Razvoj i testiranje web aplikacije sa sustavom stvaranja preporuka za potporu u dijagnosticiranju, liječenju i prehrani kod autoimunih bolesti

Mandić, Mihaela

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:396724

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij Računarstva

RAZVOJ I TESTIRANJE WEB APLIKACIJE SA SUSTAVOM STVARANJA PREPORUKA ZA POTPORU U DIJAGNOSTICIRANJU, LIJEČENJU I PREHRANI KOD AUTOIMUNIH BOLESTI

Diplomski rad

Mihaela Mandić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 03.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Mihaela Mandić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1225R, 07.10.2021.
OIB studenta:	89472670897
Mentor:	prof. dr. sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	prof. dr. sc. Goran Martinović
Član Povjerenstva 2:	doc. dr. sc. Tomislav Galba
Naslov diplomskog rada:	Razvoj i testiranje web aplikacije sa sustavom stvaranja preporuka za potporu u dijagnosticiranju, liječenju i prehrani kod autoimunih bolesti
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu diplomskog rada potrebno je opisati probleme i izazove pri dijagnosticiranju, liječenju i prehrani kod autoimunih bolesti, te analizirati postojeća slična rješenja i postupke stvaranja preporuka. Također, treba definirati funkcionalne i nefunkcionalne zahtjeve na web sustav, predložiti model i arhitekturu web sustava (temeljenu na predlošku MVC ili nekom drugom obliku arhitekture) na strani korisnika i na strani poslužitelja, te prikladan višekriterijski postupak stvaranja preporuka i pristup ručnog i automatiziranog testiranja sustava. Web aplikacija treba omogućiti stvaranje profila pacijenta, unos i pohranu simptoma i pokazatelja bolesti, postupak dijagnosticiranja bolesti, preporuke tijeka liječenja i terapije, te prikladnog režima prehrane uvažavajući uspješnost liječenja i nuspojave terapije. Koristeći programski jezik Java, razvojne okvire Spring i Bootstrap, web aplikaciju s bazom podataka treba programski ostvariti, obaviti ručno i automatizirano testiranje, te ispitivanje i analizu programskog rješenja za zastupljenije autoimune bolesti i slučajeve korištenja. Tema rezervirana za studenta: Mihaela Mandić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	03.09.2023.
Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.09.2023.

Ime i prezime studenta:	Mihaela Mandić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1225R, 07.10.2021.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj i testiranje web aplikacije sa sustavom stvaranja preporuka za potporu u dijagnosticiranju, liječenju i prehrani kod autoimunih bolesti**

izrađen pod vodstvom mentora prof. dr. sc. Goran Martinović

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Autoimune bolesti predstavljaju složene medicinske poremećaje u kojima vlastiti imunološki sustav napada tjelesne stanice, tkiva i organe umjesto da ih štiti. Takav neispravan odgovor imunološkog sustava može dovesti do različitih stanja kao što su reumatoidni artritis, sistemski lupus eritematosus, Hashimotov tireoiditis i mnogi drugi. Utvrđivanje dijagnoze autoimunih bolesti zahtijeva integraciju kliničkih znakova, laboratorijskih testova i medicinskih snimaka kako bi se prepoznali specifični pokazatelji i simptomi bolesti. Liječenje autoimunih bolesti često uključuje kombiniranje farmakoterapije, upravljanja simptomima i reguliranja imunološkog odgovora. Imunosupresivni lijekovi koriste se kako bi se suzbio pretjerani imunološki odgovor i smanjila upala. Također se primjenjuje biološka terapija koja cilja specifične imunološke mehanizme. Upravljanje prehranom i životnim stilom često su ključni čimbenici za kontrolu simptoma autoimunih bolesti. Rano prepoznavanje, pristupi liječenju prilagođeni svakom pojedincu te suradnja između pacijenta i medicinskog tima igraju ključnu ulogu u uspješnom upravljanju ovim stanjima.

Osnovni zadatak ovoga rada jest razviti web aplikaciju koja će unaprijediti preciznost dijagnostike autoimunih bolesti te pružiti prilagođene preporuke za terapiju i prehrambene navike. Temelj ove aplikacije leži u upotrebi sustava za podršku donošenju odluka, koristeći relevantne informacije iz područja medicinske dijagnostike i nutricionizma kako bi stvorila personalizirane smjernice za svakog korisnika. Cilj ove aplikacije je olakšati proces dijagnosticiranja, terapije i pravilne prehrane kod osoba koje se suočavaju s autoimunim oboljenjima.

U drugom poglavlju su obuhvaćene autoimune bolesti o kojima će ovaj rad pružiti opširniji pregled, a istovremeno će se istaknuti neki od izazova povezanih s dijagnozom bolesti i pravilnim propisivanjem terapije i prehrambenih smjernica. Treće poglavlje posvećeno je detaljnom prikazu strukture i modela razvijenog web sustava. U četvrtom poglavlju prikazano je programsko rješenje, uključujući opis implementiranih funkcionalnosti te alata i programskih jezika koji su korišteni. Peto poglavlje opisuje procese testiranja ostvarenoga web rješenja, obuhvaćajući ručno i automatizirano testiranje. Šesto poglavlje daje uvid u načine korištenja aplikacije sa stajališta liječnika i administratora, te njeno ispitivanje i analizu.

2. SUSTAVI ZA POTPORU DIJAGNOSTICIRANJU BOLESTI, ODABIR LIJEKOVA I PREHRANE NA TEMELJU PREPORUKA I PREGLED STANJA U PODRUČJU

Kako bi se razvio sustav za potporu u dijagnosticiranju autoimunih bolesti i omogućila preporuka lijeka i prehrane, najprije se treba razumjeti što su to autoimune bolesti, koje su najpoznatije autoimune bolesti i koji simptomi upućuju na njih kako bi se mogle dijagnosticirati. Osim toga, potrebno je znati što je to lijek te kako se odabire prikladan lijek za određenu bolest uzimajući u obzir određene atribute pacijenta. Nadalje, specifična prehrana može pripomoći kod liječenja autoimunih bolesti pa se, stoga, uzimajući u obzir pacijenta i dijagnosticiranu bolest, treba odabrati prikladna prehrana. U nastavku su prikazani primjeri sustava za potporu u dijagnozi bolesti te primjeri sustava za preporuku lijeka i prehrane. Također, u nastavku će biti rečeno nešto više o problematici dijagnoze bolesti i preporuke lijeka i prehrane te o važnosti testiranja sustava koji omogućuje navedene mogućnosti.

2.1. Izazovi dijagnoze autoimunih bolesti i pregled najpoznatijih autoimunih bolesti

U svrhu boljeg razumijevanja što su to autoimune bolesti i kako su nastale, najprije se treba reći nešto više o imunološkom sustavu. Prema [1], imunološki sustav je obrambeni mehanizam u tijelu koji brani sva dobra tkiva, stanice i organe u tijelu od svih patogenih organizama, bakterija, virusa te transformiranih stanica ili tkiva, odnosno tumora. Problemi u radu imunološkog sustava mogu dovesti do toga da se organizam ne brani od patogenih organizama, loših stanica ili tkiva što dovodi do toga da domaćin brzo podlegne invaziji bolesti bez ikakvog otpora, te, s druge strane, imunološki sustav može neispravno identificirati dobar dio organizma kao neispravan, odnosno kao tumor ili patogeni organizam te na taj način napadati i boriti se protiv ispravnog organizma što dovodi do autoimunih bolesti.

Kao što je navedeno u [2], autoimune bolesti obuhvaćaju više od 80 različitih bolesti koje se temelje na istom mehanizmu nastanka, odnosno imunološkom napadu organizma na vlastite organe, tkiva i stanice. One mogu zahvatiti različite dijelove tijela, što rezultira raznolikim kliničkim simptomima. Iako su većina ovih bolesti rijetke pojedinačno, kada se gledaju zajedno, predstavljaju jednu od najčešćih oboljenja u industrijaliziranim društvima. One često dovode do onesposobljenosti, gubitka funkcionalnosti organa i smanjenja produktivnosti te zahtijevaju intenzivnu skrb.

Prema [3, 4], ove bolesti sve više postaju prijetnja za javno zdravlje diljem svijeta te je poznato da genetika i spol utječu na mogućnost razvoja i pojave autoimune bolesti, no, također novija istraživanja navode da je incidencija autoimunih bolesti rapidno porasla u

(post)industrijaliziranim i bogatim društvima, što ukazuje na to da promjene u ekologiji i načinu života također potiču razvoj autoimunih bolesti. Najčešći čimbenici koji spadaju u ekologiju i način života su prehrana, mikrobiom, infekcije, lijekovi, tjelesna aktivnost, stres, spavanje, rad u smjenama, izloženost onečišćenju te pušenje. Nadalje, epidemiološka istraživanja ukazuju da 80% pacijenata s autoimunim bolestima su žene, autoimune bolesti češće se pojavljuju kod žena te incidencija nekih autoimunih bolesti brže raste kod žena nego kod muškaraca. Najčešće su pogođene žene u dobi između puberteta i menopauze, što upućuje da spolni hormoni imaju utjecaj na razvoj autoimunih bolesti.

Jedne od poznatijih autoimunih bolesti, za čiju će dijagnozu ovaj sustav biti potpora, su: reumatoidni artritis, sistemski lupus eritematozus, psorijaza, Sjögrenov sindrom, multipla skleroza, Hashimotov tireoiditis, tip 1 dijabetes, celijakija, Crohnova bolest te Gravesova bolest. Prema [5], reumatoidni artritis je dugotrajna, opća upalna bolest koja zahvaća vezivno tkivo, od kojih prvenstveno zglobove, no, također, utječe i na druge dijelove tijela, a uzrok nastanka nije poznat. Pogađa oko 1% svjetske populacije te se obično manifestira između 35. i 50. godine života kod 80% pacijenata. Sklonost obolijevanju češće je kod žena nego kod muškaraca gdje je omjer 3:1. Tijek bolesti je raznolik, varirajući od blage i kratkotrajne bolesti koja pogađa samo nekoliko zglobova s minimalnim oštećenjima, do progresivnog poliartritisa koji uzrokuje značajna funkcionalna oštećenja i deformitete. Kao što je navedeno u [6], sistemski lupus eritematozus je autoimuna bolest koja može zahvatiti mnoge i različite organe i tkiva, poput mišića, zglobova, mozga, perifernog živčanog sustava, pluća, srce, bubrega, kože te krvnih komponenata. Uzrok ove bolesti nije poznat, a najčešće se pojavljuje kod žena te najčešće zahvaća dob od 15. do 45. godine života. Vrlo je teško dijagnosticirati ovu bolest jer simptomi vrlo brzo dolaze i odlaze te nalikuju simptomima drugih bolesti. Osim toga, ne postoji niti jedan laboratorijski test koji može sa sigurnošću potvrditi da se radi o ovoj bolesti. Prema [7], psorijaza je kronična upalna bolest kože koja je karakterizirana ubrzanom promjenom kože s prisutnošću crvenila i ljuštenja. Prisutna je u otprilike 2% opće populacije te se najčešće javlja između 20. i 30. godine te između 50. i 60. godine života gdje podjednako pogađa muškarce i žene. Psorijaza može nastati zbog genetskih i okolišnih čimbenika, no to ne mora uvijek biti slučaj. Primjerice, ukoliko jedan roditelj ima dijagnosticiranu psorijazu, rizik da će i dijete imati tu bolest je otprilike 14%, no ukoliko oba roditelja imaju navedenu bolest, tada postotak raste i do 41%. Nadalje, okolišni čimbenici poput fizičke traume, infekcije, niske razine kalcija u krvi, stresa te primjena određenih lijekova koji sadržavaju litij, beta-blokatore, antimalarike, interferon može utjecati na pojavu psorijaze. Osim navedenih, brza promjena tjelesne težine, konzumacija alkohola te pušenje mogu biti povezani s pojavom psorijaze, no

to nije u potpunosti dokazano. Po uzoru na [8], Sjögrenov sindrom je kronična autoimuna bolest koju karakterizira upala žlijezda s vanjskim lučenjem, poput upala slinovnice i suzne žlijezde što dovodi do najčešćeg simptoma u bolesnika, a to je suhoća usta i očiju. Također, ova bolest može zahvatiti i kožu, dišne puteve, urogenitalni trakt te može utjecati i na druge dijelove tijela poput zglobova. U 95% slučajeva se manifestira kod ženskih bolesnika te uglavnom pogađa populaciju od 40-te do 60-te godine života. Osim da je uzrokovana oštećenjem rada imunološkog sustava, ne zna se sa sigurnošću što uzrokuje ovu bolest. Smatra se da određeni virusi, poput Epstein-Barrovog virusa ili humanog T-limfotropnog virusa, mogu utjecati na pojavu ove bolesti te, također, kombinacija okolišnih i životnih čimbenika utječe na njenu pojavu. Prema [9], multipla skleroza je bolest kod koje su živčana vlakna u mozgu, živcima za vid i leđnoj moždini oštećena ili uništena. Ova bolest može uzrokovati probleme s koordinacijom ili hodanjem, utrnulost i trnce u udovima, slabost mišića, probleme s vidom, govorom te gutanjem. Češće se manifestira kod žena nego kod muškaraca te se najčešće pojavljuje u razdoblju od 23-te do 40-te godine života, no može se pojaviti bilo kada. Također, može se pojaviti zbog genetskih predispozicija te okolišnih čimbenika gdje se smatra da smanjena izloženost sunčevoj svjetlosti (unos vitamina D) te pušenje utječe na češću pojavu ove bolesti. Osim navedenog, unos određene prehrane, poput konzumacije ribe utječe na smanjenje pojave ove bolesti. Kao što je objašnjeno u [10], Hashimotov tireoiditis je kronična upala štitnjače gdje imunološki sustav napada štitnjaču i usporava njen rad. Ova bolest češće pogađa žene nego muškarce i obično se javlja između 30. i 50. godine života. Uzrok ove bolesti je najčešće genetika, no, također, ovu bolest mogu uzrokovati stres, određeni virusi i bakterije. Često se pojavljuje u kombinaciji s drugim autoimunim bolestima, poput Gravesove bolesti, Sjögrenovog sindroma, sistemskog lupusa eritematosusa te reumatoidnog artritisa. Neki od simptoma ove bolesti su debljanje, umor, osjetljivost na hladnoću, suha koža i kosa, povećana štitnjača te depresija. Prema [11], dijabetes tip 1 je poremećaj koji proizlazi iz kroničnog autoimunog uništavanja beta stanica gušterače koje proizvode inzulin. Najzastupljenija je među djecom i adolescentima do 20. godine života te se prenosi genetski, putem gena za antigene ljudskih leukocita, i negenetski, putem okolišnih čimbenika kao što su prehrana, lijekovi te toksini. Podjednako pogađa žene i muškarce, a simptomi ove bolesti mogu biti umor, često mokrenje, povećana žeđ i glad, neobjašnjiv gubitak tjelesne težine, zamagljen vid, sporo zarastanje rana ili česte infekcije. Celijakija je, kao što je objašnjeno u [12], kronični autoimuni poremećaj koji je posljedica poremećenog imunosnog odgovora sluznice crijeva pri susretu s glutenom. Pogađa ljude svih dobnih skupina, ali se često prvi put manifestira kod djece u dobi od 1. do 7. godine. Simptomi variraju ovisno o dobi: kod mlađe djece mogući su kronični

proljev, zaostajanje u rastu, povraćanje i bol u trbuhu, dok starija djeca mogu imati anemiju, rahitis, probleme s ponašanjem i školovanjem. Bolest može uzrokovati i druge simptome poput zatvora, neuroloških simptoma, a ponekad i epilepsije. Glavni uzrok bolesti je genetska predispozicija i reakcija na gluten, sastojak prisutan u žitaricama. Prema [13], Crohnova bolest je kronična upalna bolest crijeva te utječe na osobe različitih dobnih skupina, ali često se počinje manifestirati u mladosti. Incidencija je viša među bijelom i židovskom populacijom, a što se tiče spola, muškarci i žene su podjednako pogođeni. Simptomi uključuju bol u trbuhu, proljev, umor i smanjen apetit. Uzroci bolesti nisu poznati, no najčešće se povezuju s infekcijama, disfunkcijom imunološkog sustava i prehranom. Po uzoru na [14], Gravesova bolest je autoimuna bolest kod koje zbog prekomjernog izlučivanja hormona štitnjače ili pretjerane aktivnosti štitnjače dolazi do poremećaja u metabolizmu. Najčešće pogađa žene u dobi od 30-te do 50-te godine života i najčešće se manifestira u obliku guše ili izbočenja očiju. Ostali simptomi koji su prisutni kod ove bolesti su gubitak težine, pojačan apetit, nervoza, umor, ubrzan srčani ritam, podrhtavanje dijelova tijela te povećano znojenje. Također, kao i kod većine autoimunih bolesti, nije poznat točan uzrok nastanka ove bolesti, no smatra se da je među najbitnijima genetski čimbenik, a zatim uzrok mogu biti i infekcije, povećani unos joda te određeni tumori, poput tumora štitne žlijezde te hipofize.

Kao što je navedeno u [15], budući da postoji jako puno autoimunih bolesti od kojih je većina jako rijetka ili neuobičajena, liječnicima je vrlo izazovno dati ispravnu dijagnozu za autoimunu bolest s obzirom da ih nisu navikli vidjeti. Osim toga, liječnicima je vrlo izazovno uopće prepoznati da se pacijentov problem radi o autoimunoj bolesti, budući da jako puno simptoma koji upućuju na autoimunu bolest, također, upućuju i na mnoge druge bolesti, poput umora, slabosti, bolova u zglobovima ili jednostavno se pacijent ne osjeća dobro. Prema Američkoj udruzi za autoimune bolesti, u prosijeku pacijenti koji boluju od uobičajenih i ozbiljnih autoimunih bolesti provedu 4 - 6 godina tražeći dijagnozu i u tom razdoblju posjete 4 – 8 liječnika. Nadalje, simptomi koji se pojavljuju mogu biti zbunjujući gdje vrlo brzo dolaze i nestaju te utječu na različite dijelove tijela što, također, liječnicima otežava dijagnosticiranje. Ponekad testiranje može pripomoći u dijagnozi, no samo ukoliko su prisutni klinički znakovi, kao kod primjerice dijabetesa, ali većina autoimunih bolesti nema jednostavan test koji bi pomogao u dijagnozi gdje čak i ako test pokaže negativne markere, svejedno se niti tada ne može isključiti ta bolest. Mnogi pacijenti su mlade žene koje izgledaju zdravo pa liječnici nemaju osjećaj da se radi o autoimunoj bolesti. U konačnici, dijagnoza autoimunih bolesti nerijetko zahtjeva suradnju različitih medicinskih stručnjaka, uključujući reumatologe,

imunologe, dermatologe i drugo specijalizirano osoblje što ponekad može biti izazovno zbog različitih specijalnosti i pristupa.

2.2. Izazovi odabira lijekova i prehrane kod autoimunih bolesti

Budući da nijedna od autoimunih bolesti trenutno nema potpuno izlječenje, a većina njih ima kronični tijek, pacijenti se suočavaju s dugotrajnim ozbiljnim oboljenjem i skupim tretmanima kao što je navedeno u [2]. Prema [16], trenutno liječenje bolesnika s autoimunim bolestima se uglavnom bazira na imunosupresivima, koji sprječavaju imunološki sustav da pogreškom napada zdrave stanice i tkiva, te na određenoj prehrani, koja može poboljšati imunološki sustav ili olakšati simptome bolesti. Imunosupresivi ograničavaju rad imunološkog sustava bolesnika i na taj način pomažu u sprječavanju daljnjeg oštećenja stanica i upale, što dovodi do toga da pacijenti imaju smanjene simptome ili pak mogu dovesti autoimunu bolest u remisiju gdje pacijent nema nikakvih znakova bolesti. Usprkos tome što imunosupresivi mogu pomoći kod liječenja autoimunih bolesti, ti lijekovi su izuzetno opasni i mogu prouzrokovati brojne nuspojave, poput dijabetesa, gubitka ili rasta kose, glavobolje, umora, ranica u usnama, želučanih tegoba, osteoporoze, debljanja, mučnina i mnoge druge. Također, kod uzimanja imunosupresiva pacijenti trebaju biti vrlo oprezni jer preskakanje doza ili ne uzimanje doza prepisanih od strane liječnika može dovesti do iznenadnog pogoršanja stanja pacijenta koji ponekad mogu biti opasni i po život. Osim toga, pacijenti se trebaju dodatno paziti jer imunosupresivi blokiraju rad imunološkog sustava i s time povećavaju rizik od infekcija. Također, većina lijekova je ograničena ili zabranjena za trudnice ili dojilje jer mogu uzrokovati defekt djeteta ili pobačaj. Određeni lijekovi se ne smiju propisivati maloj djeci jer im štete ili nisu dovoljno ispitani na maloj djeci. Nadalje, svaki pacijent je individualan, u određenom je stadiju bolesti, ima drugačiju genetiku, drugačije je dobi te može drugačije reagirati na isti lijek. Dakle, ne postoji pravilo da će isti lijekovi jednako pomoći kod istih autoimunih bolesti. Postoji potreba za prilagodbom doziranja i tretmana kako bi se postigao optimalan terapijski odgovor. Pacijenti s autoimunim bolestima često uzimaju više lijekova istovremeno, što može dovesti do interakcija između lijekova. Ove interakcije mogu smanjiti učinkovitost terapije ili povećati nuspojave. Potrebno je pažljivo razmotriti sve lijekove koje pacijent uzima kako bi se izbjegle potencijalne negativne posljedice. Većina autoimunih bolesti je kronična i zahtijeva dugotrajnu terapiju gdje održavanje pacijentove suradnje i pridržavanja terapije dugoročno može biti izazovno, posebno ako pacijenti doživljavaju poboljšanja ili ako su suočeni s nuspojavama. Autoimune bolesti često prolaze kroz faze aktivnosti i remisije. Lijekovi i doziranje moraju se moći prilagoditi prema trenutnom stanju bolesti, što zahtijeva redovita praćenja pacijenta i

pravovremene prilagodbe terapije. Neki lijekovi za autoimune bolesti mogu biti skupi i teško dostupni. Ovo može postaviti prepreke za pacijente koji ne mogu priuštiti potrebne tretmane, što dodatno komplicira upravljanje bolesti.

