.com/blog/django-vs-flask-python-framework/>
11. Negnevitsky, M. (2011). *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley.
12. Nejati, H. (2023). *What is the Data Mapper Pattern?*. Medium. Preuzeto s <https://hosseinnejati.medium.com/what-is-the-data-mapper-pattern-219ce6484e28>
13. Nucamp. (2024). *Securing Your Flask Web Application: Best Practices and Tools*. Preuzeto s <https://www.nucamp.co/blog/coding-bootcamp-back-end-with-python-and-sql-securing-your-flask-web-application>
14. NumberFour Analytics. (2024). *A Deep Dive into Rule-Based Expert Systems*. Preuzeto s <https://www.numberanalytics.com/blog/rule-based-expert-systems-deep-dive>
15. OWASP. (2021). *OWASP Top 10:2021*. Preuzeto s(<https://owasp.org/Top10/>)
16. Pallets Projects. (n.d.). *Security Considerations for Flask*. Preuzeto s <https://flask.palletsprojects.com/en/stable/web-security/>
17. PingCap. (2024). *10 Essential Tips for Mastering SQLAlchemy in Python*. Preuzeto s <https://www.pingcap.com/article/10-essential-tips-mastering-sqlalchemy-python/>
18. PromptLayer. (n.d.). *Zephyr-7B-Beta Model Card*. Preuzeto s <https://www.promptlayer.com/models/zephyr-7b-beta>
19. Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. arXiv preprint arXiv:2305.18290.
20. Ražov, R. (2024). *IZRADA ANDROID APLIKACIJE ZA PRAĆENJE FITNES NAPRETKA* (Diplomski rad). Sveučilište u Splitu, Fakultet elektrotehnike, strojarstva i brodogradnje, Split. Preuzeto s <https://urn.nsk.hr/urn:nbn:hr:179:049721>
21. Relevance AI. (2024). *How to Implement Hugging Face Zephyr 7B Model for Your Projects*. Preuzeto s <https://relevanceai.com/llm-models/implement-hugging-face-zephyr-7b-model-for-your-projects>
22. Rochester Institute of Technology. (n.d.). *Rule-Based Expert Systems*. Preuzeto s <https://www.cs.rit.edu/~lr/courses/ai/lectures/topic%205.pdf>
23. Rostami, A., Jain, R., & Rahmani, A. M. (2024). *Food Recommendation as Language Processing (F-RLP): A Personalized and Contextual Paradigm*. arXiv preprint arXiv:2402.07477.
24. Schafer, C. (2018). *Python Flask Tutorial: Full-Featured Web App Part 1 - Getting Started* [Video]. YouTube. Preuzeto s(https://www.youtube.com/watch?v=MwZwr5Tvyxo)
25. Schneider, S. (2023). *Understanding Attention and Transformers*. Medium. Preuzeto s <https://medium.com/@stefanbschneider/understanding-attention-and-transformers-d84b016cd352>
26. Shaped.ai. (2024). *Deep Reinforcement Learning for Recommender Systems: A Survey*. Preuzeto s <https://www.shaped.ai/blog/deep-reinforcement-learning-for-recommender-systems--a-survey>
27. Shapovalova, N. N., Rybalchenko, O. H., & Striuk, A. M. (2024). A Survey on Reinforcement Learning for Recommender Systems. *CEUR Workshop Proceedings, 3917*, 428-442.
28. Simplilearn. (2023). *Flask vs Django: Which Is the Best Python Framework in 2024?*. Preuzeto s <https://www.simplilearn.com/flask-vs-django-article>
29. Soshace. (2023). *Optimizing Database Interactions in Python: SQLAlchemy Best Practices*. Preuzeto s <https://soshace.com/optimizing-database-interactions-in-python-sqlalchemy-best-practices/>
30. Superannotate. (2024). *Direct Preference Optimization (DPO) Explained*. Preuzeto s <https://www.superannotate.com/blog/direct-preference-optimization-dpo>
31. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.
32. Sveučilište u Zagrebu, Filozofski fakultet, Odsjek za informacijske i komunikacijske znanosti. (n.d.). *Upute i pravila za prijavu, pisanje i obranu rada - v8*.
33. Traversy, B. (2017). *Python Flask From Scratch - [Part 1] - Getting Started* [Video]. YouTube. Preuzeto s(<https://www.youtube.com/watch?v=zRwy8gtgJ1A>)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N.,... & Polosukhin, I. (2017). *Attention is all you need*. Advances in neural information processing systems, 30.
35. Wang, Y., Wen, Y., Wu, X., Wang, L., & Cai, H. (2025). Reinforcement Learning-Based Food Recommendation System for Dietary Optimization and Health Management. *Journal of Artificial Intelligence and Information*.
36. Zaki, M., Abdelsamea, M. M., & Gaber, M. M. (2024). A Collaborative Filtering-Based Reinforcement Learning Approach for Personalized Meal Recommendation Systems. *Applied Sciences, 14*(3), 1222.