Budući da postoji veliki broj različitih autoimunih bolesti, izazovno je prilagoditi prehranu za svaki tip autoimune bolesti. Prema [17], teško je pronaći liječnika ili nutricionista koji poznaje prirodu autoimunih bolesti i da zna da pacijent može imati kombinaciju više autoimunih bolesti. Primjerice, postoji veza između celijakije i drugih autoimunih bolesti, poput dijabetesa tipa 1, Sjögrenovog sindroma, Chronove bolesti, Hashimotovog tireoditisa i Gravesove bolesti gdje bi nutricionisti trebali biti svjesni ovih veza gdje bi pacijente s navedenim autoimunim bolestima trebali testirati na celijakiju ako imaju simptome povezane s osjetljivošću na gluten. Nutricionisti bi trebali usmjeriti svoju specijalizaciju prema autoimunim bolestima kao što je celijakija, budući da su ovi poremećaji izrazito složeni. Ako se nutricionisti ne posvete temeljitoj analizi ovog područja, mogu više naštetiti nego koristiti. Osim navedenog, kod preporuke prehrane bi se trebalo uzeti u obzir i koji lijekovi su preporučeni pacijentu jer određeni lijekovi mogu reagirati s određenim hranjivim tvarima, poput folne kiseline i vitamina B12.

2.3. Stvaranje preporuka za izbor prikladnih lijekova i prehrane

Po uzoru na [18], stvaranje preporuka za odabir prikladnih lijekova i prehrane predstavlja ključan aspekt medicinske skrbi i savjetovanja o zdravlju. Ovaj proces podrazumijeva detaljnu analizu povijesti bolesti pacijenta, dijagnostičkih podataka te osobnih karakteristika kako bi se izradile prilagođene preporuke koje će najbolje odgovarati njihovim specifičnim potrebama. Prvi korak u ovom procesu je prikupljanje relevantnih informacija o pacijentu, uključujući medicinsku povijest, trenutne dijagnoze, primijenjene terapije, alergijske reakcije i druge važne medicinske podatke. Na temelju ovih podataka, vrši se temeljita analiza dijagnostičkih informacija kako bi se bolje razumjelo trenutno zdravstveno stanje pacijenta. Također se uzimaju u obzir osobne karakteristike pacijenta kao što su dob, spol i prehrambene navike kako bi se izradile preporuke prilagođene njihovim potrebama. Posebna pažnja posvećuje se mogućim interakcijama između različitih lijekova, kako bi se izbjegle nepoželjne reakcije ili smanjila učinkovitost terapije. Na temelju sveobuhvatnih informacija, izrađuju se personalizirane preporuke za odabir lijekova i prehrane. Prilikom odabira lijekova, uzimaju se u obzir dijagnoze, simptomi, potencijalne nuspojave i interakcije. Preporuke vezane uz prehranu uzimaju u obzir prehrambene potrebe pacijenta, alergijske reakcije na hranu i postavljene terapijske ciljeve. Ovaj pristup prilagođenih preporuka omogućuje bolje prilagodbe

terapije individualnim potrebama svakog pacijenta, potiče postizanje optimalnih zdravstvenih rezultata i minimizira rizik od neželjenih reakcija ili komplikacija. Sve ovo predstavlja integraciju medicinskog znanja sa specifičnostima svakog pacijenta kako bi se osigurala najkvalitetnija zdravstvena skrb.

2.4. Značaj testiranja pri razvoju web sustava za potporu u dijagnosticiranju, liječenju i prehrani kod autoimunih bolesti

Kreiranje složenog sustava za preporuku odgovarajućih lijekova, prehrane i dijagnosticiranja bolesti temelji se na korisničkim simptomima koji su temeljan pristup kako bi se osigurala visokokvalitetna preporuka. Izazovi koji se pojavljuju u tom procesu uključuju uzimanje u obzir nuspojava, kontraindikacija, alergija, interakcija s drugim lijekovima te mnogi drugi. Ključ uspješnog sustava leži u uspješnom povezivanju preporuke lijekova s dijagnosticiranjem bolesti. Budući da se simptomi često preklapaju, sustav treba biti sposoban precizno razlikovati različite bolesti koje imaju slične simptome i preporučiti odgovarajuće lijekove za svaku pojedinu dijagnozu. Dodatno, uloga prehrane u liječenju autoimunih bolesti naglašava važnost integracije prehrambenih preporuka u sustav za podršku. Iako bi razvoj sustava koji sugerira lijekove samo temeljem simptoma mogao izgledati jednostavan, važno je izbjegavati takav pristup kako ne bi ugrozili sigurnost i učinkovitost liječenja. Sustav mora pažljivo balansirati sve relevantne čimbenike prije donošenja preporuke. Također, treba biti oprezan u pogledu preporuka lijekova koji imaju samo blage simptome koji se podudaraju s pacijentovom anamnezom. Važno je izbjeći situaciju u kojoj bi pacijent bio izložen nepotrebnom riziku ili nuspojavama umjesto da mu se ponudi sigurnije i učinkovitije rješenje.

Prema [22], testiranje takvih sustava postaje ključno kako bi se osigurala njihova pouzdanost i funkcionalnost. Sustavi bi trebali biti testirani s raznolikim pacijentima i scenarijima, uključujući i rijetke rubne slučajeve. To će omogućiti dubinsku analizu kako sustav reagira u različitim situacijama i kako se nosi s različitim potrebama korisnika. Smanjenje mogućnosti pogrešaka važan je cilj, posebno kad se radi o primjeni u medicinskom kontekstu. Testiranje omogućava identifikaciju slabosti, problema u algoritmima ili nepredviđenih reakcija na različite scenarije. Kroz testiranje, sustav se može optimizirati kako bi pružao što bolje i sigurnije preporuke za liječenje, doprinoseći kvalitetnijoj zdravstvenoj skrbi. U zaključku, složenost i važnost sustava za preporuku lijekova i prehrane zahtijeva sveobuhvatno testiranje čime se osigurava da pacijenti dobiju najbolje moguće preporuke uz minimalan rizik od nuspojava ili nepotrebnih tretmana.

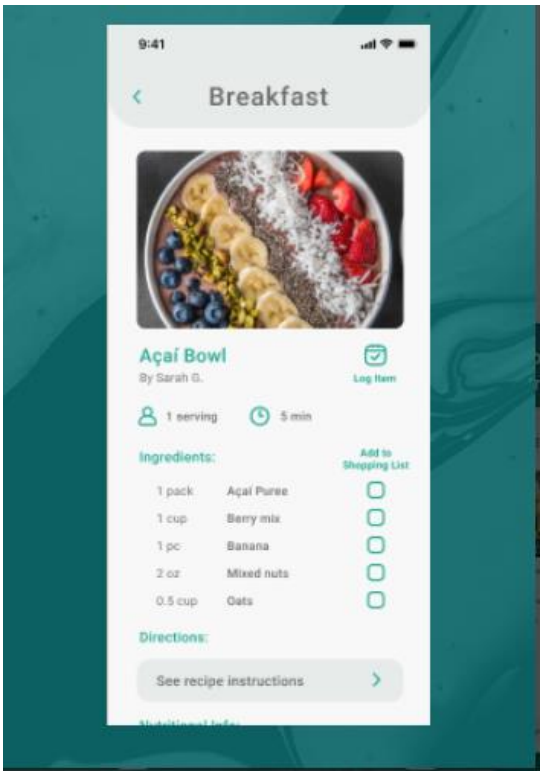
2.5. Postojeća rješenja za dijagnozu bolesti i stvaranje preporuka u liječenju i prehrani

Pregledom postojećih programskih rješenja sličnih ovome web sustavu omogućava se bolje razumijevanje tržišta, korisničke potrebe i konkurencije te izbjegavanje ponavljanja. Ovim pregledom moguće je identificirati praznine ili nedostatke u postojećim proizvodima koje se mogu iskoristiti kako bi se stvorilo bolje programsko rješenje.

2.5.1. Aplikacija Aila Health

Jedno od sličnih rješenja predstavlja aplikacija Aila Health koja ima za zadaću pružiti individualiziranu skrb svakom pojedinačnom pacijentu. Ova aplikacija koristi naprednu analitiku kako bi osnažila zdravstvene radnike i pacijente da donose informirane odluke o zdravlju temeljene na podacima. Kroz ovu aplikaciju je omogućeno svakodnevno praćenje simptoma i vođenje dnevnika u svrhu evidentiranja informacija i fotografija o napretku pacijentovog zdravlja. Spremljene podatke je moguće podijeliti s timom za brigu, čime se poboljšava sveukupna kvaliteta zdravstvene skrbi. Osim toga, putem ove aplikacije postoji mogućnost zakazivanja virtualnih sastanaka s posvećenim integrativnim trenerom za zdravlje kako bi se postavili ciljevi i izradili prilagođeni plan za dobrobit pacijenta. Aila Health aplikacija, također, omogućuje personalizirane prehrambene smjernice i vodiče za obroke koji smanjuju upalu, unapređuju zdravlje crijeva te jačaju imunološki sustav, uzimajući u obzir preferencije i osjetljivosti pacijenta. Slika 2.1 i slika 2.2 prikazuju izgled opisane aplikacije.

Slika 2.1. *Prikaz dijela Aila Health aplikacije [34]*



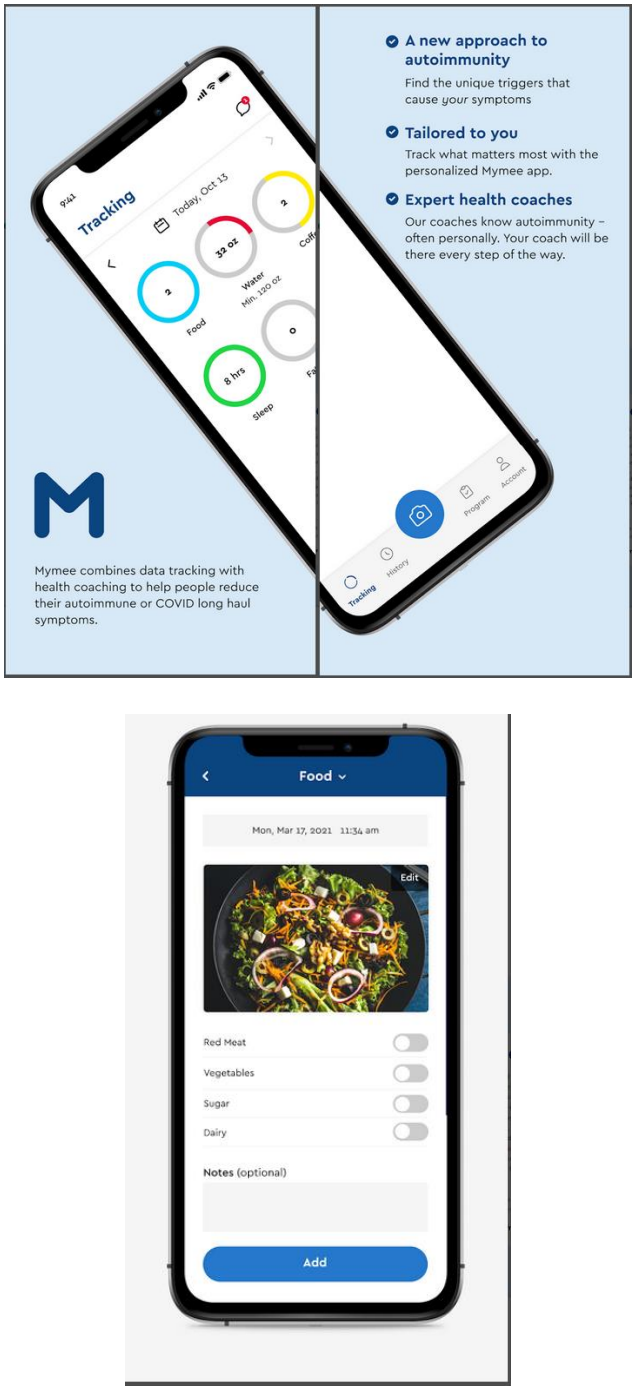
Slika 2.2. *Prikaz izgleda aplikacije Aila Health [34]*

2.5.2. Program Mymee

Program Mymee predstavlja revolucionaran pristup u liječenju autoimunih bolesti, uključujući reumatoidni artritis, lupus, psorijatični artritis, Crohnovu bolest te rijetke ili definirane

poremećaje autoimunosti. Ovaj program suštinski je dizajniran kako bi pružio potpuno personaliziranu skrb za svakog pojedinog pacijenta.

Počevši od uvođenja i procjene ekspozoma, pacijenti stupaju u kontakt s posebno obučenim zdravstvenim trenerom. Putem prikupljenih podataka, kao što su simptomi, prehrambene navike, dodaci prehrani, recepti, farmakogenetika, cijepljenja, infekcije, kirurška povijest te okoliš, identificiraju se prioriteti u zdravstvenim rizicima i simptomima. U situacijama kada je prisutan rizik od lijekova, specijalizirani zdravstveni trener iz Mymee usklađuje konzultaciju s farmaceutom. Tijekom konzultacije detaljno se analiziraju propisani lijekovi, dodaci prehrani i farmakogenetika. Sve preporuke za prilagodbu doziranja ili optimizaciju lijekova podijeljene su s liječnicima i pacijentima te se uključuju u personalizirani plan ispitivanja i njege. Visoko specijalizirani treneri za autoimune bolesti vode tjedna analitička i skrbna savjetovanja. Kroz intuitivno korisničko sučelje personalizirane mobilne aplikacije, pacijenti jednostavno bilježe simptome i potencijalne okidače. Neprestano praćenje ovih okidača te prilagodbe u prehrani i načinu života, temeljeni na čvrstim dokazima, omogućuju precizno utvrđivanje uzroka koji ublažavaju ili pogoršavaju simptome. S vremenom, dnevne aktivnosti se proširuju putem kontinuiranog praćenja dok se ne ostvare postavljeni zdravstveni ciljevi. Na završetku, konačna procjena rizika prepoznaje potrebe za daljnjom njegom i podrškom za smanjenje rizika, ukoliko je to potrebno. Pacijenti koji i dalje nose rizik od neriješenih simptoma, okidača i komorbiditeta, imaju pristup pristupačnoj personaliziranoj podršci kako bi se smanjili napadi. Dodatno, izvješća o farmakogenetici omogućuju pacijentima i njihovim liječnicima izbjegavanje poznatih interakcija lijekova te genetskih čimbenika rizika. Sve u svemu, Mymee program označava revoluciju u liječenju autoimunih bolesti, pružajući svakom pacijentu individualiziranu i učinkovitu njegu. Slika 2.3 predstavlja izgled aplikacije Mymee.



Slika 2.3. *Prikaz izgleda Mymee aplikacije [35]*

2.6. Prikaz stanja u području

Stanje u području razvoja programskih rješenja u medicinske svrhe je u sve većem porastu gdje postoje aplikacije za zdravstvo, odnosno mobilno zdravlje (engl. *mHealth*) koje se koriste u suradnji sa zdravstvenim stručnjacima kako bi pružile savjete, pomoć u dijagnozi ili preporučile terapiju. Prema [36], globalno tržište mobilnog zdravlja u 2022. godini vrijedilo je 56,8 milijardi američkih dolara i očekuje se godišnji rast od 10,8% od 2023. do 2030. Glavni

pokretač rasta je povećani fokus na osobno zdravlje i kondiciju putem pametnih uređaja i nosivih tehnologija. Rast svijesti o aplikacijama za praćenje pacijenata na daljinu, upravljanje lijekovima, povećana prisutnost kroničnih bolesti i visoka preporuka zdravstvenih stručnjaka za korištenje mobilnih zdravstvenih aplikacija za poboljšanje kliničkih rezultata i uključenost pacijenata dodatno će potaknuti brži godišnji rast ovog segmenta. Kao što je navedeno u [37], mnoge studije su procijenile korisnost mobilnog zdravlja kao alata za poboljšanje upravljanja kroničnim bolestima putem praćenja i povratnih informacija, edukativnih i životnih intervencija, kliničke podrške u donošenju odluka, pridržavanja terapije lijekovima, procjene rizika i podrške rehabilitaciji. Telezdravstvo i na webu zasnovane tehnologije imaju obećavajuće rezultate koji se kreću od olakšanja simptoma povezanih s bolešću, poboljšanja pridržavanja terapiji lijekovima te smanjenja stopa ponovnog hospitaliziranja i smrtnosti. Dosadašnji dokazi pokazali su da mobilno zdravlje pruža koristi pacijentima s kroničnim bolestima koje su barem jednako dobre kao i tradicionalni oblici zdravstvene skrbi. Nova generacija uređaja za mobilno zdravlje temeljenih na različitim tehnologijama vjerojatno će pružiti učinkovitije i personalizirane programe zdravstvene skrbi za pacijente.

Sadašnje stanje unutar informacijske i komunikacijske tehnologije (ICT) u vezi s autoimunim bolestima i generiranjem preporuka za terapiju i prehranu dinamično napreduje. Pojavila su se brojna programska rješenja i inovativne metode tijekom proteklog desetljeća, stvarajući obećavajući temelj za unapređenje dijagnostike i liječenja. U pogledu programske podrške, dostupne su web i mobilne aplikacije koje pružaju informacije o autoimunim oboljenjima, njihovim simptomima te općim smjernicama za terapiju. No, prilagođene preporuke za personaliziranu terapiju i prehranu još uvijek su u fazi razvoja. Većina trenutnih sustava oslanja se na ograničene skupove podataka i algoritama, što može utjecati na preciznost i važnost dobivenih preporuka. Kao što se može vidjeti u [38], metode strojnog učenja i dubokog učenja sve više se primjenjuju kako bi unaprijedile dijagnostiku i predviđanje ishoda autoimunih bolesti. Ove tehnike mogu analizirati velike količine podataka kako bi otkrile obrasce i trendove koji su teško uočljivi konvencionalnim metodama. U sferi preporuka za terapiju i prehranu, postoje algoritmi koji uzimaju u obzir medicinske i nutricionističke informacije kako bi generirali opće smjernice. Međutim, personalizirane preporuke koje obuhvaćaju genetske, okolišne i individualne čimbenike i dalje predstavljaju izazov. Iako postoji značajan napredak u ICT-u, programski alati i pristupi vezani uz autoimune bolesti i pružanje personaliziranih preporuka za terapiju i prehranu i dalje se razvijaju kako bi se osigurala pouzdana i učinkovita podrška pacijentima i stručnjacima unutar medicinske zajednice.

3. IDEJNO RJEŠENJE, MODEL I GRAĐA WEB SUSTAVA ZA DIJAGNOZU AUTOIMUNIH BOLESTI I STVARANJE PREPORUKA ZA LIJEČENJE I PREHRANU

Prilikom izrade ove web aplikacije ključno je definirati model, funkcionalne i nefunkcionalne zahtjeve te postupke koji će se koristiti prilikom dijagnoze autoimune bolesti, stvaranja preporuka prikladnih lijekova i prehrane te postupke koji će se izvoditi za testiranje takvog sustava. Na taj način se postiže jasno razumijevanje što aplikacija treba postići i kako treba funkcionirati te se osigurava kvaliteta aplikacije.

3.1. Funkcionalni i nefunkcionalni zahtjevi na web sustav

Analiza zahtjeva ključan je korak u ocjenjivanju uspješnosti sustava ili programskog projekta. Prema [19], zahtjevi se obično dijele na dvije glavne kategorije: funkcionalne i nefunkcionalne. Funkcionalni zahtjevi predstavljaju osnovne funkcionalnosti koje krajnji korisnik zahtijeva od sustava. Tu se definiraju ulazi koje sustav prima, operacije koje sustav treba izvesti i očekivan izlazi. Funkcionalni zahtjevi proizlaze iz korisničkih potreba i jasno su vidljivi u konačnom proizvodu. Nefunkcionalni zahtjevi se odnose na kvalitativna ograničenja koja sustav mora ispuniti sukladno projektu. Prioriteti i opseg ovih aspekata variraju ovisno o specifičnostima projekta. Primjeri nekih nefunkcionalnih zahtjeva su prenosivost, sigurnost, održivost, pouzdanost te performanse.

3.1.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi koje ovaj web sustav ostvaruje su:

- Mogućnost prijave administratora i liječnika
- Pohranjivanje informacija o prijavi unutar Spring Security kolačića
- Automatsko odjavljivanje korisnika nakon isteka vremena valjanosti tokena unutar kolačića
- Kreiranje profila liječnika od strane administratora
- Mogućnost izmjene i brisanja liječnika od strane administratora
- Arhiviranje i brisanje postojećih pacijenata od strane administratora
- Kreiranje i izmjena profila pacijenta od strane liječnika
- Dodavanje pacijenta liječniku koji je bio dodijeljen drugom liječniku
- Uklanjanje pacijenta s određenog liječnika gdje pacijent mora i dalje postojati u sustavu
- Mogućnost dodavanja simptoma bolesti pacijentu od strane liječnika

- Sustav mora vratiti dijagnozu bolesti na osnovi simptoma i karakteristika pacijenta ukoliko liječnik to zatraži
- Liječnik mora moći dijagnosticirati bolest bez pomoći sustava
- Sustav mora, ukoliko liječnik zatraži, preporučiti lijekove i prehranu koju pacijent treba jesti i izbjegavati s obzirom na dijagnosticiranu bolest od strane sustava i karakteristike pacijenta
- Liječnik mora moći unijeti propisane lijekove i propisan plan prehrane neovisno o preporuci sustava
- Liječnik mora moći dodavati, izmjenjivati te brisati rezultate praćenja tijeka bolesti i nuspojava pacijenta
- Sustav mora, na osnovi simptoma, vratiti liječniku trenutno stanje pacijenta prilikom praćenja rezultata tijeka bolesti
- Odgovarajuće rješenje za bazu podataka

Web sustav započinje prijavom gdje se u sustav mogu prijaviti liječnik ili administrator popunjavajući formu za e-mail i lozinku. Spring Security koristi sesiju za upravljanje prijavom korisnika koja omogućava održavanje stanja korisnika tijekom više zahtjeva tako da korisnik ostaje prijavljen dok je sesija aktivna. Kada korisnik zatvori preglednik, bude neaktivan određeno vrijeme ili se odjavi, sesija se uništava i korisnik se odjavljuje. Nadalje, administratora kreira sustav te kao takav on ima pravo kreirati, izmijeniti te brisati liječnike kao korisnike web sustava. Osim toga, administrator ima uvid u pacijente gdje ima pravo arhivirati pacijenta prilikom čega će pacijent biti uklonjen sa svoga liječnika te se više neće moći pridijeliti niti jednom liječniku. Također, administrator ima mogućnost brisanja pacijenata iz sustava.

Liječnik, za razliku od administratora, može kreirati pacijentov profil unoseći osnovne informacije o pacijentu, poput imena, prezimena, OIB-a, adrese stanovanja, e-mail-a, broja telefona, spola, dobi, informacija o trudnoći i dojenju te može dodati alergije na prehranu ukoliko ih pacijent ima. Svaki pacijent je jedinstven po OIB-u, stoga postoji validacija OIB-a prilikom kreiranja pacijenta kako bi se omogućilo dodjeljivanje pacijenta liječniku ukoliko pacijent već postoji u sustavu te, u suprotnom, omogućilo stvaranje novog pacijenta. Također, liječnik može izmijeniti profil pacijenta te ga može ukloniti s liste svojih pacijenata prilikom čega pacijent i dalje postoji u sustavu te može biti dodijeljen istom ili drugom liječniku.

Svaki pacijent ima svoj medicinski karton gdje liječnik ima uvid u pacijentove simptome, dijagnozu bolesti, propisane lijekove i prehranu te u rezultate praćenja tijeka liječenja i

nuspojava. Unutar medicinskog kartona, liječnik unosi pacijentove simptome te može sam dijagnosticirati bolest ili, na liječnikov zahtjev, sustav mu može vratiti dijagnozu bolesti. Čak i kada sustav vrati dijagnozu bolesti, liječnik ju može izmijeniti te time gubi daljnju pomoć prilikom preporuke lijekove i prehrane. Ukoliko liječnik odluči koristiti pomoć sustava, tada mu sustav vraća preporuku lijekova s obzirom na bolest te s obzirom na pacijentove karakteristike i vraća mu preporuku prehrane koju bi pacijent trebao konzumirati te prehranu koju bi pacijent trebao izbjegavati. Bitno je za napomenuti da je liječnikova odluka konačna i da liječnik sam unosi propisane lijekove i prehranu. Unutar medicinskog kartona, liječnik ima mogućnost pratiti rezultate liječenja i nuspojave bilježeći simptome koje pacijent ima, specifične simptome ili nuspojave, stanje pacijenta te sustav na osnovi simptoma vraća konačan zaključak rezultata za svaki unos.

Za ostvarivanje prethodno opisanih funkcionalnosti, neophodno je imati odgovarajuću bazu podataka koja je sposobna pohraniti različite vrste informacija. Baza podataka će biti konstruirana tako da omogući odvojeno čuvanje podataka o ulogama korisnika, korisnicima, bolestima, simptomima, lijekovima, prehrani te njihovim međusobnim vezama, svaka u zasebnim tablicama.

3.1.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi predstavljaju temelj za stvaranje kvalitetne web aplikacije koja zadovoljava visoke standarde performansi, sigurnosti i korisničkog iskustva. Ovaj web sustav ispunjava slijedeće nefunkcionalne zahtjeve:

- Sigurnost
- Prilagodljivost
- Pouzdanost
- Upotrebljivost
- Robusnost

Sigurnost ima iznimno važnu ulogu kao osnovni nefunkcionalni zahtjev koji štiti web aplikaciju od raznolikih prijetnji i neovlaštenih pristupa. Ovaj aspekt obuhvaća sve tehnike i strategije koje osiguravaju cjelovitost, povjerljivost i dostupnost podataka, te štite aplikaciju od različitih oblika napada. Unutar ovoga web sustava su implementirane metode za prijavu u sustav gdje se provjerava ispravnost korisničkog e-mail-a i lozinke i time onemogućava pristup neovlaštenim osobama. Nadalje, omogućeno je šifriranje osjetljivih informacija poput lozinki te adekvatno upravljanje korisničkim sesijama gdje nakon isteka kolačića korisnik biva odjavljen. Prilagodljivost se odnosi na sposobnost web aplikacije da se uspješno prilagodi

različitim okruženjima i uređajima na kojima korisnici pristupaju aplikaciji. Ovaj zahtjev je ostvaren na način da se web sustav prilagođava različitim rezolucijama ekrana i web preglednicima. Pouzdanost je, također, jedna od ključnih karakteristika ovoga sustava gdje se osigurava dosljedno i stabilno izvođenje web aplikacije. Izuzetno je važan i nefunkcionalan zahtjev upotrebljivost gdje se osiguralo da web sustav bude intuitivan i lako upotrebljiv za korisnike. Upotrebljivost je ostvarena putem organizacije korisničkog sučelja, jasno označenih kontrola, intuitivne navigacije i brzog pristupa informacijama. U konačnici, robusnost označava sposobnost web aplikacije da se nosi s neočekivanim situacijama i nevaljanim ulazima te da nastavi pravilno funkcionirati. Ovaj zahtjev naglašava potrebu da aplikacija bude otporna na greške i da može ispravno reagirati čak i u nepredviđenim scenarijima, umanjujući rizik od padova ili ozbiljnih problema koji bi mogli utjecati na korisničko iskustvo.

3.2. Postupci dijagnoze autoimune bolesti

Kako bi se ispunio funkcionalan zahtjev gdje sustav treba moći dijagnosticirati bolest, treba se istražiti i pronaći najbolji pristup dijagnoze bolesti. Klasični pristupi dijagnozi često se oslanjaju na laboratorijske testove i medicinsku povijest pacijenta. Po uzoru na poglavlje 2.1. i prema [5, 6, 7, 8, 9, 10, 11, 12, 13, 14] predlaže se dijagnosticiranje bolesti na način da se kombiniraju pacijentovi simptomi, dob i spol te s obzirom na njih dođe do zaključka o autoimunoj bolesti. Ovaj dijagnostički pristup stavlja simptome pacijenta u središte dijagnostičkog procesa. Svaki simptom ima određenu težinu koja se temelji na stupnju učestalosti simptoma u okviru specifične autoimune bolesti. Na taj način je omogućeno preciznije identificiranje ključnih simptoma koji mogu ukazivati na prisutnost određene bolesti te simptomi koji su češći kod određenih autoimunih bolesti dobivaju veću težinu. Dakle, proces dijagnoze bolesti započinje filtriranjem bolesti po simptomima. Ako bolest sadržava simptom koji pacijent ima, tada će se ta bolest uzeti u obzir. Nakon početnog filtriranja, ukoliko se ne dođe do jedinstvenog rješenja o kojoj se bolesti radi, tada slijedi filtriranje po težinama simptoma. Nadalje, neke autoimune bolesti su puno češće kod žena ili se u većini pojavljuju samo kod žena, stoga ukoliko se prilikom računanja autoimune bolesti na osnovi težine simptoma ne dođe do konačnog zaključka o kojoj se bolesti radi, tada se dalje filtriraju bolesti s obzirom na spol pacijenta. Ukoliko niti nakon filtriranja po spolu ne postoji jedna bolest kao konačno rješenje, tada se uzima u obzir dob pacijenta. Dob je zadnja varijabla po kojoj se filtriraju autoimune bolesti jer, iako postoji prosjek godina kada se određena autoimuna bolest najčešće pojavljuje, zapravo bilo kada u životu pacijent može dobiti autoimunu bolest. Ovakav pristup dijagnozi autoimunih bolesti može pripomoći liječniku i na taj način rezultirati bržim

dijagnosticiranjem, boljim usmjeravanjem pacijenta prema pravom liječenju i, konačno, optimiziranim ishodima liječenja.

3.3. Postupci stvaranja preporuka prikladnih lijekova i prehrane

S ciljem ispunjenja zahtjeva za funkcionalnošću, koja podrazumijeva sposobnost sustava da temeljem dijagnosticiranog oboljenja preporučuje odgovarajuće lijekove i prehrambene opcije, potrebno je provesti istraživanje i identificirati optimalan pristup.

3.3.1. Višekriterijski postupak stvaranja preporuka

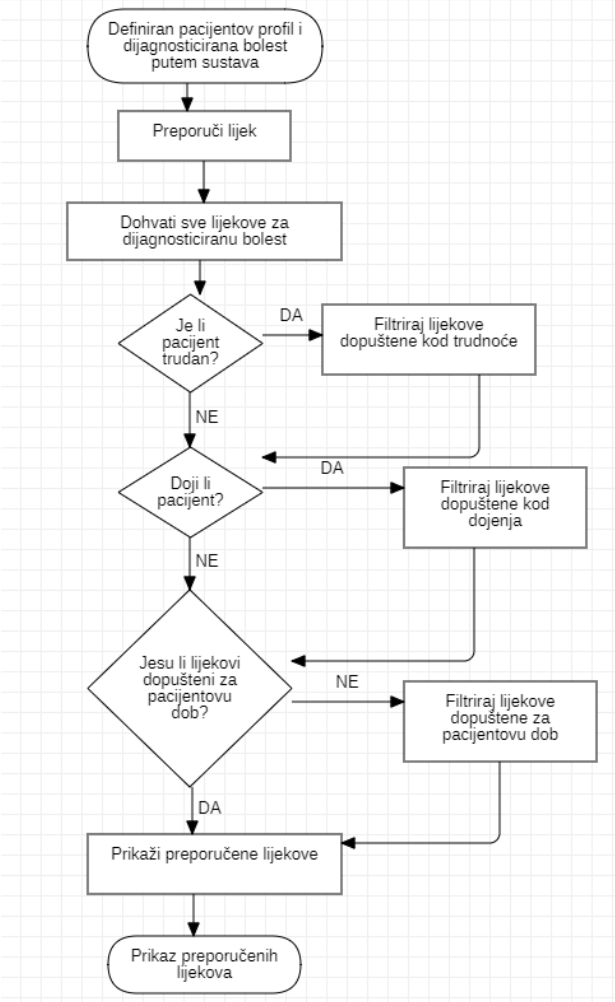
Višekriterijsko donošenje odluka je popularna grana procesa donošenja odluka, u kojoj donositelji odluka trebaju izabrati opciju na temelju niza kriterija odlučivanja. Prema [20], višekriterijsko donošenje odluka je moguće podijeliti u dvije kategorije: višeciljno odlučivanje i višeatributno odlučivanje, to jest višekriterijska analiza. Višeciljni model donošenja odluka je prikladan za situacije koje se nazivaju "dobro strukturiranim" problemima. Ovdje se radi o situacijama u kojima su trenutačno stanje, željeno konačno stanje (ciljevi) i metode za postizanje tih ciljeva jasno definirani. Model uključuje velik broj mogućih rješenja, često nepoznatih na početku, i uz ograničenja. Odluka o najboljem rješenju postiže se putem rješavanja matematičkog modela. Model višeatributnog odlučivanja ili višekriterijske analize koristi se za situacije koje se nazivaju "loše strukturiranim" problemima. Ovdje se radi o situacijama u kojima su ciljevi izuzetno složeni, često nejasno definirani te postoje mnoge neizvjesnosti. Slaba strukturiranost ovakvih problema sprječava postizanje jedinstvenog rješenja. U okviru ovog modela, postoji ograničen broj mogućih rješenja koji su poznati na početku. Rješenje problema postiže se identifikacijom najbolje opcije ili skupa dobrih opcija u odnosu na definirane atribute/kriterije i njihove važnosti.

Objasnivši dvije podjele višekriterijskog postupka stvaranja preporuka, vidljivo je da će više atributno, to jest višekriterijska analiza, najviše odgovarati prilikom stvaranja preporuka za lijekove i prehranu. Kao što je već prethodno objašnjeno, na temelju prethodno definiranih atributa iliti kriterija, što će u ovom web sustavu biti dijagnosticirana bolest te određene karakteristike pacijenta se donosi rješenje za preporuke.

3.4. Postupak odabira lijekova i prehrane u skladu s mogućnostima i ograničenjima pacijenta

Kao što je već u prethodnom poglavlju rečeno, prilikom postupka odabira preporučenih lijekova i prehrane koristiti će se postupak višekriterijske analize. Prije postupka preporuke, važno je pravilno definirati pacijentov profil od strane liječnika i dijagnosticirati bolest od

strane sustava. Budući da ovaj rad pokriva nekolicinu najčešćih autoimunih bolesti, ukoliko liječnik odluči sam dijagnosticirati bolest, tada neće imati pomoć preporuka za lijekove i prehranu jer ovaj web sustav pokriva preporuku za nekolicinu autoimunih bolesti koje su u drugom poglavlju spomenute. Uzevši u obzir pacijentova stanja i bolest omogućuje se personalizirana preporuka terapije i prehrane što može olakšati liječniku davanje ispravne terapije. Kao što je navedeno u [16], prilikom preporuke lijekova uzimaju se u obzir kriteriji pacijentovog stanja, kao što su dijagnosticirana bolest, trudnoća, dojenje i dob. Određeni lijekovi su kontraindicirani za trudnice ili dojilje zbog potencijalnih rizika za plod ili novorođenče, stoga, ako postoji bilo kakva indikacija da bi pacijent mogao biti trudan ili dojiti, takvi lijekovi neće biti preporučeni. U slučajevima gdje je potrebno izbjeći propisivanje određenih lijekova trudnicama ili dojiljama, ali u određenim situacijama se mogu prepisati ukoliko je to neophodno, sustav će generirati upozoravajuće napomene liječniku kako bi se osigurala sigurna i ispravna terapija. Nadalje, kao kriterij prilikom preporuke lijekova se uzima i dob pacijenta jer su određeni lijekovi zabranjeni za određenu dobnu skupinu, najčešće malu djecu, te se tada takvi lijekovi neće preporučiti liječniku prilikom propisivanja terapije. Prikaz tijeka preporuke lijekova se može vidjeti na slici 3.1. Bitno je za napomenuti da, prilikom preporuke lijeka, sustav samo pomaže liječniku te je liječnik odgovoran za propisanu terapiju.



Slika 3.1. *Dijagram toka preporuke lijeka*

Prilikom preporuke prehrane također se uzimaju u obzir već definirani kriteriji, a to su dijagnosticirana bolest i alergije na prehranu. S obzirom na bolest koju pacijent ima i s obzirom na prehranu na koju je alergičan, preporučit će se prehrana koju bi pacijent trebao jesti filtrirajući onu prehranu na koju pacijent nije alergičan kako bi se liječniku olakšala preporuka prehrane. Nadalje, s obzirom na bolest će se preporučiti prehrana koju pacijent treba izbjegavati. U konačnici, liječnik propisuje prehranu za pacijenta te mu sustav samo pripomaže u savjetovanju, odnosno preporuci.

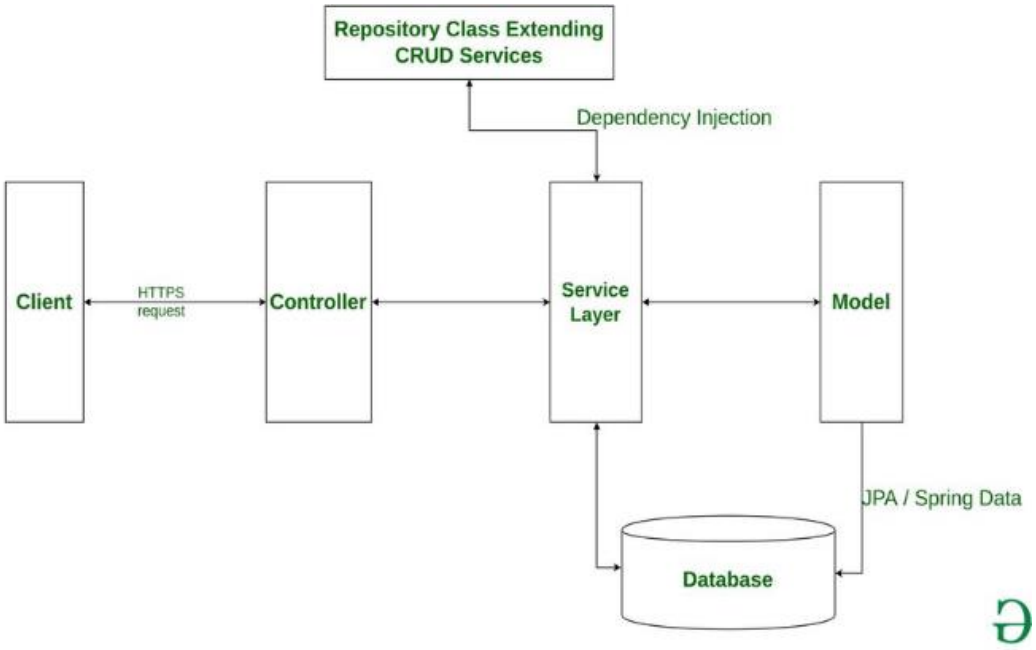
3.5. Programska arhitektura web sustava

Općenito, ovaj web sustav je ostvaren koristeći MVC (engl. *Model View Controller*) arhitekturu. Po uzoru na [32], unutar MVC arhitekture, aplikacija je podijeljena na tri glavne komponente: model, prikaz (engl. *view*) i kontroler. Model predstavlja logičke podatke koji se mogu prenositi između prikaza i kontrolera te mogu sadržavati poslovnu logiku. Podaci, koji predstavljaju model, mogu biti jedan objekt ili zbirka objekata. Prikaz predstavlja sve logičke

dijelove korisničkog sučelja aplikacije te kontroler predstavlja sučelje između komponenata modela i prikaza te služi za obradu svih podataka, poslovne logike i dolaznih zahtjeva.
Detaljnije govoreći, a po uzoru na [21], ovaj sustav koristi Spring Boot okvir koji je baziran na Java programskom jeziku te je otvoren za korištenje. Ovaj okvir se koristi za izgradnju samostalnih i produkcijski spremnih aplikacija koje se jednostavno pokreću te su potrebne minimalne konfiguracije prilikom postavljanja okruženja za razvoj aplikacija. Koristeći ovaj okvir, omogućen je iznimno lakši i razumljiviji razvoj aplikacija, povećava se produktivnost te smanjuje vrijeme razvoja. Nadalje, u Spring Boot-u je sve automatski konfigurirano, jedino treba znati za koje funkcije koristiti te konfiguracije. Koristan je za razvoj REST API-ja te ima ugrađen Tomcat poslužitelj koji omogućuje dostupnost Java web aplikacije, koja je postavljena unutar njega, putem HTTP zahtjeva. Postoje četiri glavna sloja u Spring Boot-u:

- Prezentacijski sloj – sastoji se od prikaza, obrađuje HTTP zahtjeve, provjerava autentičnost zahtjeva, prevodi JSON parametar u objekt i obrnuto
- Poslovni sloj – sastoji se od klasa usluga i koristi usluge koje pruža sloj pristupa podacima, obrađuje svu poslovnu logiku, obavlja autorizaciju i provjeru valjanosti
- Sloj postojanosti - sadrži svu logiku pohrane baze podataka, odgovoran je za pretvaranje poslovnih objekata u red baze podataka i obrnuto
- Sloj pristupa podacima – sadržava operacije nad bazom podataka (kreiranje, dohvaćanje, ažuriranje, brisanje)

Na slici 3.2, prema [21], je prikazana arhitektura Spring Boot-a gdje je vidljivo da nakon što klijent na prikazu zatraži HTTP zahtjev, zahtjev je dalje proslijeđen kontroleru koji mapira zahtjev, obrađuje ga i poziva logiku poslužitelja. Poslovna logika se odvija u klasama usluga, odnosno na sloju usluga. Ondje su uz pomoć JPA (engl. *Java Persistence Library*), koji predstavlja sloj postojanosti, podaci preslikani u klasu modela i nad njima se izvodi sva logika. Za kraj se vraća odgovor kontrolera s odgovarajućim prikazom i podacima.