**Integracija Neuronskih Mreža u Personalizirano Savjetovanje o Prehrani: Razvoj i Primjena Web Aplikacije**

# Sažetak

U svijetu gdje je prehrana postala centralni dio osobnog zdravlja i dobrobiti, potreba za personaliziranim prehrambenim savjetima postaje sve očitija. Ovaj rad bavi se razvojem inovativne web aplikacije namijenjene personaliziranom savjetovanju o prehrani, koristeći napredne tehnološke pristupe, uključujući neuronske mreže i tehnologije obrade prirodnog jezika (NLP). Cilj aplikacije je pružiti korisnicima personalizirane prehrambene planove i emocionalnu podršku, nadmašujući ograničenja tradicionalnih aplikacija za praćenje prehrane i vježbanja. Kroz integraciju genetskih podataka, analize mikrobioma i metaboličkih informacija, aplikacija cilja na identifikaciju specifičnih nutritivnih potreba pojedinca, omogućavajući izradu prehrambenih preporuka koje odgovaraju biološkoj jedinstvenosti korisnika. Dodatno, primjenom NLP-a i analize sentimenta u analizi emocionalnog stanja korisnika, aplikacija nudi strategije za upravljanje emocionalnim jedenjem i potiče zdrave prehrambene izbore usklađene s emocionalnim potrebama korisnika. Aplikacija također uključuje društvene aspekte, poput grupa podrške, foruma i mogućnosti dijeljenja postignuća, promičući društvenu interakciju i zajedničku podršku među korisnicima. Elementi igre i nagradni bodovi dodatno motiviraju korisnike na aktivno korištenje aplikacije. Ovaj rad detaljno razmatra proces razvoja aplikacije, od konceptualizacije do implementacije, s fokusom na korištene tehnologije, metodologije, te evaluaciju korisničkog iskustva i učinkovitosti aplikacije u ostvarivanju personaliziranih prehrambenih ciljeva. Predstavljajući sinergiju između tehnologije i nutricionizma, aplikacija nudi korisnicima jedinstven alat za ostvarivanje prehrambenih i zdravstvenih ciljeva, prilagođen njihovim individualnim potrebama i emocionalnom stanju.

**Integration of Neural Networks into Personalized Nutrition Counseling: Development and Application of a Web Application**

# Summary

In a world where nutrition has become a central part of personal health and well-being, the need for personalized dietary advice is increasingly evident. This paper focuses on the development of an innovative web application designed for personalized nutrition counseling, utilizing advanced technological approaches, including neural networks and natural language processing (NLP) technologies. The goal of the application is to provide users with personalized meal plans and emotional support, surpassing the limitations of traditional diet and fitness tracking applications.

Through the integration of genetic data, microbiome analysis, and metabolic information, the application aims to identify an individual’s specific nutritional needs, enabling the creation of dietary recommendations tailored to each user’s biological uniqueness. Additionally, by applying NLP and sentiment analysis to assess the user’s emotional state, the application offers strategies to manage emotional eating and encourages healthy food choices aligned with the user’s emotional needs.