Slika 3.2. *Arhitektura Spring Boot-a [21]*

3.6. Postupci ručnog i automatiziranog testiranja web sustava

Kako bi se osigurala ispravnost rada, spriječilo ispadanje sustava iz rada te osiguralo pokrivanje svih funkcionalnosti koje su zatražene, važno je provesti kvalitetno testiranje sustava. Testovi koji će se provesti su ručno i automatizirano testiranje sustava na strani klijenta.

3.6.1. Ručno testiranje

Prema [22], ručno testiranje je ključna faza u razvoju aplikacija koja predstavlja nezamjenjivu ulogu u osiguranju njihove visoke kvalitete prije nego što postanu dostupne korisnicima. Ovom pristupu cilj je detaljno istražiti funkcionalnosti i korisnička sučelja aplikacija kako bi se identificirale potencijalne skrivene greške i probleme koji bi mogli narušiti ukupno korisničko iskustvo. Ručno testiranje donosi niz koristi te se sastoji od ključnih koraka koji ga čine ključnim dijelom procesa razvoja. U svijetu programskog inženjerstva, ručno testiranje ima važnu ulogu u stvaranju cjelovite slike o performansama i ispravnosti aplikacija. Time se obuhvaćaju raznovrsni scenariji koje automatski testovi ne bi mogli u potpunosti pokriti.

Testeri imaju sposobnost simuliranja stvarnih interakcija korisnika, pružajući dublji uvid u način na koji aplikacija reagira u različitim situacijama. Dodatna vrijednost ručnog testiranja je njegova sposobnost intuitivnog razumijevanja kvalitete korisničkog iskustva. Ljudski testeri mogu uočiti neprimjetne probleme u korisničkom sučelju i upotrebljivosti, doprinoseći unapređenju cjelokupnog iskustva korisnika. Osim očiglednih grešaka, ručno testiranje se također iznimno dobro pokazalo u otkrivanju neočekivanih problema. Testeri mogu istražiti

aplikaciju dublje, otkrivajući potencijalne probleme koji se ne bi mogli predvidjeti unaprijed. Ova dubinska analiza omogućava otkrivanje ranjivosti koje bi inače moguće predvidjeli.

Važno je napomenuti da je temeljito pripremljeni niz testnih scenarija ključan za uspješno ručno testiranje. Ovi scenariji simuliraju stvarne situacije i interakcije korisnika kako bi se osiguralo obuhvaćanje svih relevantnih aspekata aplikacije. Tokom samog izvođenja testova, pažljivo se prati svaki korak, bilježeći rezultate i evidentirajući potencijalne probleme. Ručno testiranje ne samo da identificira greške i probleme u funkcionalnosti, već također procjenjuje performanse, stabilnost i ukupno korisničko iskustvo aplikacije. Kroz ovaj pristup, osigurava se da aplikacija zadovoljava zadane zahtjeve i specifikacije te pruža pozitivno i kvalitetno iskustvo korisnicima. Konačno, ručno testiranje je ključna komponenta razvojnog procesa programskih rješenja. Kroz intuitivno ispitivanje, otkrivanje neočekivanih problema te detaljnu analizu performansi i korisničkog iskustva, ručno testiranje doprinosi stvaranju stabilnih, funkcionalnih i korisnički orijentiranih aplikacija koje odgovaraju visokim standardima kvalitete.

3.6.2. Automatizirano testiranje

Po uzoru na [23], automatizirano testiranje ima ključnu ulogu u razvoju programskih rješenja. Koristi se posebnim alatima i skriptama za izvođenje unaprijed definiranih testnih slučajeva i procjenu funkcionalnosti aplikacija. Ova metoda pruža niz prednosti u smislu olakšavanja procesa, proširenja opsega testiranja i ubrzanja postupaka testiranja. Automatizirano testiranje zauzima važno mjesto u životnom ciklusu razvoja programa i znatno pridonosi održavanju kvalitete i učinkovitosti proizvoda. Jedna od glavnih prednosti automatiziranog testiranja je učinkovitost i brzina. Automatizirani testovi se izvode brže od ručnih testova, pružajući brzu povratnu informaciju o kvaliteti programskog proizvoda. Ova brzina je posebno korisna za projekte s čestim izmjenama koda ili strogim vremenskim rokovima za izdanje. Automatizirano testiranje također osigurava ponovljivost i dosljednost. Testovi se mogu točno i dosljedno ponoviti, čime se osigurava primjena istih uvjeta testiranja pri svakom izvršenju. To eliminira varijabilnost koja može nastati kod ručnog testiranja. Osim toga, automatizirano testiranje se ističe po obuhvatu. Može obuhvatiti različite scenarije i testne slučajeve koji bi bili teško ili vremenski zahtjevno izvedivi ručno. To rezultira širim opsegom testiranja i sveobuhvatnijom procjenom programa. Još jedna prednost leži u testiranju regresije.

Automatizacija je nezamjenjiva za testiranje regresije, što uključuje provjeru da nove izmjene u kodu ne utječu negativno na postojeće funkcionalnosti. Automatizacija također omogućava paralelno testiranje gdje je omogućeno istovremeno izvođenje više testova što štedi vrijeme i

resurse, posebno kod složenih aplikacija ili velikih skupova testova. Automatizirani testovi također se ističu točnošću. Izvršavaju zadatke precizno prema skriptama, smanjujući mogućnost ljudskih pogrešaka koje se mogu dogoditi kod ručnog testiranja. Unatoč tim prednostima, postoje izazovi i razmatranja vezana uz automatizirano testiranje. Postavljanje početne automatizacije i izrada skripti zahtijevaju početno ulaganje vremena i truda, što može privremeno odgoditi trenutačne prednosti automatizacije. Održavanje je stalni izazov kako program napreduje. Skripte za automatizaciju moraju se ažurirati kako bi odražavale promjene u aplikaciji. Pažljivo održavanje je ključno kako bi se osiguralo da testovi ostanu relevantni. Neki scenariji koji uključuju kompleksne interakcije korisničkog sučelja mogu biti izazovni za učinkovitu automatizaciju. Također, važno je napomenuti da iako je automatizacija prikladna za ponavljajuće i stabilne testne slučajeve, postoje situacije koje još uvijek imaju koristi od ručnog testiranja, posebno oni koji zahtijevaju istraživačko testiranje i subjektivnu procjenu. Automatizirano testiranje zahtijeva posebne vještine u izradi skripti i korištenju alata za automatizaciju testiranja, što možda nije univerzalno dostupno unutar tima za testiranje. Sve u svemu, automatizirano testiranje je snažan pristup koji značajno poboljšava kvalitetu i učinkovitost razvoja programa. Poboljšava procese, osigurava dosljednost i opsežnost te pomaže u ranom otkrivanju regresija. No, pažljivo planiranje, izvršenje i održavanje su ključni kako bi se ostvarili optimalni rezultati i iskoristio puni potencijal automatizacije.

4. PROGRAMSKO RIJEŠENJE OSTVARENOG WEB SUSTAVA

Svaki web sustav je podijeljen na korisnički dio i poslužiteljski dio. Korisnički dio predstavlja vidljiv dio web aplikacije s kojim korisnik ostvaruje različite interakcije putem internetskih preglednika. Poslužiteljski dio web sustava predstavlja logiku koja omogućava funkcioniranje web aplikacije te se izvršava obrada podataka i poslovne logike, komunikacija s bazama podataka i osiguravanje odgovarajućih resursa za korisničke zahtjeve. To je središnji dio aplikacije koji nije vidljiv korisnicima, ali je ključan za pružanje funkcionalnosti i podrške korisničkom dijelu.

4.1. Korišteni programski jezici i alati

Za razvoj bilo kojeg programskog rješenja, neophodni su programski jezici i alati koji imaju ključan utjecaj na kvalitetu i učinkovitost krajnjeg proizvoda. Važno je znati odabrati prikladan programski jezik i alate kako bi se što jednostavnije razvio proizvod i omogućila podrška za tražene funkcionalnosti.

4.1.1. Programski jezici

Za razvoj korisničkog dijela ostvarenoga web sustava, korišteni su HTML, CSS i JavaScript jezici. Prema [24], HTML (engl. *HyperText Markup Language*) predstavlja prezentacijski jezik za izradu web aplikacija dok CSS (engl. *Cascading Style Sheets*) predstavlja stilski jezik koji se koristi za opisivanje i upravljanje izgledom dokumenta napisanog pomoću HTML jezika. JavaScript je programski jezik za web razvoj koji ima funkciju ažuriranja i promijene HTML i CSS elemenata te može manipulirati podacima.

Kod razvoja poslužiteljskog dijela ovoga web sustava, korišten je programski jezik Java. Prema [25], Java se široko primjenjuje za različite vrste aplikacija, uključujući web, mobilne, desktop te poslužiteljske aplikacije, no također se primjenjuje i za razvoj igrica, Internet objekata, umjetne inteligencije. Nadalje, neovisna je o platformi što znači da se programi napisani u Javi mogu izvoditi na različitim operacijskim sustavima. Java je također prepoznata po svojoj snažnoj podršci za objektno orijentirano programiranje i osiguranju sigurnosnih mehanizama koji pružaju izolaciju između aplikacija i sprečavaju potencijalno rizične operacije.

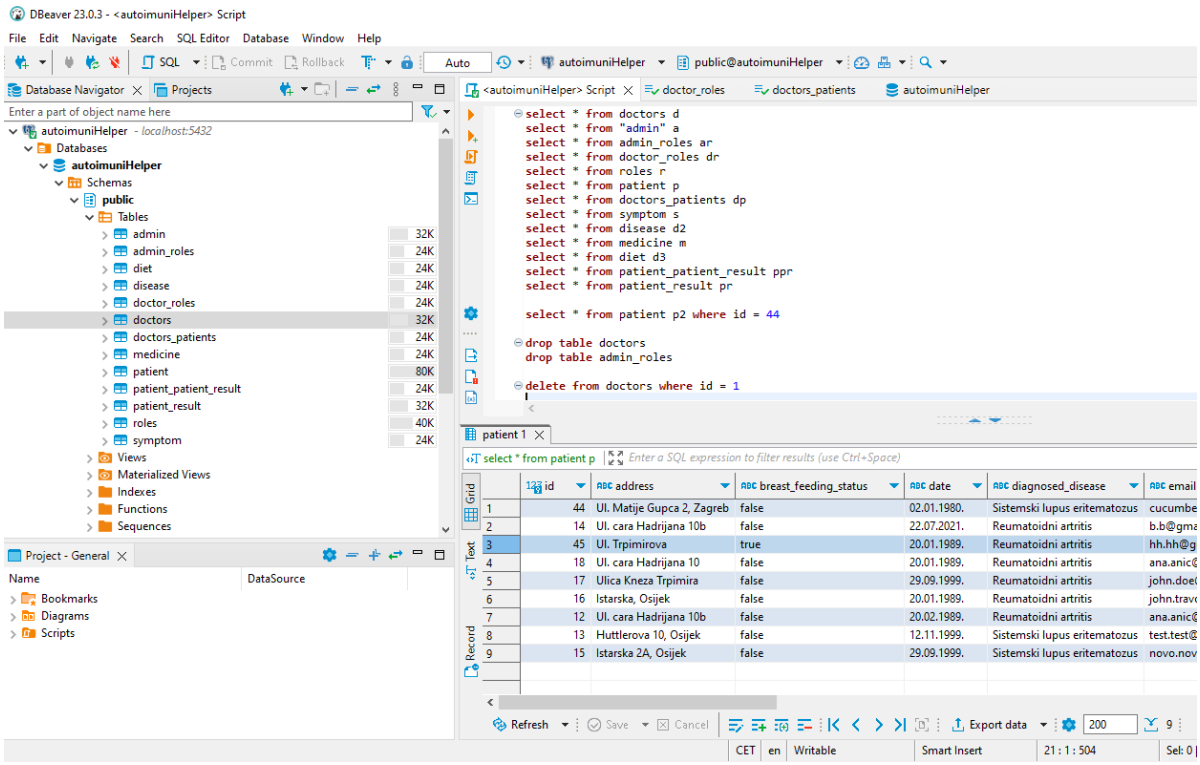
4.1.2. Programski alati

Po uzoru na [26], programski alat može biti bilo koji program ili alat koji pomaže programerima ili stvarateljima programa pri stvaranju, uređivanju, otklanjanju grešaka, održavanju i/ili obavljanju bilo kojeg zadatka vezanog za programiranje ili razvoj. U sklopu razvoja ovoga web

sustava korišteno je mnogo programskih alata koji se mogu podijeliti na programske okvire i programska razvojna okruženja.

Programski okviri korišteni u ovom radu su Bootstrap, Spring Boot te Thymeleaf. Prema [24], Bootstrap je najpopularniji CSS okvir za razvoj responzivnih i mobilnih web stranica gdje je u ovome radu korištena najnovija verzija, odnosno Bootstrap 5. Ovaj okvir pruža set stilova, komponenata i skripti koje olakšavaju izradu privlačnih korisničkih sučelja. Spring Boot je okvir koji se temelji na mikrousluzi i izradama aplikacija spremnih za proizvodnju u što kraćem vremenu, a sam okvir je detaljnije objašnjen u 3.5. podnaslovu. Prema [27], Thymeleaf je predložak temeljen na Javi za web i samostalna okruženja, sposoban za obradu HTML-a, XML-a, JavaScripta, CSS-a pa čak i običnog teksta. Ovaj predložak omogućuje paralelni rad na poslužiteljskoj i korisničkoj strani te može izravno pristupiti objektu i povezati ga s korisničkim sučeljem, odnosno HTML-om.

Programska okruženja korištena u ovome radu su: Visual Studio Code, IntelliJ IDEA, DBeaver te GitHub. Visual Studio Code je lagan i moćan uređivač izvornog koda koji je dostupan za različite operacijske sustave te dolazi s ugrađenom podrškom i proširenjima za razne programske jezike. IntelliJ IDEA je integrirano razvojno okruženje (IDE) za JVM jezike (Java, Kotlin, Scala, Groovy i mnoge druge) koje pruža pametno dovršavanje koda, statičku analizu koda te refaktoriranje i na taj način osigurava veću produktivnost i ugodnije iskustvo programera. DBeaver je besplatni alat za baze podataka koji je podržan na različitim operacijskim sustavima te podržava sve popularne SQL baze podataka kao što su MySQL, MariaDB, PostgreSQL, SQLite i mnoge druge. DBeaver pruža grafičko korisničko sučelje koje omogućava jednostavno pregledavanje baza podataka i njihovih objekata te izbor opcija putem intuitivnih izbornika. Uz mogućnost uređivanja podataka izravno unutar tablica, omogućuje brzu i učinkovitu promjenu podataka. Također, nudi alate za analizu baze podataka, uključujući stvaranje vizualnih dijagrama i upravljanje metapodacima. DBeaver ističe svoj SQL uređivač koji omogućuje jednostavno stvaranje i izvođenje ad hoc SQL upita te pruža napredne značajke poput sintaksne evidencije i automatskog dovršavanja. Na slici 4.1 je prikazan izgled DBeaver alata. U ovome radu je korištena PostgreSQL baza podataka koja, prema [28], može pohranjivati podatke kao objekte te sadrži četiri važne značajke: atomičnost, konzistentnost, izolaciju te izdržljivost, koje osiguravaju da transakcije ostanu točne i pouzdane tijekom procesa pisanja ili ažuriranja podataka. Slikom 4.1 prikazan je izgled DBeaver aplikacije

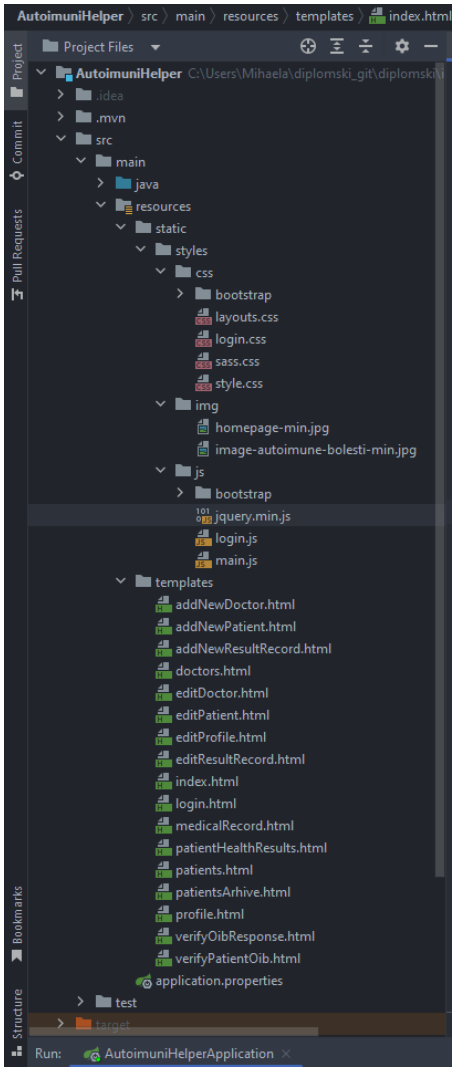


Slika 4.1. *DBeaver sučelje*

Nadalje, prilikom razvoja ovoga web sustava, korištena je platforma GitHub koji pruža suradnju, upravljanje verzijama koda i praćenje napretka projekta te se koristi kao okruženje za razvoj, upravljanje i suradnju na projektima.

4.2. Programsko rješenje web sustava

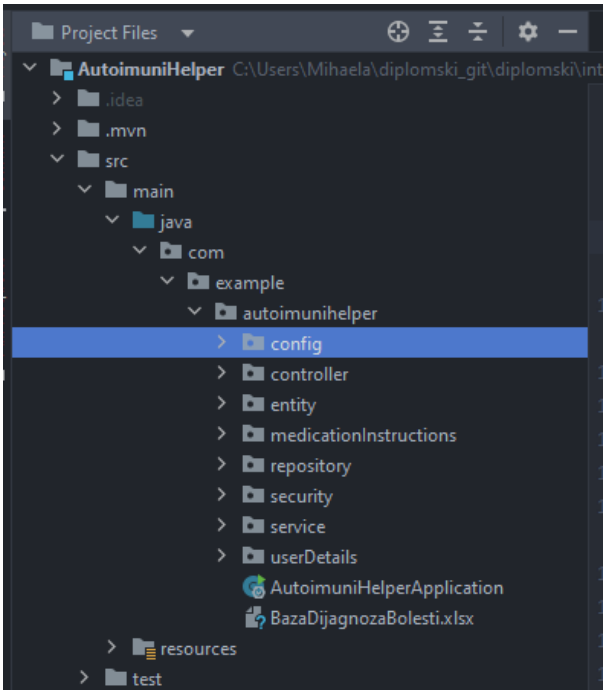
Prije nego li se započnu opisivati dijelovi aplikacije, dobro je spomenuti strukturu projekta. Kako bi se ostvarila MVC arhitektura te koristio Spring Boot s Thymeleaf okvirom, postoje određena pravila izgleda strukture projekta. Po standardu Spring Boot s Thymeleafom očekuje HTML datoteke, koji predstavljaju prikaze unutar MVC arhitekture, unutar src/main/resources/templates lokacije, ukoliko one nisu ondje definirane te ako u application.properties datoteci nije definiran drugi direktorij za pronalazak HTML datoteka, tada sustav neće moći pokrenuti aplikaciju, odnosno na internetskom pregledniku će se pojaviti stranica s greškom. Nadalje, po standardu CSS i JavaScript datoteke te slike moraju biti definirane na src/main/resources/static putanji, u suprotnom se neće moći koristiti. Na slici 4.2 se može vidjeti struktura projekta s naglaskom na putanju HTML, CSS te JavaScript datoteka i slika.