The application also includes social aspects, such as support groups, forums, and the ability to share achievements, promoting social interaction and mutual support among users. Gamification elements and reward points further motivate users to actively engage with the application.

This paper thoroughly examines the process of developing the application, from conceptualization to implementation, with a focus on the technologies used, the methodologies employed, and the evaluation of user experience and the application’s effectiveness in achieving personalized dietary goals. By presenting a synergy between technology and nutrition science, the application offers users a unique tool for achieving their dietary and health objectives, tailored to their individual needs and emotional well-being.

1. Schafer, C. (2018). Python Flask Tutorial: Full-Featured Web App Part 1 - Getting Started [Video]. YouTube. https://www.youtube.com/watch?v=MwZwr5Tvyxo [↑](#footnote-ref-1)
2. Simplilearn. (2023). Flask vs Django: Which Is the Best Python Framework in 2024?. Preuzeto s https://www.simplilearn.com/flask-vs-django-article [↑](#footnote-ref-2)
3. Greengeeks. (2023). Django vs Flask: Which Python Framework Should You Choose?. Preuzeto s https://www.greengeeks.com/blog/django-vs-flask-python-framework/ [↑](#footnote-ref-3)
4. Traversy, B. (2017). Python Flask From Scratch - [Part 1] - Getting Started [Video]. YouTube. https://www.youtube.com/watch?v=zRwy8gtgJ1A [↑](#footnote-ref-4)
5. Pallets Projects. (n.d.). Security Considerations for Flask. Preuzeto s https://flask.palletsprojects.com/en/stable/web-security/ [↑](#footnote-ref-5)
6. Flask Pallets. (2024). Larger Applications as Packages. Preuzeto s https://flask.palletsprojects.com/en/stable/patterns/packages/ [↑](#footnote-ref-6)
7. Soshace. (2023). Optimizing Database Interactions in Python: SQLAlchemy Best Practices. Preuzeto s https://soshace.com/optimizing-database-interactions-in-python-sqlalchemy-best-practices/ [↑](#footnote-ref-7)
8. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional. [↑](#footnote-ref-8)
9. Nejati, H. (2023). What is the Data Mapper Pattern?. Medium. Preuzeto s https://hosseinnejati.medium.com/what-is-the-data-mapper-pattern-219ce6484e28 [↑](#footnote-ref-9)
10. Corgea. (2025). Flask Security Best Practices 2025. Preuzeto s https://corgea.com/blog/flask-security-best-practices-2025 [↑](#footnote-ref-10)
11. PingCap. (2024). 10 Essential Tips for Mastering SQLAlchemy in Python. Preuzeto s https://www.pingcap.com/article/10-essential-tips-mastering-sqlalchemy-python/ [↑](#footnote-ref-11)
12. NumberFour Analytics. (2024). A Deep Dive into Rule-Based Expert Systems. Preuzeto s https://www.numberanalytics.com/blog/rule-based-expert-systems-deep-dive [↑](#footnote-ref-12)
13. Negnevitsky, M. (2011). Artificial Intelligence: A Guide to Intelligent Systems. Addison-Wesley. [↑](#footnote-ref-13)
14. GeeksforGeeks. (n.d.). Rule-based system in AI. Preuzeto s https://www.geeksforgeeks.org/artificial-intelligence/rule-based-system-in-ai/ [↑](#footnote-ref-14)
15. Rochester Institute of Technology. (n.d.). Rule-Based Expert Systems. Preuzeto s https://www.cs.rit.edu/~lr/courses/ai/lectures/topic%205.pdf [↑](#footnote-ref-15)
16. Rochester Institute of Technology. (n.d.). *Rule-Based Expert Systems*. Preuzeto s <https://www.cs.rit.edu/~lr/courses/ai/lectures/topic%205.pdf> [↑](#footnote-ref-16)
17. Boost Commerce. (2023). Rule-Based vs. AI-Powered Product Recommendation: Which One Is for You?. Preuzeto s https://blog.boostcommerce.net/posts/rule-based-ai-powered-recommendation [↑](#footnote-ref-17)
18. GeeksforGeeks. (n.d.). Rule-based system in AI. Preuzeto s https://www.geeksforgeeks.org/artificial-intelligence/rule-based-system-in-ai/ [↑](#footnote-ref-18)
19. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press. [↑](#footnote-ref-19)
20. Shapovalova, N. N., Rybalchenko, O. H., & Striuk, A. M. (2024). A Survey on Reinforcement Learning for Recommender Systems. CEUR Workshop Proceedings, 3917, 428-442. [↑](#footnote-ref-20)
21. Applying Machine Learning. (n.d.). Reinforcement Learning for Recommender Systems. Preuzeto s https://applyingml.com/resources/rl-for-recsys/ [↑](#footnote-ref-21)
22. Shaped.ai. (2024). Deep Reinforcement Learning for Recommender Systems: A Survey. Preuzeto s https://www.shaped.ai/blog/deep-reinforcement-learning-for-recommender-systems--a-survey [↑](#footnote-ref-22)
23. Wang, Y., Wen, Y., Wu, X., Wang, L., & Cai, H. (2025). Reinforcement Learning-Based Food Recommendation System for Dietary Optimization and Health Management. Journal of Artificial Intelligence and Information. [↑](#footnote-ref-23)
24. Zaki, M., Abdelsamea, M. M., & Gaber, M. M. (2024). A Collaborative Filtering-Based Reinforcement Learning Approach for Personalized Meal Recommendation Systems. Applied Sciences, 14(3), 1222. [↑](#footnote-ref-24)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N.,... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30. [↑](#footnote-ref-25)
26. Schneider, S. (2023). Understanding Attention and Transformers. Medium. Preuzeto s https://medium.com/@stefanbschneider/understanding-attention-and-transformers-d84b016cd352 [↑](#footnote-ref-26)
27. Dataloop AI. (n.d.). HuggingFaceH4/zephyr-7b-beta Model Card. Preuzeto s https://dataloop.ai/library/model/huggingfaceh4\_zephyr-7b-beta/ [↑](#footnote-ref-27)
28. Relevance AI. (2024). How to Implement Hugging Face Zephyr 7B Model for Your Projects. Preuzeto s https://relevanceai.com/llm-models/implement-hugging-face-zephyr-7b-model-for-your-projects [↑](#footnote-ref-28)
29. Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv preprint arXiv:2305.18290 [↑](#footnote-ref-29)
30. Superannotate. (2024). Direct Preference Optimization (DPO) Explained. Preuzeto s https://www.superannotate.com/blog/direct-preference-optimization-dpo [↑](#footnote-ref-30)
31. Dataloop AI. (n.d.). HuggingFaceH4/zephyr-7b-beta Model Card. Preuzeto s https://dataloop.ai/library/model/huggingfaceh4\_zephyr-7b-beta/ [↑](#footnote-ref-31)
32. Dataloop AI. (n.d.). HuggingFaceH4/zephyr-7b-beta Model Card. Preuzeto s https://dataloop.ai/library/model/huggingfaceh4\_zephyr-7b-beta/ [↑](#footnote-ref-32)
33. Ražov, R. (2024). *IZRADA ANDROID APLIKACIJE ZA PRAĆENJE FITNES NAPRETKA* (Diplomski rad). Sveučilište u Splitu, Fakultet elektrotehnike, strojarstva i brodogradnje, Split. Preuzeto s <https://urn.nsk.hr/urn:nbn:hr:179:049721> [↑](#footnote-ref-33)
34. Babić, M. (2021). *Aplikacija za praćenje fitness treninga* (Završni rad). Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek. Preuzeto s <https://repozitorij.etfos.hr/islandora/object/etfos:3411> [↑](#footnote-ref-34)