Slika 4.2. *Struktura projekta vezana za korisnički dio*

Objasnivši gdje su smješteni prikazi i dijelovi za korisnički dio web sustava, potrebno je objasniti strukturu na strani poslužitelja koja je vidljiva na slici 4.3. Poštujući MVC arhitekturu i slojeve od kojih je sastavljen Spring Boot, poslužiteljski dio je podijeljen na kontrolere, zadužene za primanje HTTP zahtjeva, pripremu modela i vraćanja prikaza s ispravnim podacima na internetskom poslužitelju, usluge gdje se odvija sva programska logika, sučelja za komunikaciju s bazama podataka, entitete koji su java objekti i predstavljaju podatke koji mogu biti prisutni u bazi podataka. Nadalje, za potrebe razvoja ovoga web sustava, kreirana je klasa za postavljanje početnih podataka u bazi. Zatim kreirane su klase koje koriste podatke o korisnicima kako bi se omogućila prijava te dohvaćanje podataka o korisniku prijavljenom u sustav te klasa gdje su definirana pravila sigurnosti. Također, u ovome dijelu prisutni su i podaci koji predstavljaju određenu bazu podataka gdje jedna sadržava upute o lijekovima, a druga sadrži pravila prilikom dijagnoze bolesti i preporuka lijekova i prehrane te njihov popis.

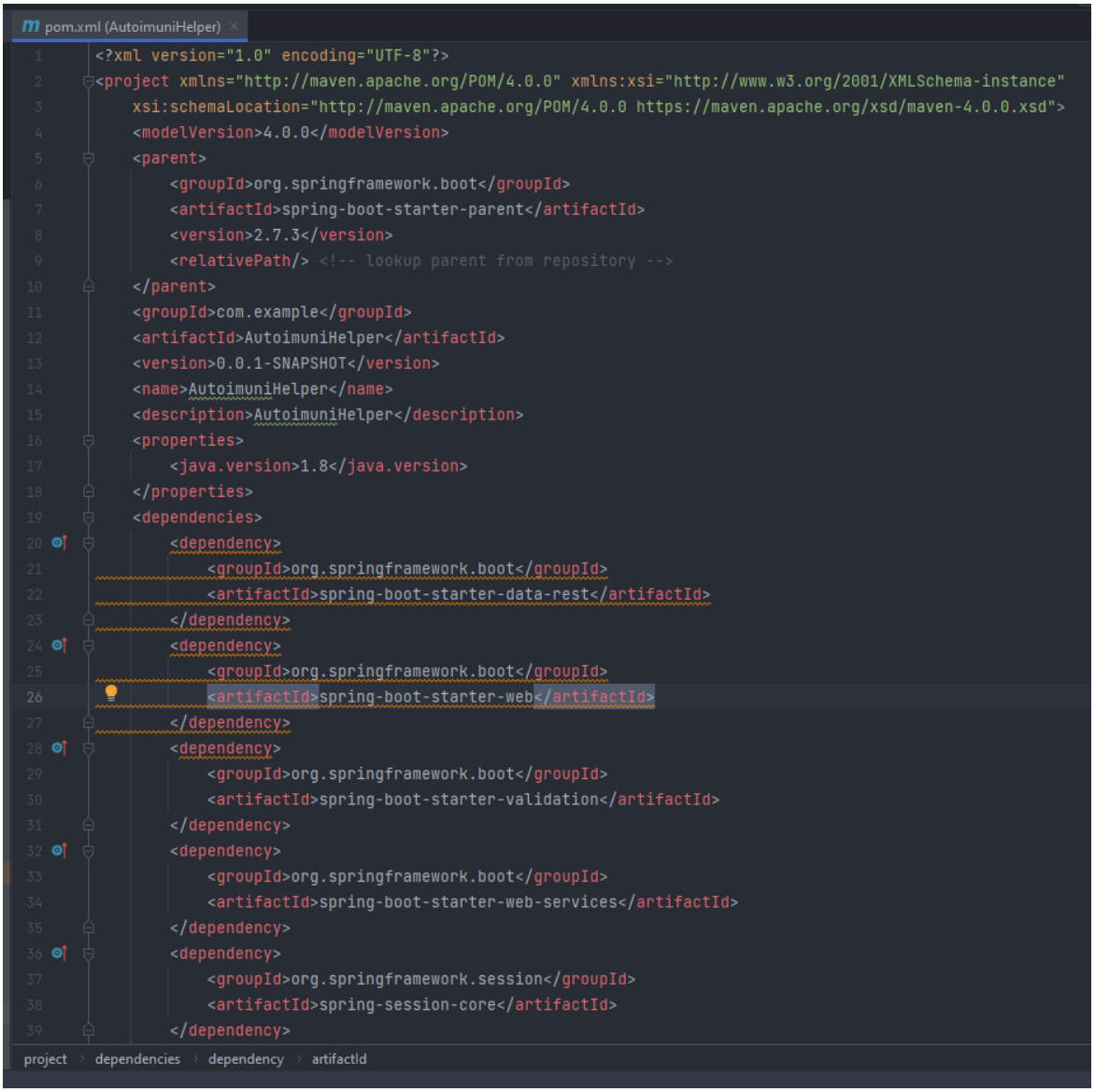
U konačnici, ovaj sustav ima definiranu klasu koja će napraviti konfiguraciju i pokrenuti samu aplikaciju.



Slika. 4.3. *Struktura projekta na poslužiteljskoj strani*

4.2.1. Pokretanje aplikacije i inicijalne postavke

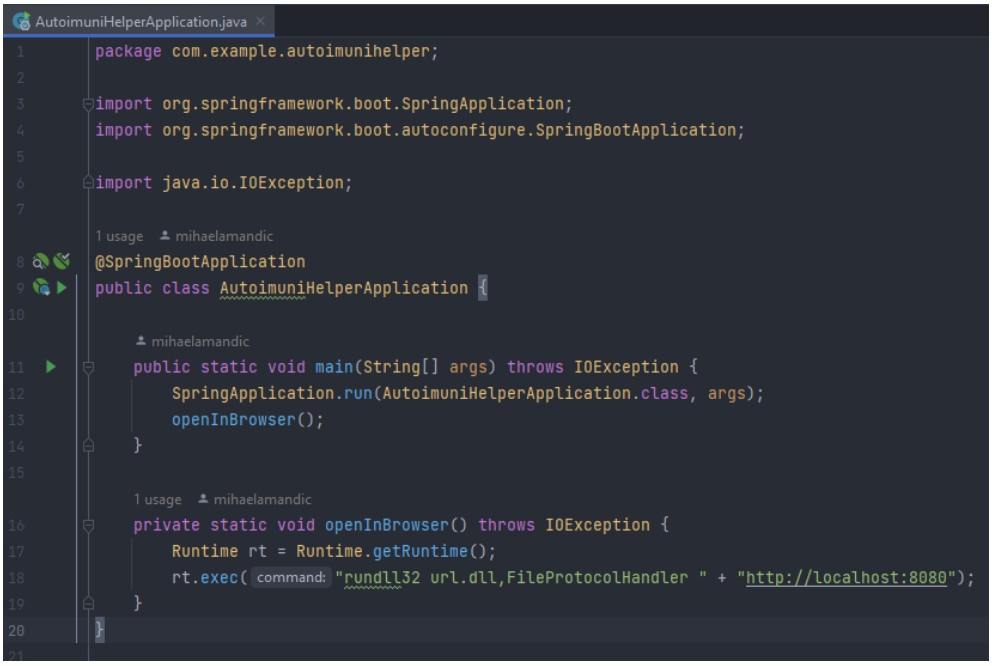
Budući da se razvija web aplikacija u Spring Boot-u, potrebno je definirati ovisnosti (engl. *dependencies*) koje su potrebne projektu prilikom prevođenja, gradnje, pokretanja i testiranja projekta, primjer određenih ovisnosti je vidljiv na slici 4.3.



Slika 4.3. *Primjer definiranih ovisnosti*

Za pokretanje aplikacije, što se može vidjeti na slici 4.4, potrebno je definirati klasu koja će koristiti Spring Boot za pokretanje web poslužitelja na lokalnom računalu i otvarati web stranice u zadanim web preglednicima. Ta klasa se u ovome web sustavu naziva AutoimuniHelperApplication. Pomoću anotacije @SpringBootApplication omogućava se označavanje navedene klase kao glavne klase i automatski se konfigurira Spring Boot aplikacija. U main funkciji se definira glavna metoda aplikacije koja će se izvršiti kad se aplikacija pokrene, a u njoj je definirano pokretanje glavne klase AutoimuniHelperApplication i otvaranje internetskih preglednika. Dakle, pomoću programskoga koda SpringApplication.run(AutoimuniHelperApplication.class, args) pokreće se učitavanje konfiguracije Spring Boot aplikacije i pokreće se ugrađeni web poslužitelj na zadanoj adresi, koja je u ovom slučaju http://localhost:8080. Funkcija openInBrowser() će otvoriti zadani URL

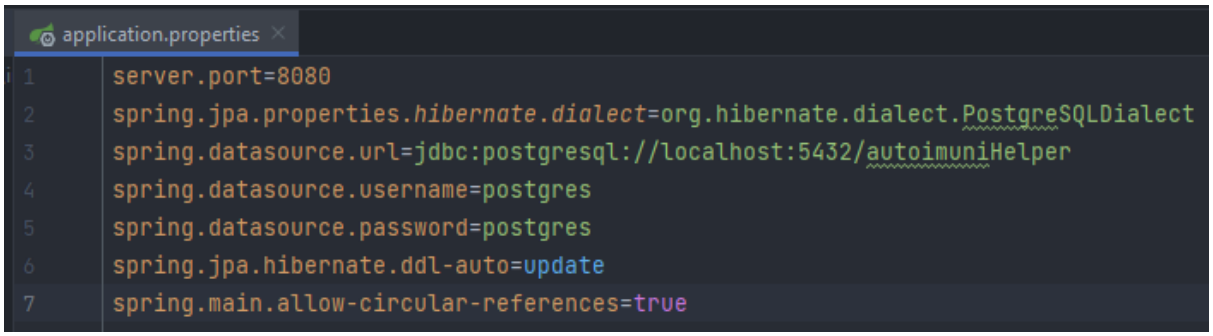
("http://localhost:8080") u zadanim web preglednicima koristeći instancu klase Runtime koja omogućuje izvršavanje vanjskih procesa iz Java aplikacije. Detaljnije govoreći, kada se na instanci Runtime klase pozove exec() funkcija, koja izvodi naredbe u operativnom sustavu, ona će izvesti predanu naredbu rundll32 url.dll,FileProtocolHandler + http://localhost:8080 gdje se korištenjem Windows naredbe rundll32 omogućava otvaranje predanog http://localhost:8080 URL-a u zadanim web preglednicima.



Slika 4.4. *Programsko rješenje glavne klase web sustava*

Nadalje, potrebno je definirati postavke u application.properties datoteci koja predstavlja konfiguracijsku datoteku te se koristi u Spring Boot aplikacijama kako bi se postavile različite konfiguracijske opcije i parametri. Ova datoteka omogućuje prilagodbu ponašanja aplikacije bez potrebe za mijenjanjem izvornog koda te se koristi kako bi se definirale različite postavke za aplikaciju, kao što su postavke baze podataka, postavke web poslužitelja, konfiguracija za vanjske usluge, postavke sigurnosti, i druge specifične postavke aplikacije. Na slici 4.5 se mogu vidjeti postavke definirane za ovaj web sustav. Pomoću server.port=8080 postavke definira se da će Spring Boot web poslužitelj slušati dolazne zahtjeve na 8080 portu. Od druge do šeste linije programskoga koda se definiraju postavke za bazu podataka gdje se u 2. i 6. liniji definiraju postavke za Hibernate, java okvir koji se koristi za spremanje java objekata u relacijske baze podataka, gdje se u 2. liniji definira dijalekt koji će se koristiti prilikom komunikacije s PostgreSQL bazom podataka, a u 6. liniji se definira da se automatski stvori tablica u bazi na temelju klasa entiteta u aplikaciji. Kako bi se sustav povezao te kako bi se

koristila PostgreSQL baza, u 3.,4. i 5. liniji koda je definiran URL do baze te korisničko ime i lozinka. U zadnjoj liniji koda se omogućuje Spring Bootu da podržava cirkularne reference između komponenata, što može biti korisno u određenim situacijama gdje se komponente referenciraju međusobno [29, 30].



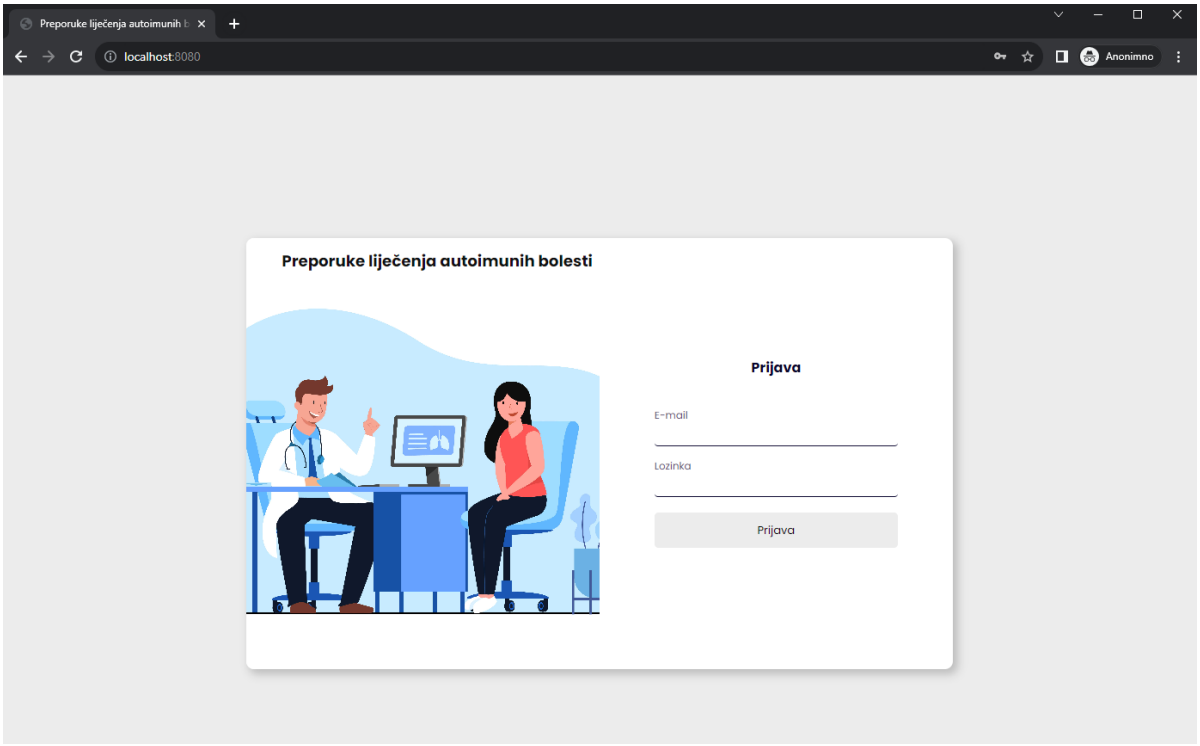
Slika 4.5 *Definirane postavke web sustava*

Također, za potrebe ovoga sustava gdje ukoliko ne postoje uloge, admin korisnik, bolesti, simptomi, lijekovi te prehrane u bazi podataka, kreirana je klasa SetupDataLoader koja će pokretanjem aplikacije, odnosno kada se dogodi osvježavanje konteksta aplikacije, pozvati funkciju onApplicationEvent kojom će ispuniti inicijalnu bazu s navedenim podacima. Primjer navedene funkcije te primjer pozvanih metoda za kreiranje admin korisnika i uloga je vidljiv na slici 4.6.

Slika 4.6. *Programski kod inicijalnog ispunjavanja baze podataka*

4.2.2. Prijava u web sustav i prikaz početne stranice

Pokretanjem web aplikacije, na internetskom pregledniku se otvara stranica za prijavu u sustav koja se može vidjeti na slici 4.7.



Slika 4.7. *Stranica za prijavu u sustav*

Dio programskog koda za izradu web prikaza se može vidjeti na slici 4.8 te je ondje vidljivo da se prilikom unosa i pritiska na unose za e-mail i lozinku pozivaju javascript funkcije validateEmailForm() i validatePasswordForm() koje ispisuju odgovarajuće poruke o pogreškama ukoliko je unesena neispravna e-mail adresa ili ako je prazna e-mail adresa ili lozinka, a primjer programskog koda se može vidjeti na slici 4.9.

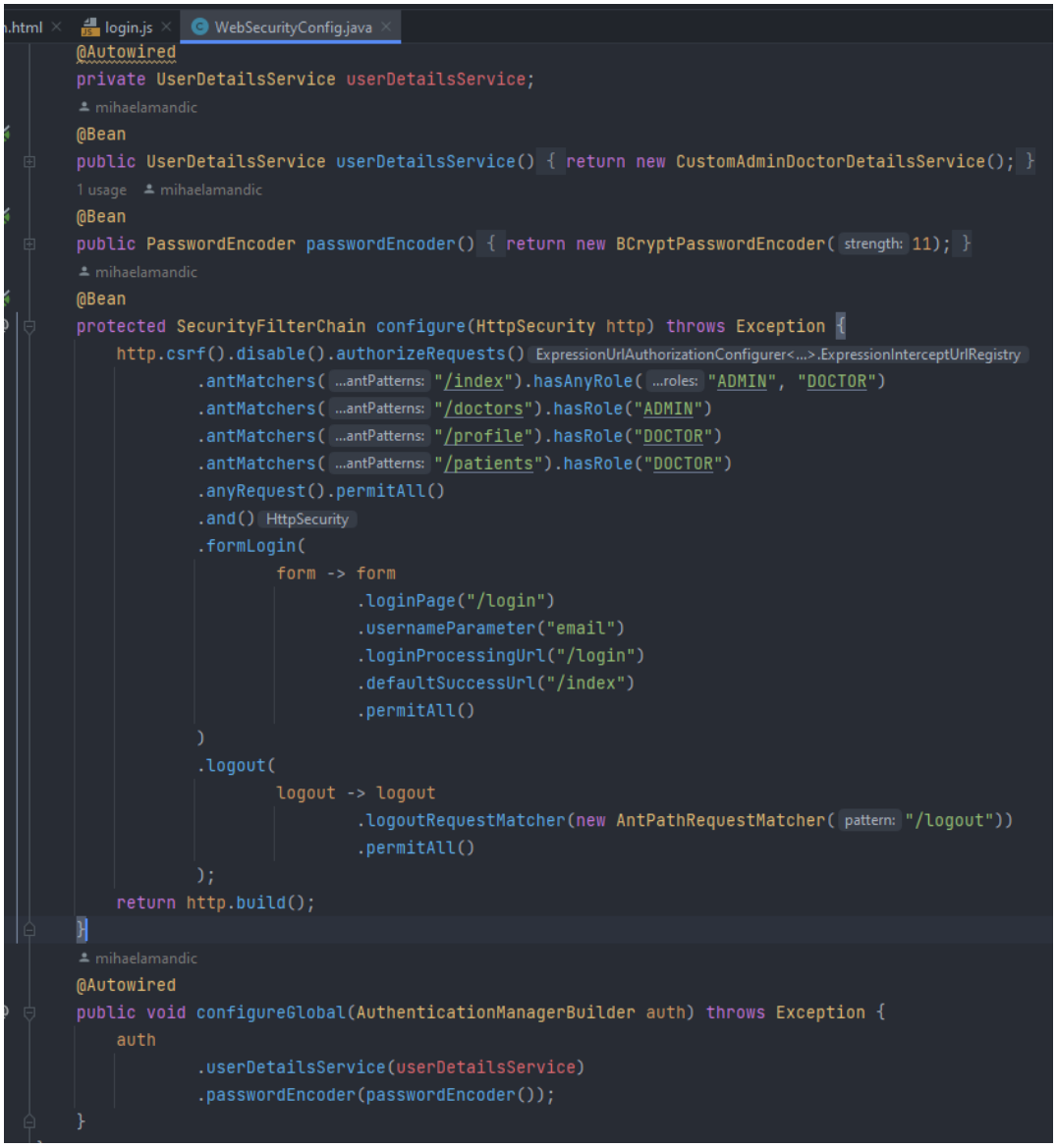
```
login.html
<div class="container main">
    <div class="row">
        <div class="col-md-6 side-image">
            <div class="text">
                <p id="title"><b>Preporuke liječenja autoimunih bolesti</b></p>
            </div>
        </div>
        <div class="col-md-6 right">
            <div class="input-box">
                <h6 id="loginHeader">Prijava</h6>
                <form class="" method="post" role="form" th:action="@{/login}">
                    <div class="input-field">
                        <input type="text" class="input form-control" id="email" name="email"
                               onclick="validateEmailForm()" onkeyup="validateEmailForm()" autocomplete="off">
                        <label for="email">E-mail</label>
                        <div id="error-Email" class="error"></div>
                    </div>
                    <div class="input-field">
                        <input type="password" class="input form-control" id="password" name="password"
                               onclick="validatePasswordForm()" onkeyup="validatePasswordForm()">
                        <label for="password">Lozinka</label>
                        <div id="error-Password" class="error"></div>
                    </div>
                    <div class="input-field">
                        <input type="submit" class="submit" id="loginSubmit" value="Prijava">
                    </div>
                    <div th:if="${param.error}">
                        <div class="alert alert-danger">Neispravan email ili lozinka.</div>
                    </div>
                    <div th:if="${param.logout}">
                        <div class="alert alert-success">Uspješno ste izlogirani.</div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<script th:src="@{/styles/js/login.js}"></script>
```