35. Golja, D. (2024). *Izrada aplikacije za fitness i zdravlje primjenom razvojnog okvira Quasar i Firebase* (Specijalistički diplomski stručni rad). Veleučilište u Rijeci, Rijeka. Preuzeto s <https://urn.nsk.hr/urn:nbn:hr:125:980358> [↑](#footnote-ref-35)
36. Flask Pallets. (2024). Larger Applications as Packages. Preuzeto s https://flask.palletsprojects.com/en/stable/patterns/packages/ [↑](#footnote-ref-36)
37. U.S. Department of Agriculture, Agricultural Research Service. (2019). *FoodData Central*. <https://fdc.nal.usda.gov> [↑](#footnote-ref-37)
38. Bansal, N. (2022). *Gym Exercise Dataset*. Kaggle. <https://www.kaggle.com/datasets/niharika41298/gym-exercise-data> [↑](#footnote-ref-38)
39. Mooney, P. T. (2020). *RecipeNLG (cooking recipes dataset)*. Kaggle. <https://www.kaggle.com/datasets/paultimothymooney/recipenlg> [↑](#footnote-ref-39)
40. Withana, N. (2023). *Emotions dataset for NLP*. Kaggle. <https://www.kaggle.com/datasets/nelgiriyewithana/emotions> [↑](#footnote-ref-40)
41. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025> [↑](#footnote-ref-41)
42. OWASP. (2021). *OWASP Top 10:2021*. Preuzeto s <https://owasp.org/Top10/> [↑](#footnote-ref-42)
43. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025> [↑](#footnote-ref-43)
44. Nucamp. (2024). *Securing Your Flask Web Application: Best Practices and Tools*. Preuzeto s <https://www.nucamp.co/blog/coding-bootcamp-back-end-with-python-and-sql-securing-your-flask-web-application> [↑](#footnote-ref-44)
45. Pallets Projects. (n.d.). *Security Considerations for Flask*. Preuzeto s <https://flask.palletsprojects.com/en/stable/web-security/> [↑](#footnote-ref-45)
46. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025> [↑](#footnote-ref-46)
47. Soshace. (2023). *Optimizing Database Interactions in Python: SQLAlchemy Best Practices*. Preuzeto s <https://soshace.com/optimizing-database-interactions-in-python-sqlalchemy-best-practices/> [↑](#footnote-ref-47)
48. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025> [↑](#footnote-ref-48)
49. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025> [↑](#footnote-ref-49)
50. Corgea. (2025). *Flask Security Best Practices 2025*. Preuzeto s <https://corgea.com/blog/flask-security-best-practices-2025> [↑](#footnote-ref-50)
51. Pallets Projects. (n.d.). *Security Considerations for Flask*. Preuzeto s <https://flask.palletsprojects.com/en/stable/web-security/> [↑](#footnote-ref-51)
52. Applying Machine Learning. (n.d.). *Reinforcement Learning for Recommender Systems*. Preuzeto s <https://applyingml.com/resources/rl-for-recsys/> [↑](#footnote-ref-52)
53. Shaped.ai. (2024). *Deep Reinforcement Learning for Recommender Systems: A Survey*. Preuzeto s <https://www.shaped.ai/blog/deep-reinforcement-learning-for-recommender-systems--a-survey> [↑](#footnote-ref-53)
54. Zaki, M., Abdelsamea, M. M., & Gaber, M. M. (2024). A Collaborative Filtering-Based Reinforcement Learning Approach for Personalized Meal Recommendation Systems. *Applied Sciences*, 14(3), 1222. [↑](#footnote-ref-54)
55. Rostami, A., Jain, R., & Rahmani, A. M. (2024). *Food Recommendation as Language Processing (F-RLP): A Personalized and Contextual Paradigm*. arXiv preprint arXiv:2402.07477. [↑](#footnote-ref-55)
56. Zaki, M., Abdelsamea, M. M., & Gaber, M. M. (2024). A Collaborative Filtering-Based Reinforcement Learning Approach for Personalized Meal Recommendation Systems. *Applied Sciences*, 14(3), 1222. [↑](#footnote-ref-56)