Slika 4.8. *Dio programskog koda za izradu web prikaza stranice za prijavu*

```
login.html  login.js
function validateEmailForm(){
    var name=document.getElementById( elementId: "email").value;
    if(name == null || name == ""){
        document.getElementById( elementId: "error-Email").innerHTML="E-mail polje mora biti upisano.";

    }
    else{
        if ((/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(name)))
            document.getElementById( elementId: "error-Email").innerHTML="";
        else{
            document.getElementById( elementId: "error-Email").innerHTML="Neispravna e-mail adresa.";
        }
    }
}
function validatePasswordForm(){
    var password = document.getElementById( elementId: "password").value;
    if(password == null || password == ""){
        document.getElementById( elementId: "error-Password").innerHTML="Lozinka mora biti upisana.";
    }
    else{
        document.getElementById( elementId: "error-Password").innerHTML="";
    }
}
```

Slika 4.9. *Javascript funkcije za validaciju unosa za e-mail i lozinku*

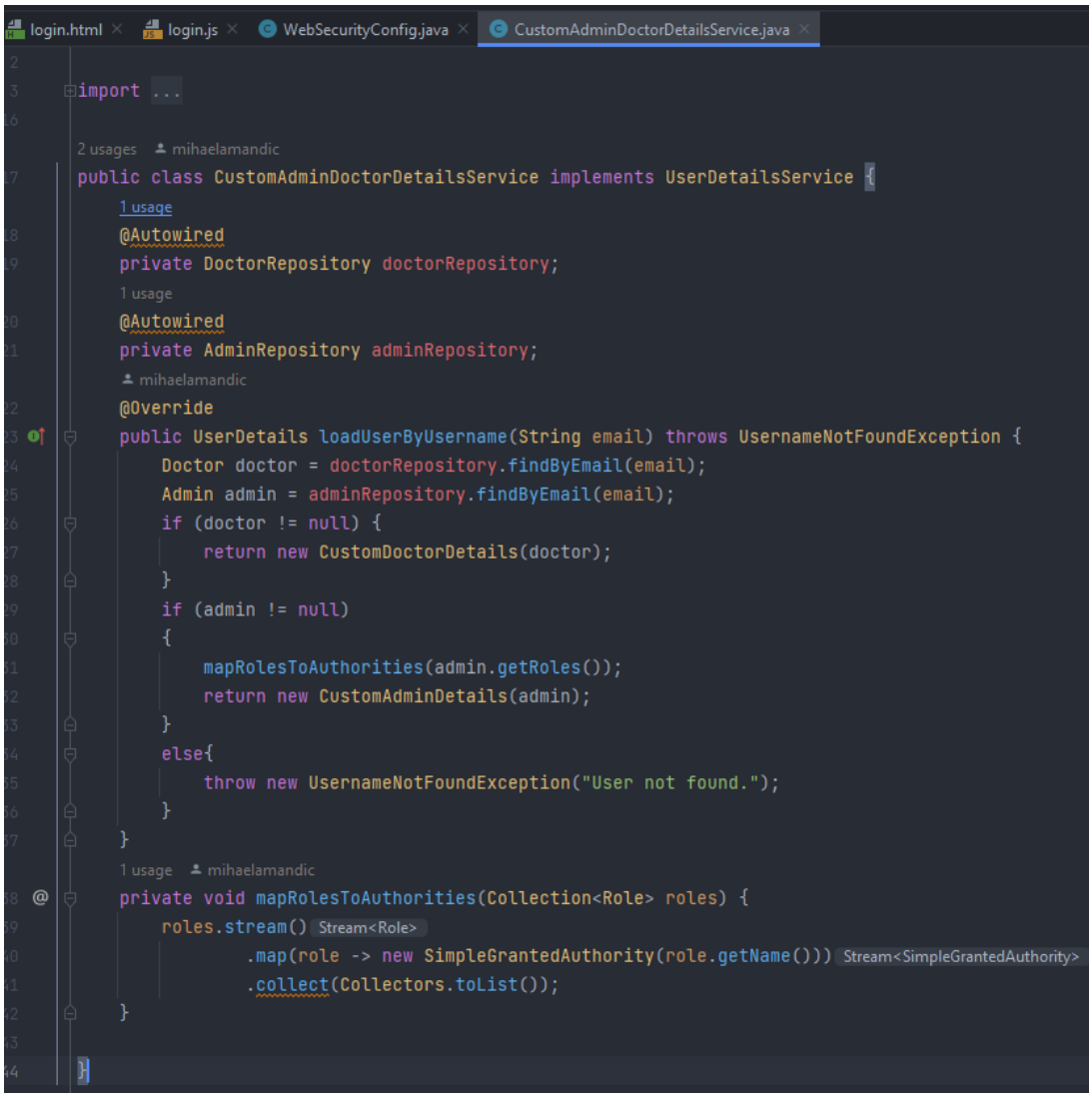
Ovim putem se u sustav mogu prijaviti samo admin korisnik, koji je kreiran samom aplikacijom, te svi doktor korisnici koje admin kreira. Ispunjavajući e-mail i lozinku te pritiskom na Prijava gumb, dolazi do validacije unesenih parametara gdje, ukoliko je unesen ispravan e-mail i password, odnosno ukoliko korisnik ima ovlaštenje za korištenje aplikacije, tada se korisnik uspješno prijavljuje u sustav, a u suprotnom se prikazuje odgovarajuća poruka. Provjera ovlaštenja se provodi unutar WebSecurityConfig klase koja predstavlja konfiguracijsku klasu za Spring Security te omogućava kontrolu pristupa na temelju korisničkih uloga, definira obrazac za prijavu i postavlja konfiguraciju za odjavu, a programski kod klase se može vidjeti na slici 4.10. Spring Security se koristi za upravljanje sigurnošću i autentifikacijom u web aplikacijama. U programskome kodu je vidljivo korištenje @Configuration anotacije koja označava klasu kao konfiguracijsku klasu, što znači da će Spring framework obrađivati ovu klasu kako bi konfigurirao različite aspekte aplikacije, i @EnableWebSecurity anotacija koja označava da se koristi Spring Security kako bi se omogućila sigurnost web aplikacije. Unutar configure() funkcije su definirana pravila autentifikacije i autorizacije za različite URL-ove gdje antMatchers() prima URL od stranice kojoj se pristupa, a hasAnyRole() predstavlja sve uloge koje mogu pristupiti tom URL-u gdje se mogu definirati više uloga, te hasRole() definira jednu ulogu koja može pristupiti URL-u. Metoda formLogin() definira konfiguraciju za obrazac za prijavu. Postavke uključuju prilagodbu URL-ova za prijavu, parametara korisničkog imena i obrade prijave. Metoda logout() definira konfiguraciju za odjavu u kojoj se konfigurira URL i dozvola za odjavu. Također, unutar metode configureGlobal() se konfigurira globalnog AuthenticationManagera koji se koristi za autentifikaciju korisnika te se konfigurira userDetailsService koji služi za dohvaćanje podataka o korisniku i passwordEncoder koji se koristi prilikom enkripcije lozinki.



Slika 4.10. *Programski kod WebSecurityConfig klase*

Detaljnije kako izgleda kod koji koristi UserDetailsService sučelje kako bi vratio informacije o korisniku se može vidjeti na slici 4.11. U ovom programskom kodu definirana je klasa nazvana CustomAdminDoctorDetailsService koja implementira sučelje UserDetailsService što zahtjeva implementaciju loadUserByUsername() funkcije. Ova klasa se koristi za dohvaćanje detalja korisnika (bilo doktora ili administratora) na temelju njihove e-mail adrese. Detalji korisnika su zatim korišteni za stvaranje prilagođenih implementacija UserDetails sučelja koje Spring Security koristi za autentifikaciju i autorizaciju. Dakle, u loadUserByUsername() funkciji provodi se pretraga korisnika putem njihove e-mail adrese. Pretraga se obavlja u dvije tablice: doctorRepository (tablica doktora) i adminRepository (tablica administratora). Ako je korisnik pronađen u tablici doktora, stvara se nova instanca

klase CustomDoctorDetails, koja implementira sučelje UserDetails. Ova instanca sadrži detalje koji su bitni za procese autentifikacije i autorizacije. U slučaju da se korisnik pronađe u tablici administratora, poziva se metoda mapRolesToAuthorities. Ova metoda mapira uloge administratora u ovlasti, koje su ključne za utvrđivanje prava pristupa. Nakon toga, stvara se nova instanca klase CustomAdminDetails. Ako korisnik nije pronađen ni u jednoj od dvije tablice, iznimka UsernameNotFoundException signalizira nepostojećeg korisnika. Privatna metoda mapRolesToAuthorities koristi Java Stream API za mapiranje uloga administratora u ovlasti. Svaka uloga se pretvara u objekt tipa SimpleGrantedAuthority, a rezultati se prikupljaju u listi.



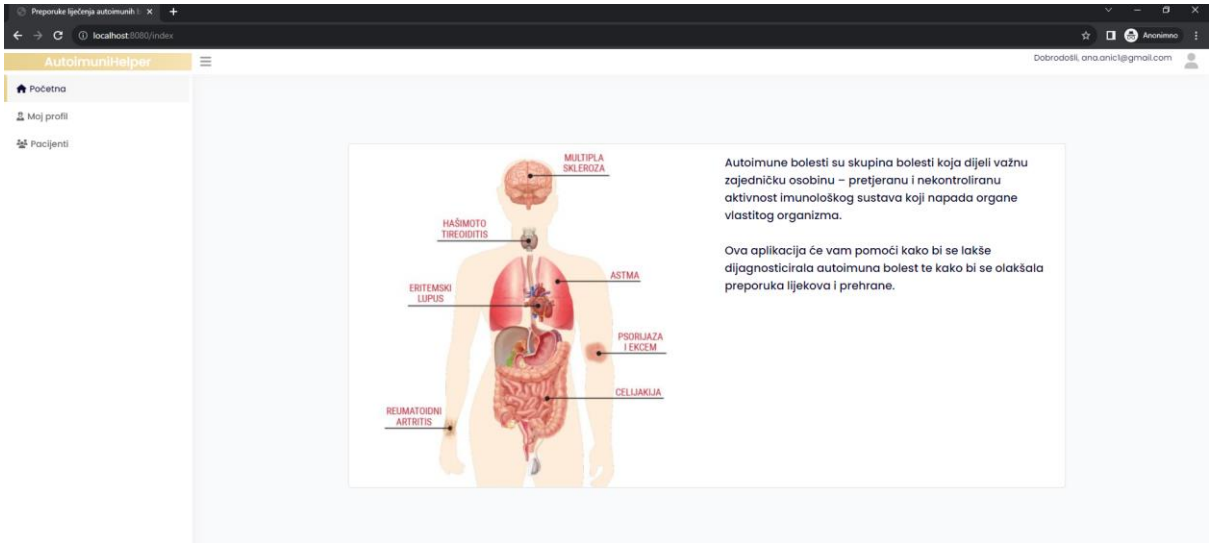
Slika 4.11. *Programski kod za dohvat informacija o korisniku*

Primjer kako izgleda CustomDoctorDetails klasa koja omogućuje dohvat informacija o liječniku se može vidjeti na slici 4.12. Detalji liječnika uključuju informacije kao što su ime, prezime, e-mail adresa, organizacija, telefonski broj i druge relevantne informacije te, također,

klasa sadržava metodu getAuthorities() koja dohvaća uloge povezane s korisnikom te ih mapira u ovlasti koje se kasnije koriste za utvrđivanje prava pristupa.

Slika 4.12. *Programski kod za dohvat informacija o liječniku*

Kada se korisnik uspješno ulogira u sustav, ovisno o ulozi (admin/liječnik) vidi naslovnu stranicu s različitim mogućnostima u navigaciji što je definirano u HTML-u. Dakle, u WebSecurityConfig klasi se omogućavao pristup određenim stranicama s obzirom na uloge, dok u HTML-u su skriveni elementi s obzirom na uloge. Prikaz izgleda naslovne stranice od strane doktora je vidljiv na slici 4.13 te se na slici 4.14 može vidjeti HTML kod gdje se manipulira vidljivošću elemenata s obzirom na uloge.



Slika 4.13. *Naslovna stranica web sustava*

index.html

```
<!-- SIDEBAR START-->
<div class="left-menu">
    <div class="menubar-content">
        <nav class="animated bounceInDown">
            <ul id="sidebar">
                <li class="active" sec:authorize="isAuthenticated()">
                    <a th:href="@{/index}"><i class="fa-solid fa-house"></i>Početna</a>
                </li>
                <li sec:authorize="hasRole('DOCTOR')">
                    <a th:href="@{/profile}"><i class="fa-solid fa-user-nurse"></i>Moj profil</a>
                </li>
                <li sec:authorize="hasRole('DOCTOR')">
                    <a th:href="@{/patients}"><i class="fa-solid fa-users"></i>Pacijenti</a>
                </li>
                <li sec:authorize="hasRole('ADMIN')">
                    <a th:href="@{/doctors}"><i class="fa-solid fa-users"></i>Doktori</a>
                </li>
                <li sec:authorize="hasRole('ADMIN')">
                    <a th:href="@{/patientsArhive}"><i class="fa-solid fa-users"></i>Pacijenti</a>
                </li>
            </ul>
        </nav>
    </div>
</div>
```

Slika. 4.14. *HTML kod kojim se upravlja prikazom elemenata s obzirom na ulogu*

4.2.3. Kreiranje pacijenata

Kreiranje pacijenta započinje verifikacijom OIB-a gdje ukoliko pacijent već postoji u sustavu, tada se omogućuje doktoru da izabere želi li si pripisati pacijenta, ukoliko pacijent ne postoji u sustavu, tada se otvara mogućnost kreiranja pacijenta i ispunjavanja informacija o pacijentu te ukoliko je pacijent arhiviran od strane ADMIN-a, tada je nemoguće dodijeliti pacijenta. Također, ovdje su prisutne i validacije OIB-a gdje ukoliko je OIB prazan ili ne zadovoljava broj znamenki, tada se ispisuje odgovarajuća poruka o pogreški. Na slici 4.15 se može vidjeti

programski kod s korisničke strane gdje se pritiskom na gumb Verificiraj pacijenta šalje HTTP POST zahtjev do kontrolera koji sadržava putanju /verifyPatientAlreadyExist.

```
<form id="formCreateNewPatient" method="POST" role="form"
      th:action="@{/verifyPatientAlreadyExist}" th:object="${patient}">
    <div class="container">
        <div class="row">
            <div class="mb-5 ms-5 col-md-3">
                <label for="oib" class="form-label">OIB</label>
                <input class="form-control" type="text" name="oib"
                       placeholder="11111111111" id="oib" th:field="*{oib}"/>
                <p th:if="${resultOfOib == -1}" class="error">OIB ne smije biti
                    prazan.</p>
                <p th:if="${resultOfOib == -2}" class="error">OIB mora sadržavati 11
                    znamenki.</p>
            </div>
        </div>
        <div class="mt-2">
            <button type="submit" class="btn btn-primary me-2">Verificiraj pacijenta
            </button>
            <a th:href="@{/patients}" class="btn btn-outline-secondary">Natrag</a>
        </div>
    </div>
</form>
```

Slika 4.15. *Programski kod s korisničke strane gdje se odvija validacija OIB-a*

U kontroleru se u model varijablu resultOfOib spremaju rezultati provjere OIB-a te, s obzirom na validaciju i verifikaciju OIB-a, vraća se određeni prikaz unutar kojega je moguće, s obzirom na model varijablu, prikazati određenu poruku što je prikazano slikom 4.16. Dakle, ukoliko je OIB neispravnog oblika, tada se ispisuju poruke koje su vidljive na slici 4.15, a u suprotnom se vraća drugi prikaz i ispisuju poruke koje su vidljive na slici 4.17.

Slika 4.16. *Programski kod za provjeru postojećeg korisnika*

verifyOibResponse.html

```
<form id="formVerifyOibResponse" method="POST" role="form"
      th:action="@{/reassignPatient/{id}(id=${patient.id})}" th:object="${patient}">
    <div class="container">
        <div class="row">
            <div class="mb-5 ms-5 col-md-3">
                <label for="oib" class="form-label">OIB</label>
                <input class="form-control" type="text" name="oib" id="oib"
                       placeholder="11111111111" th:field="*{oib}" disabled/>
            </div>
        </div>
        <div th:if="${resultOfOib == 0}">
            <div class="alert alert-danger">Pacijent s tim oibom ne postoji</div>
            <div class="mt-2">
                <a th:href="@{/addNewPatient/{oib}(oib=${oib})}"
                   class="btn btn-primary me-2">Kreiraj novog pacijenta</a>
                <a th:href="@{/patients}" class="btn btn-outline-secondary">Natrag</a>
            </div>
        </div>
        <div th:if="${resultOfOib == 1}">
            <div class="alert alert-success"> Pacijent već postoji u sustavu. Želite li
                si dodijeliti pacijenta?
            </div>
            <div class="mt-2">
                <button type="submit" class="btn btn-primary me-2">Dodijeli postojećeg
                    pacijenta
                </button>
                <a th:href="@{/patients}" class="btn btn-outline-secondary">Natrag</a>
            </div>
        </div>
        <div th:if="${resultOfOib == 2}">
            <div class="alert alert-danger">Pacijent s tim oibom je arhiviran.</div>
            <div class="mt-2">
                <a th:href="@{/patients}" class="btn btn-outline-secondary">Natrag</a>
            </div>
        </div>
    </div>
</form>
```

Slika 4.17. P*rogramski kod s korisničke strane gdje se obavještava liječnika o stanju pacijenta u sustavu*

Upisavši ispravan OIB s kojim nije kreiran niti jedan pacijent, doktor dolazi do prikaza za unošenje podataka o pacijentu. Slika 4.18 prikazuje API kreiranja pacijenta koji se nalazi u kontroleru.

```
@PostMapping("/createPatient")
public String createPatient(@AuthenticationPrincipal CustomDoctorDetails doctorDetails,
                            @Valid @ModelAttribute("patient") Patient patient, BindingResult result, Model model){
    if (result.hasErrors()) {
        model.addAttribute( attributeName: "foodAllergiesFromDb", dietService.getAllDiets());
        return "addNewPatient";
    }
    patientService.createPatient(doctorDetails.getId(), patient);
    return "redirect:/patients";
}
```

Slika 4.18. *Prikaz API-ja za kreiranje pacijenta*

API može vratiti jedan od dva tipa prikaza, ovisno o tome sadrži li rezultat pacijent objekta pogreške, gdje ukoliko sadrži pogreške, tada vraća prikaz za ispunjavanje profila pacijenta s

odgovarajućim validacijskim porukama, a u suprotnom vraća prikaz s tablicom liječnikovih pacijenata gdje se, prije prikaza svih pacijenata, poziva metoda createPatient koja radi logiku kreiranja pacijenta unutar usluge patientService, a prima ID od liječnika koji je izvršio kreiranje i objekt pacijenta te je implementacija metode prikazana slikom 4.19.

```
@Override
public void createPatient(Long docId, Patient patient) {
    Patient newPatient = new Patient();
    String pregnancyStatus = getCheckboxStatus(request, checkboxName: "pregnancyStatus");
    String breastFeedingStatus = getCheckboxStatus(request, checkboxName: "breastFeedingStatus");
    newPatient.setName(patient.getName());
    newPatient.setSurname(patient.getSurname());
    newPatient.setAddress(patient.getAddress());
    newPatient.setDate(patient.getDate());
    newPatient.setGender(patient.getGender());
    newPatient.setEmail(patient.getEmail());
    newPatient.setOib(patient.getOib());
    newPatient.setTelephoneNumber(patient.getTelephoneNumber());
    newPatient.setPregnancyStatus(pregnancyStatus);
    newPatient.setBreastFeedingStatus(breastFeedingStatus);
    newPatient.setStatus("Aktivan");
    newPatient.setFoodAllergies(patient.getFoodAllergies());
    newPatient.setSystemAssists(true);
    Doctor doctor = doctorService.getDoctorById(docId);
    newPatient.setDoctor(doctor);
    patientRepository.save(newPatient);
}
```

Slika 4.19. *Programski kod metode kreiranja pacijenta*

Metoda createPatient ima zadatak kreirati novog pacijenta u sustavu koristeći informacije o pacijentu i identifikatoru liječnika. Ovaj proces slijedi nekoliko koraka. Na početku metode, stvara se nova instanca klase Patient, koja će sadržavati informacije o novom pacijentu. Metodom getCheckboxStatus izvlače se informacije o statusima trudnoće i dojenja pacijenta, koje će biti spremljene kako bi se kasnije postavile u novog pacijenta. Podaci poput imena, prezimena, adrese, datuma rođenja, spola, e-maila, OIB-a (Osobni Identifikacijski Broj) i broja telefona kopiraju se iz predanih informacija o pacijentu u novog pacijenta. Dobivene vrijednosti statusa trudnoće i dojenja postavljaju se u svoje odgovarajuće polje u novom pacijentu. Pacijentu se, također, dodjeljuju status aktivan, prehrambene alergije iz informacija o pacijentu te informacija da pacijent koristi pomoć sustava prilikom dijagnosticiranja bolesti i preporuka lijekova i prehrane. Korištenjem identifikatora doktora, poziva se usluga doctorService.getDoctorById kako bi se dobio odgovarajući doktor za novog pacijenta. Novi pacijent se dodjeljuje doktoru s vezom 1..N gdje jedan doktor može imati više pacijenata, odnosno više pacijenata može imati jednog doktora. Naposljetku, novi pacijent se sprema u

bazu podataka putem poziva save funkcije koristeći patientRepository sučelje. CreatePatient metoda osigurava da se pravilno prikupe i pohrane informacije o novom pacijentu te da se uspješno poveže s odgovarajućim doktorom u sustavu. Na slici 4.20 može se vidjeti model pacijenta koji za određene atribute ima dodanu validaciju koja se prilikom kreiranja/izmijene profila pacijenta provjerava i prikazuje ukoliko dođe do pogreške prilikom ispunjavanja određenih polja.

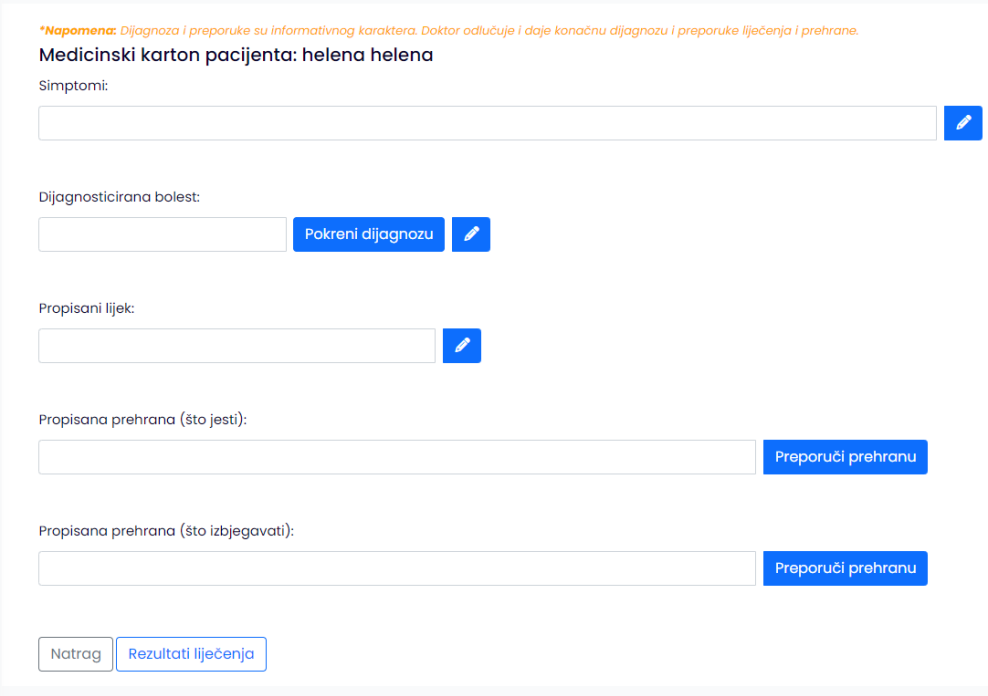
```java
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "patient")
public class Patient {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    2 usages
    @Column(nullable = false)
    @NotBlank(message = "Ime ne smije biti prazno.")
    private String name;

    2 usages
    @Column(nullable = false)
    @NotBlank(message = "Prezime ne smije biti prazno.")
    private String surname;

    2 usages
    @Column(nullable = false)
    @NotBlank(message = "E-mail ne smije biti prazan.")
    @Email(message = "Neispravna e-mail adresa.")
    private String email;

    2 usages
    @NotBlank(message = "OIB ne smije biti prazan.")
    @Column(nullable = false, unique = true)
    private String oib;

    2 usages
    @Column(nullable = false)
    @NotBlank(message = "Adresa ne smije biti prazna.")
    private String address;

    2 usages
    @Column(nullable = false)
    @NotBlank(message = "Broj mobitela ne smije biti prazan.")
    @Size(min = 7, max = 9 , message = "Broj mobitela mora imati od 7 do 9 znamenki.")
    @Pattern(regexp = "[0-9]+", message = "Broj mobitela mora sadržavati samo znamenke.")
```

Slika 4.20. *Prikaz modela pacijenta s validacijama*

4.2.4. Dijagnoza bolesti i propisivanje lijekova i prehrane

Nakon uspješnog kreiranja pacijenta od strane liječnika, liječnik ima mogućnost navigacije u medicinski karton pacijenta gdje se omogućuje dodavanje simptoma koje pacijent ima, dijagnosticiranje bolesti te propisivanje lijekova i prehrane. Medicinski karton pacijenta se može vidjeti na slici 4.21, a HTML dio programskog koda koji prikazuje stranicu sa slike 4.21,

prikazan je slikom 4.22. Na slici 4.22 nije bilo moguće prikazati cijeli kod stranice zbog veličine programskoga koda.



Slika 4.21. *Prikaz medicinskoga kartona pacijenta*

Slika 4.22. *Prikaz dijela programskog koda medicinskoga kartona pacijenta*

Kako bi se prikazali ažurirani podaci pacijenta unutar medicinskoga kartona, prilikom svakog prikaza medicinskoga kartona izvršava se HTTP GET zahtjev gdje se dohvaćaju iz baze sve informacije o pacijentu i spremaju u patient model. Nadalje, dohvaćaju se svi simptomi koji postoje u bazi kako bi se omogućio odabir simptoma koje pacijent ima. Osim toga, dohvaća se pacijentova dijagnosticirana bolest, dohvaćaju se lijekovi za dijagnosticiranu bolest te lijekovi s obzirom na bolest i pacijenta te prehrana. Navedeno dohvaćanje podataka te vraćanje prikaza medicinskog kartona sa svim relevantnim informacijama dodanim modelu je vidljivo na slici 4.23.

```
@GetMapping("/medicalRecord/{id}")
public String medicalRecord(@PathVariable Long id, Model model){
    Patient patient = patientService.getPatientById(id);
    model.addAttribute( attributeName: "patient", patient);
    model.addAttribute( attributeName: "symptomsFromDb", symptomService.getAllSymptoms());
    String disease = patient.getDiagnosedDisease();
    List<Medicine> medicinesForDisease = medicineService.getAllMedicinesForDisease(disease);
    model.addAttribute( attributeName: "medicinesByDisease", medicinesForDisease);
    model.addAttribute( attributeName: "recommendedMedicinesMap", medicineService.recommendMedicines(medicinesForDisease
    List<Diet> dietsOk = dietService.getAllDietsOkNokForDisease(disease, isOk: true,patient);
    List<Diet> dietsNok = dietService.getAllDietsOkNokForDisease(disease, isOk: false,patient);
    model.addAttribute( attributeName: "dietsOk", dietsOk);
    model.addAttribute( attributeName: "dietsNok", dietsNok);
    return "medicalRecord";
}
```

Slika 4.23. *Programski kod dohvaćanja podataka unutar medicinskoga kartona pacijenta*

Odabravši pacijentove simptome, liječnik klikom na gumb Spremi okida API za spremanje označenih simptoma što je prikazano slikom 4.24.

```
@PostMapping("/addSymptomsToPatientById/{id}")
public String addSymptomsToPatientById(@PathVariable Long id, @ModelAttribute("patient") Patient symptomPatient){
    Patient patient = patientService.getPatientById(id);
    patient.setSymptoms(symptomPatient.getSymptoms());
    patientRepository.save(patient);
    return "redirect:/medicalRecord" + "/" + id;
}
```

Slika 4.24. *Programski kod spremanja simptoma pacijenta*

Metoda addSymptomsToPatientById reagira na HTTP POST zahtjev i ima svrhu dodavanja simptoma određenom pacijentu. Kada se pozove ova metoda, najprije se dohvaća pacijent iz baze podataka putem patientService.getPatientById(id) te se sprema u varijablu patient. Zatim, putem obrasca koji je predan kroz @ModelAttribute, dobivaju se simptomi pacijenta koji se žele dodati. Ovi simptomi se nalaze unutar objekta tipa Patient nazvanog symptomPatient. Dalje, kopiraju se simptomi iz symptomPatient u originalni objekt pacijenta, patient, koristeći metodu setSymptoms. Time se postojeći simptomi zamjenjuju novim simptomima. Nakon toga, promjene na pacijentu se spremaju u bazu podataka pozivom metode patientRepository.save(patient). Konačno, kako bi se liječniku omogućilo da vidi ažurirane informacije, liječnik se preusmjerava na stranicu medicinskog kartona za istog pacijenta. To se postiže korištenjem return "redirect:/medicalRecord" + "/" + id.

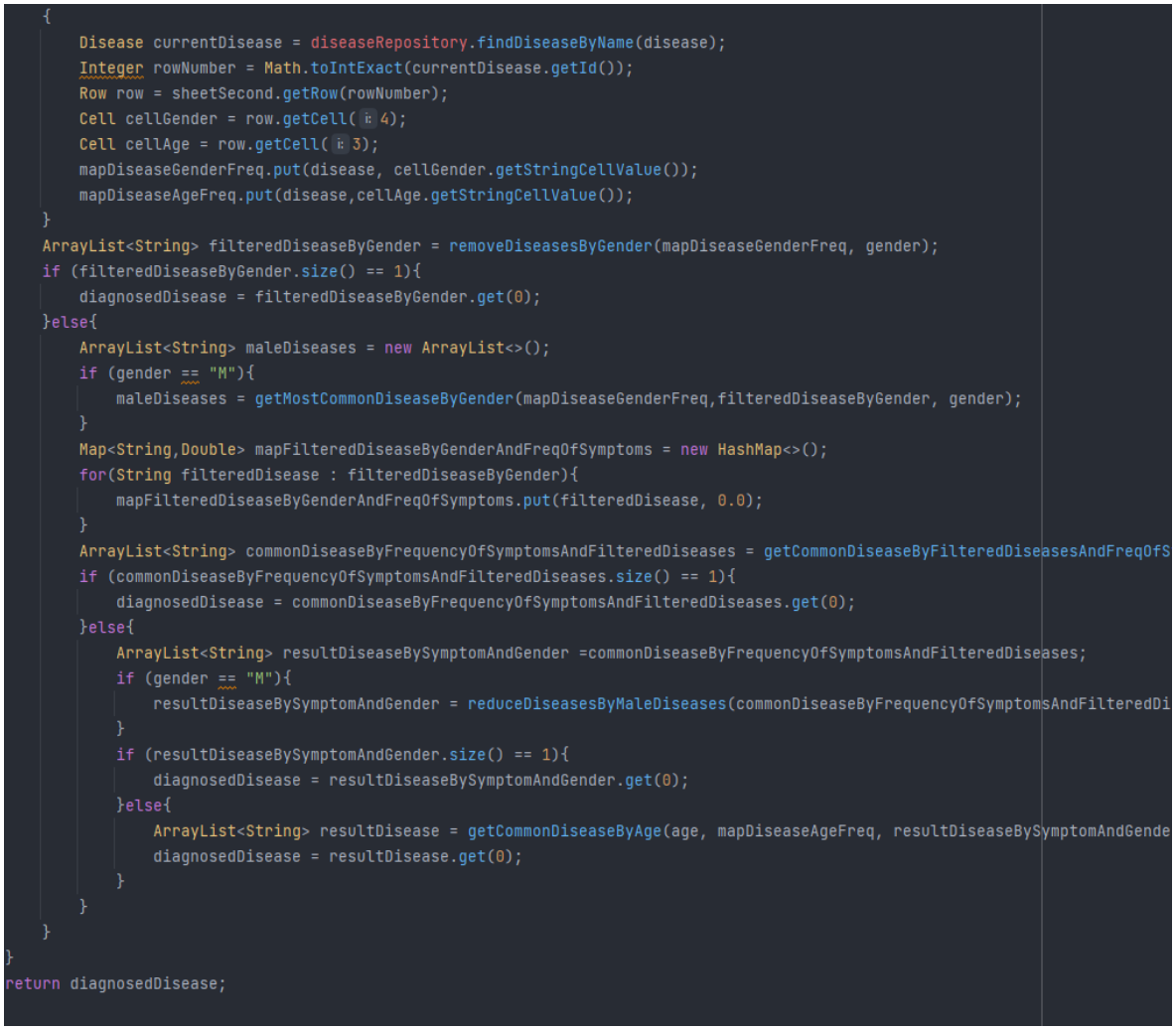
Nakon uspješnog spremanja simptoma, idući korak je dijagnoza bolesti. Klikom doktora na gumb Pokreni dijagnozu, okida se API za dijagnosticiranje bolesti koji je prikazan slikom 4.25.

Slika 4.25. *Programski kod API-a za dijagnosticiranje bolesti*

Metoda diagnoseDiseaseForPatient obrađuje HTTP POST zahtjev i ima svrhu postavljanja dijagnoze bolesti za pacijenta na temelju njegovih simptoma. Pozivom ove metode deklarira se varijabla symptomsMissing koja služi za slučaj kada pacijent nema unesene simptome te tada vraća stranicu medicinski karton s odgovarajućom porukom o pogrešci. Ako pacijent ima simptome, koristi se usluga symptomService.calculateAutoimmuneDisease kako bi se izračunala dijagnoza bolesti na temelju simptoma. Dijagnoza se postavlja u pacijent objekt putem setDiagnosedDisease metode. Također, postavlja se systemAssists na true kako bi se označilo da sustav pomaže liječniku prilikom daljnjih preporuka lijeka i prehrane. Naposljetku, promjene na pacijentu se spremaju u bazu podataka pozivom patientRepository.save(patient). Na kraju metode, korisnika se preusmjerava na stranicu medicinskog kartona za istog pacijenta kako bi mogao vidjeti ažurirane informacije. Logika računanja dijagnosticirane bolesti se odvija unutar usluge gdje se, čitajući excel s pravilima koja upućuju na pojedinu bolest, najprije u posebnu listu spremaju sve bolesti za svaki simptom te se računa frekvencija pojavljivanje svake bolesti unutar te liste. Ukoliko postoji jedinstveno rješenje autoimune bolesti koja se najviše pojavljuje, tada se ta bolest vraća kao rezultat. U suprotnom, ako postoje dvije ili više bolesti koje se pojavljuju s najvećom frekvencijom, tada se računa zbroj težina simptoma koji su odabrani kod pacijenta, a koji upućuju na bolesti iz liste. Ukoliko postoji jedinstveno

rješenje, tada se ono vraća kao rezultat metode, a u suprotnom se bolesti u listi filtriraju po spolu gdje ukoliko je pacijent muškog spola, a određene bolesti se pojavljuju uglavnom kod žena, tada se bolesti koje se pojavljuju kod žena miču iz liste. Zatim, ukoliko i dalje nema jedinstvenoga rješenja, filtriraju se bolesti unutar liste po godinama pacijenta. U konačnici, ako niti nakon filtriranja po godinama ne postoji jedno jedinstveno rješenje, tada se vraća prva bolest u zadnje filtriranoj listi. Primjer programskoga rješenja računanja dijagnosticirane bolesti je vidljiv na slici 4.26.

```
public String calculateAutoimmuneDisease(Patient patient) {
    String diagnosedDisease = null;
    String path = System.getProperty("user.dir") + "\\src\\main\\java\\com\\example\\autoimunihelper\\BazaDijag
    File file = new File(path);
    FileInputStream excelFile = null;
    try {
        excelFile = new FileInputStream(file);
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
    String symptoms = patient.getSymptoms();
    String gender = patient.getGender();
    Integer age = calculateAge(patient.getDate());
    String[] symptomsArray = symptoms.split( regex: ",", limit: 0);
    ArrayList<String> diseaseArray = new ArrayList<>();
    Map<String,String> mapDiseaseSymptomFreq = new HashMap<>();
    XSSFWorkbook workbook = null;
    try {
        workbook = new XSSFWorkbook(excelFile);
        XSSFSheet sheet = workbook.getSheetAt( index: 0);
        for(String symptom: symptomsArray) {
            Symptom currentSymptom = symptomRepository.findSymptomByName(symptom);
            Integer rowNumber = Math.toIntExact(currentSymptom.getId());
            Row row = sheet.getRow(rowNumber);
            Cell cellDisease = row.getCell( i: 2);
            Cell cellSymptomFreq= row.getCell( i: 3);
            diseaseArray.add(cellDisease.getStringCellValue());
            mapDiseaseSymptomFreq.put(cellDisease.getStringCellValue(), cellSymptomFreq.getStringCellValue());
        }
    } catch (IOException e) {
        e.getStackTrace();
    }
    ArrayList<String> diseasesWithoutComma = getArrayListWithSplittedValuesByComma(diseaseArray);
    ArrayList<String> commonDisease = getMostCommonDiseaseBySymptoms(diseasesWithoutComma);
    if (commonDisease.size() == 1){
        diagnosedDisease = commonDisease.get(0);
    }else{
        Map<String,String> mapDiseaseGenderFreq = new HashMap<>();
        Map<String,String> mapDiseaseAgeFreq = new HashMap<>();
        XSSFSheet sheetSecond = workbook.getSheetAt( index: 1);
        for(String disease: commonDisease)
```



```
    {
        Disease currentDisease = diseaseRepository.findDiseaseByName(disease);
        Integer rowNumber = Math.toIntExact(currentDisease.getId());
        Row row = sheetSecond.getRow(rowNumber);
        Cell cellGender = row.getCell( i: 4);
        Cell cellAge = row.getCell( i: 3);
        mapDiseaseGenderFreq.put(disease, cellGender.getStringCellValue());
        mapDiseaseAgeFreq.put(disease,cellAge.getStringCellValue());
    }
    ArrayList<String> filteredDiseaseByGender = removeDiseasesByGender(mapDiseaseGenderFreq, gender);
    if (filteredDiseaseByGender.size() == 1){
        diagnosedDisease = filteredDiseaseByGender.get(0);
    }else{
        ArrayList<String> maleDiseases = new ArrayList<>();
        if (gender == "M"){
            maleDiseases = getMostCommonDiseaseByGender(mapDiseaseGenderFreq,filteredDiseaseByGender, gender);
        }
        Map<String,Double> mapFilteredDiseaseByGenderAndFreqOfSymptoms = new HashMap<>();
        for(String filteredDisease : filteredDiseaseByGender){
            mapFilteredDiseaseByGenderAndFreqOfSymptoms.put(filteredDisease, 0.0);
        }
        ArrayList<String> commonDiseaseByFrequencyOfSymptomsAndFilteredDiseases = getCommonDiseaseByFilteredDiseasesAndFreqOfS
        if (commonDiseaseByFrequencyOfSymptomsAndFilteredDiseases.size() == 1){
            diagnosedDisease = commonDiseaseByFrequencyOfSymptomsAndFilteredDiseases.get(0);
        }else{
            ArrayList<String> resultDiseaseBySymptomAndGender =commonDiseaseByFrequencyOfSymptomsAndFilteredDiseases;
            if (gender == "M"){
                resultDiseaseBySymptomAndGender = reduceDiseasesByMaleDiseases(commonDiseaseByFrequencyOfSymptomsAndFilteredDi
            }
            if (resultDiseaseBySymptomAndGender.size() == 1){
                diagnosedDisease = resultDiseaseBySymptomAndGender.get(0);
            }else{
                ArrayList<String> resultDisease = getCommonDiseaseByAge(age, mapDiseaseAgeFreq, resultDiseaseBySymptomAndGende
                diagnosedDisease = resultDisease.get(0);
            }
        }
    }
}
return diagnosedDisease;
```

Slika 4.26. *Isječak programskoga koda za računanje dijagnoze bolesti*

Nakon uspješne dijagnoze bolesti, liječnik pokreće preporuku lijeka klikom na gumb Preporuči lijek. Dohvaćeni podaci o lijekovima s obzirom na bolest i podaci o lijekovima s obzirom na bolest i određene karakteristike pacijenta su već opisani gore slikom 4.23. Logika preporuke lijekova s obzirom na bolest i karakteristike pacijenta se odvija u usluzi, a metoda je prikazana slikom 4.27.

```java
@Override
public Map<String, String> recommendMedicines(List<Medicine> medicinesForDisease, Patient patient) {
    Map<String, String> medicinesAndDescription = new HashMap<>();
    String patientPregnancyStatus = patient.getPregnancyStatus();
    String patientBreastFeedingStatus = patient.getBreastFeedingStatus();
    Integer patientAge = symptomService.calculateAge(patient.getDate());
    if (medicinesForDisease.size() != 0){
        String path = System.getProperty("user.dir") + "\\src\\main\\java\\com\\example\\autoimunihelper\\BazaDijagnozaBolesti.xlsx";
        File file = new File(path);
        FileInputStream excelFile = null;
        try {
            excelFile = new FileInputStream(file);
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
        try {
            XSSFWorkbook workbook = new XSSFWorkbook(excelFile);
            XSSFSheet sheet = workbook.getSheetAt( index: 2);
            for(Medicine medicine : medicinesForDisease){
                Integer rowNumber = Math.toIntExact(medicine.getId());
                Row row = sheet.getRow(rowNumber);
                Cell cellPregnancy = row.getCell( i: 2);
                Cell cellBreastFeeding = row.getCell( i: 3);
                Cell cellAge = row.getCell( i: 4);
                Integer pregnancyRestrictionXl = Math.toIntExact((long)cellPregnancy.getNumericCellValue());
                Integer breastFeedingRestrictionXl = Math.toIntExact((long)cellBreastFeeding.getNumericCellValue());
                Integer ageRestrictionXl = Math.toIntExact((long) cellAge.getNumericCellValue());
                if (Objects.equals(patientPregnancyStatus, b: "true") && Objects.equals(patientBreastFeedingStatus, b: "true")){
                    if(pregnancyRestrictionXl != 1 && breastFeedingRestrictionXl != 1 && patientAge >= ageRestrictionXl){
                        medicinesAndDescription.put(medicine.getName(), "");
                    }
                } else if (Objects.equals(patientPregnancyStatus, b: "true") && !Objects.equals(patientBreastFeedingStatus, b: "true")) {
                    if(pregnancyRestrictionXl != 1 && patientAge >= ageRestrictionXl){
                        medicinesAndDescription.put(medicine.getName(), "");
                    }
                } else if (!Objects.equals(patientPregnancyStatus, b: "true") && Objects.equals(patientBreastFeedingStatus, b: "true")) {
                    if(breastFeedingRestrictionXl != 1 && patientAge >= ageRestrictionXl){
                        medicinesAndDescription.put(medicine.getName(), "");
                    }
                }else {
                    if(patientAge  >= ageRestrictionXl){
                        medicinesAndDescription.put(medicine.getName(), "");
                    }
                }
                if(Objects.equals(patientPregnancyStatus, b: "true") || Objects.equals(patientBreastFeedingStatus, b: "true")) {
                    if (pregnancyRestrictionXl == 2 && breastFeedingRestrictionXl == 2) {
                        medicinesAndDescription.put(medicine.getName(), "*Ne preporuča se " +
                                "korištenje ovoga lijeka u slučaju trudnoće i dojenja. Koristiti samo ako je neophodno.");
                    }
                    if (pregnancyRestrictionXl == 2 && breastFeedingRestrictionXl != 2) {
                        medicinesAndDescription.put(medicine.getName(), "*Ne preporuča se " +
                                "korištenje ovoga lijeka u slučaju trudnoće. Koristiti samo ako je neophodno.");
                    }
                    if (pregnancyRestrictionXl != 2 && breastFeedingRestrictionXl == 2) {
                        medicinesAndDescription.put(medicine.getName(), "*Ne preporuča se " +
                                "korištenje ovoga lijeka u slučaju dojenja. Koristiti samo ako je neophodno.");
                    }
                }
            }
        } catch (IOException e) {
            e.getStackTrace();
        }
    }
    return medicinesAndDescription;
}
```

Slika 4.27. *Prikaz metode preporuke lijeka*

Na početku metode, inicijalizira se prazna mapa nazvana medicinesAndDescription. Ova mapa će kasnije sadržavati preporučene lijekove zajedno s pripadajućim opisima. Sljedeći korak je

dohvaćanje osobnih karakteristika pacijenta poput statusa trudnoće, statusa dojenja i dobi pacijenta. Ove informacije su ključne jer će se na temelju njih odabrati odgovarajući lijekovi. Ako postoje lijekovi za bolest (provjerava se da li je lista medicinesForDisease prazna), otvara se Excel datoteka koja sadrži informacije o dijagnozama bolesti. Kroz čitanje podataka iz Excel tablice, uspoređuju se s osobnim karakteristikama pacijenta kako bi se filtrirali prikladni lijekovi. Na temelju osobnih karakteristika pacijenta i ograničenja lijekova iz Excela, metoda provodi filtriranje preporučenih lijekova. Lijekovi koji se smatraju sigurnima za pacijenta i odgovaraju njegovim osobnim karakteristikama dodaju se u mapu medicinesAndDescription. Ovisno o statusu trudnoće i dojenja pacijenta, metoda također dodaje posebne poruke upozorenja za lijekove koji imaju posebna ograničenja za te uvjete. Naposljetku, metoda vraća mapu medicinesAndDescription koja sadrži preporučene lijekove zajedno s pripadajućim opisima. Ova mapa je ključni rezultat metode i pruža preporučene lijekove i opise za liječnika prilikom odabira terapije.

Na taj način su dohvaćeni svi lijekovi za dijagnosticiranu bolest i s obzirom na pacijentove karakteristike. Liječnik prilikom propisivanja lijekova dobiva preporuke, no u konačnici unosi lijek prema svome stručnome mišljenju te klikom na gumb Spremi okida API prikazan slikom 4.28 gdje se na postojećeg pacijenta sprema propisani lijek.

```
@PostMapping("/addProscribedMedicineToPatientById/{id}")
public String addProscribedMedicineToPatientById(@PathVariable Long id, @ModelAttribute("patient") Patient medicinePatient){
    Patient patient = patientService.getPatientById(id);
    patient.setProscribedMedicine(medicinePatient.getProscribedMedicine());
    patientRepository.save(patient);
    return "redirect:/medicalRecord" + "/" + id;
}
```

Slika 4.28. *Prikaz API-ja za spremanje preporučenog lijeka*

Nakon preporuke lijeka, liječnik nastavlja s preporukom prehrane koju bi pacijent trebao konzumirati na temelju preporučene bolesti i na temelju pacijentovih alergija na prehrambene namirnice. Prethodno navedenom slikom 4.23, prikazan je način dohvaćanja hrane pomoću varijable dietsOk. Slika 4.29 prikazuje programski kod preporuke prehrane za konzumaciju kao i za njenu ne konzumaciju. Koristi se ista metoda za opisanu funkcionalnost, na način da boolean vrijednost koju funkcija prima inicira da li se radi preporuka za konzumaciju hrane ili za hranu koju treba izbjegavati.

```
@Override
public List<Diet> getAllDietsOkNokForDisease(String disease, boolean isOk, Patient patient) {
    String foodAllergiesOfPatient = patient.getFoodAllergies();
    Integer col = isOk ? 6 : 7;
    List<Diet> dietsAll = getAllDiets();
    List<Diet> resultDietsOkNokRegardingDisease = new ArrayList<>();
    if (disease != null){
        String path = System.getProperty("user.dir") + "\\src\\main\\java\\com\\example\\autoimunihelper\\BazaDijagnozaBolesti.xlsx";
        File file = new File(path);
        FileInputStream excelFile = null;
        try {
            excelFile = new FileInputStream(file);
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
        String dietsOkNokFromExcel = null;
        try {
            XSSFWorkbook workbook = new XSSFWorkbook(excelFile);
            XSSFSheet sheet = workbook.getSheetAt( index: 1);
            Disease currentDisease = diseaseRepository.findDiseaseByName(disease);
            Integer rowNumber = Math.toIntExact(currentDisease.getId());
            Row row = sheet.getRow(rowNumber);
            Cell cellDietsOkNokFromExcel = row.getCell(col);
            dietsOkNokFromExcel = cellDietsOkNokFromExcel.getStringCellValue();
        } catch (IOException e) {
            e.getStackTrace();
        }
        ArrayList<String> dietsSplitted = symptomService.getArrayListFromStringSplittedValuesByComma(dietsOkNokFromExcel);
        for (Diet diet : dietsAll) {
            if(col == 6){
                if(foodAllergiesOfPatient != null){
                    if (dietsSplitted.contains(diet.getName()) && !foodAllergiesOfPatient.contains(diet.getName())) {
                        resultDietsOkNokRegardingDisease.add(diet);
                    }
                }else{
                    if(dietsSplitted.contains(diet.getName())) {
                        resultDietsOkNokRegardingDisease.add(diet);
                    }
                }
            }
            else{
                if(dietsSplitted.contains(diet.getName())) {
                    resultDietsOkNokRegardingDisease.add(diet);
                }
            }
        }
    }
    return resultDietsOkNokRegardingDisease;
}
```

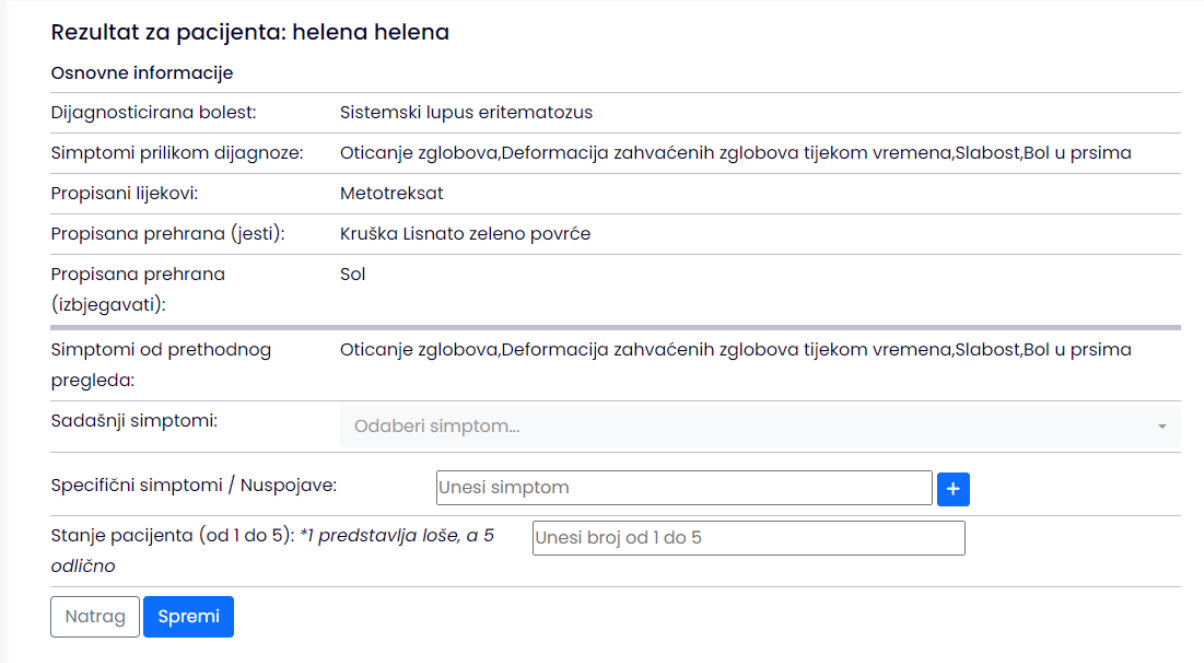
Slika 4.29. *Prikaz načina preporuke načina prehrane*

Na slici 4.29 se nalazi metoda naziva getAllDietsOkNokForDisease koja ima zadaću vratiti prehranu koja je prihvatljiva ili neprihvatljiva za pacijenta s obzirom na pacijentovu bolest i alergije. Ova metoda prima tri parametra: naziv bolesti koju pacijent ima, zastavicu koja označava treba li metoda vratiti prehranu koju pacijent smije jesti ili prehranu koju pacijent ne smije jesti i objekt pacijenta. Zatim, unutar metode se dohvaća informacija o alergijama na hranu koju pacijent ima. Sljedeći korak je odabir stupca u Excel tablici koristeći zastavicu isOk gdje, ukoliko je isOk postavljen na true, odabire se stupac koji označava prehranu koju pacijent smije jesti (6), a ako je isOk postavljen na false, odabire se stupac za prehranu koju pacijent treba izbjegavati (7), a redak koji će se iščitavati iz Excela se određuje na temelju

dijagnosticirane bolesti koju pacijent ima. Prehrana izvučena iz Excela se dalje pohranjuje u varijablu dietsSplitted. Nadalje, dohvaća se sva prehrana iz baze podataka pozivom getAllDiets i pohranjuju u listu dietsAll. Zatim, ukoliko se vraća prehrana koju pacijent treba konzumirati, prolazi se kroz sve prehrambene namirnice u dietsAll i provjerava se njihova kompatibilnost s pacijentovim alergijama na hranu i preporučenim dijetama. Rezultat kompatibilnih dijeta dodaje se u listu resultDietsOkNokRegardingDisease, koja će na kraju biti vraćena kao rezultat metode. Na kraju, metoda vraća listu resultDietsOkNokRegardingDisease, koja sadrži prehrambene namirnice prikladne za pacijenta na temelju bolesti, alergija na hranu ukoliko se vraća prehrana koju pacijent treba jesti i prehrambenog statusa za jesti ili izbjegavati.

4.2.5. Praćenje tijeka liječenja

Liječnik ima mogućnost praćenja tijeka liječenja pacijenta što je ostvareno na način da liječnik, prilikom ispitivanja pacijenta, bilježi nove simptome ako su se oni pojavili prilikom korištenja propisanoga lijeka, unos specifičnih simptoma ili nuspojava ukoliko su se pojavili konzumiranjem propisanoga lijeka te stanje pacijenta u rasponu od 1 do 5, gdje 1 predstavlja najlošije stanje, dok 5 najbolje. Slikom 4.30 prikazan je karton pacijenta gdje liječnik unosi opisano, a slikom 4.31 prikazan je način dohvaćanja i spremanja podataka unutar prozora.



Slika 4.30. *Prikaz načina unosa oporavka pacijenta*

Slika 4.31. *Programski kod API-ja za dohvaćanje i spremanje podataka*

Prva metoda, nazvana addNewResultRecord, koncipirana je kao odgovor na HTTP GET zahtjev te ona vraća prikaz addNewResultRecord koji je prikazan na slici 4.30 skupa sa svim potrebnim podacima o pacijentu i simptomima. Na navedenoj stranici se omogućuje unijeti nove rezultate zdravstvenih pregleda za određenog pacijenta. Unutar navedene metode se dohvaća informacija o pacijentu iz baze podataka pomoću jedinstvenog identifikatora pacijenta, što se ostvaruje pozivom metode patientService.getPatientById(id). Detalji o pacijentu dodaju se u "model", mehanizam koji omogućava prijenos podataka iz kontrolera u prikaz (HTML stranicu), čime se osigurava da se relevantne informacije o pacijentu prikažu na korisničkom sučelju. Zatim se kreira nova instanca klase PatientResult, koja će poslužiti za pohranu novih rezultata zdravstvenog pregleda. Ista ta instanca PatientResult dodaje se u model kako bi omogućila korisnicima unos podataka o novom pregledu. Simptomi dobiveni iz baze podataka putem metode symptomService.getAllSymptoms() također se dodaju u model kako bi se liječniku omogućio odabir novih simptoma ukoliko ih pacijent ima. Metodom se provjerava postoje li već zabilježeni rezultati zdravstvenih pregleda za pacijenta gdje, ukoliko ne postoje prethodni rezultati, simptomi od prethodnog zdravstvenog pregleda postavljaju se temeljem trenutnih simptoma pacijenta definiranih prilikom dijagnoze bolesti. U suprotnom, ako postoje prethodni rezultati, simptomi od prethodnog pregleda postavljaju se temeljem posljednjeg zabilježenog rezultata. Metoda zatim vraća ime predloška addNewResultRecord,

što omogućuje prikazivanje odgovarajuće web stranice za unos novih rezultata zdravstvenih pregleda.

Druga metoda sa slike 4.31 pod imenom addNewResultRecordOfPatient prima tri parametra: id koji predstavlja identifikacijski broj pacijenta koji se izvlači iz URL putanje, patientResult koji predstavlja objekt koji je ispunjen podacima o rezultatima pacijenta te value koji predstavlja dodatni parametar koji se izvlači iz HTTP zahtjeva i sadrži specifične simptome pacijenta. U tijelu metode, linija patientResult.setSpecificSymtpoms(value); postavlja specifične simptome pacijenta na temelju vrijednosti koja je dobivena iz parametra value. Nakon toga, poziva se usluga patientResultService gdje se odvija logika stvaranja novog rezultata pregleda pacijenta pozivom funkcije createPatientResult(id, patientResult) gdje se predaje id od pacijenta i objekt patientResult. Prilikom kreiranja rezultata pregleda stvara se veza N..1 s pacijentom gdje 1 pacijent može imati više rezultata pregleda. U konačnici, metoda vraća preusmjeravanje na URL putanju /patientHealthResults/{id}, gdje se id koristi za identifikaciju pacijenta, omogućujući prikaz tablice rezultata zdravlja pacijenta nakon dodavanja novih podataka.

5. TESTIRANJE OSTVARENOG WEB SUSTAVA

Provjeravanje implementiranog web sustava obavlja se kombinacijom automatskih i ručnih testova koristeći Ruby jezik. Automatski testovi omogućuju brzu provjeru osnovnih funkcionalnosti sustava kao što su navigacija, unos podataka i prikaz rezultata. Ti testovi će biti napisani u Ruby jeziku pomoću alata Cucumber kako bi se osigurala dosljednost i učinkovitost. S druge strane, ručni testovi su ključni za procjenu korisničkog sučelja, iskustva korisnika i drugih aspekata koji se teže mogu automatizirati. Prolaziti će se kroz različite scenarije, simulirajući korisničke interakcije kako bi se prepoznali potencijalni problemi, greške u prikazu ili drugi nedostaci koji bi mogli utjecati na kvalitetu sustava. Spoj automatskih i ručnih testova osigurava sveobuhvatan pristup provjeri sustava i smanjuje rizik od propuštanja ključnih problema prije nego što se sustav stavi u stvarnu uporabu.

5.1. Korišteni alati za testiranje

U procesu testiranja razvijenog web sustava, koristili su se raznovrsni alati koji su se pokazali iznimno korisnima u osiguravanju kvalitete i funkcionalnosti sustava. Ovaj pristup je kombinirao automatizirane i ručne testove, iskorištavajući prednosti VsCode-a i RubyMine-a kao razvojnih okruženja, Cucumber-a za izradu testova koristeći Gherkin jezik te chromedriver-a za automatizaciju interakcija s web preglednikom. VsCode je bio korišten kao izvrsno integrirano razvojno okruženje koje je omogućavalo pisanje i uređivanje Ruby skripti. RubyMine, dizajniran posebno za Ruby jezik, je olakšavao razvoj testova pružajući optimiziranu radnu sredinu. Cucumber je imao ključnu ulogu u stvaranju testova koristeći programski jezik Gherkin koji je čitljiv svima, čak i onima koji nisu upoznati sa tematikom programiranja. Svatko tko pročita napisane scenarije zna koji dio aplikacije se testira i na koji način. To je osiguravalo jasnu komunikaciju među tehničkim i ne tehničkim osobama, uz stvaranje "žive dokumentacije" koja je opisivala ponašanje sustava. Prema [33], zahvaljujući chromedriver-u i Selenium-u, uspjelo se simulirati stvarne korisničke interakcije na web stranicama, što je bilo ključno za ispitivanje ispravnosti funkcionalnosti poput navigacije, unošenja podataka i prikaza rezultata. Kombinacija ovih alata te Ruby jezika i Cucumbera je omogućila sveobuhvatan pristup testiranju te se osigurala automatizirana provjera osnovnih funkcionalnosti sustava. Ručnim testiranjem je također provjerena ispravnost rada web sustava s naglaskom na korisničko iskustvo, no u praksi je bolje što veći dio web sustava pokriti automatiziranim testovima jer su oni brži, gubi se potreba za stalnim ručnim provjerama cijelog sustava te se kod ručnog testiranja može dogoditi da se zaboravi nešto pregledati. Ručnim i

automatiziranim testiranjem je smanjen rizik od propuštanja ključnih problema te je osigurana visoka kvaliteta sustava prije stvarne uporabe.

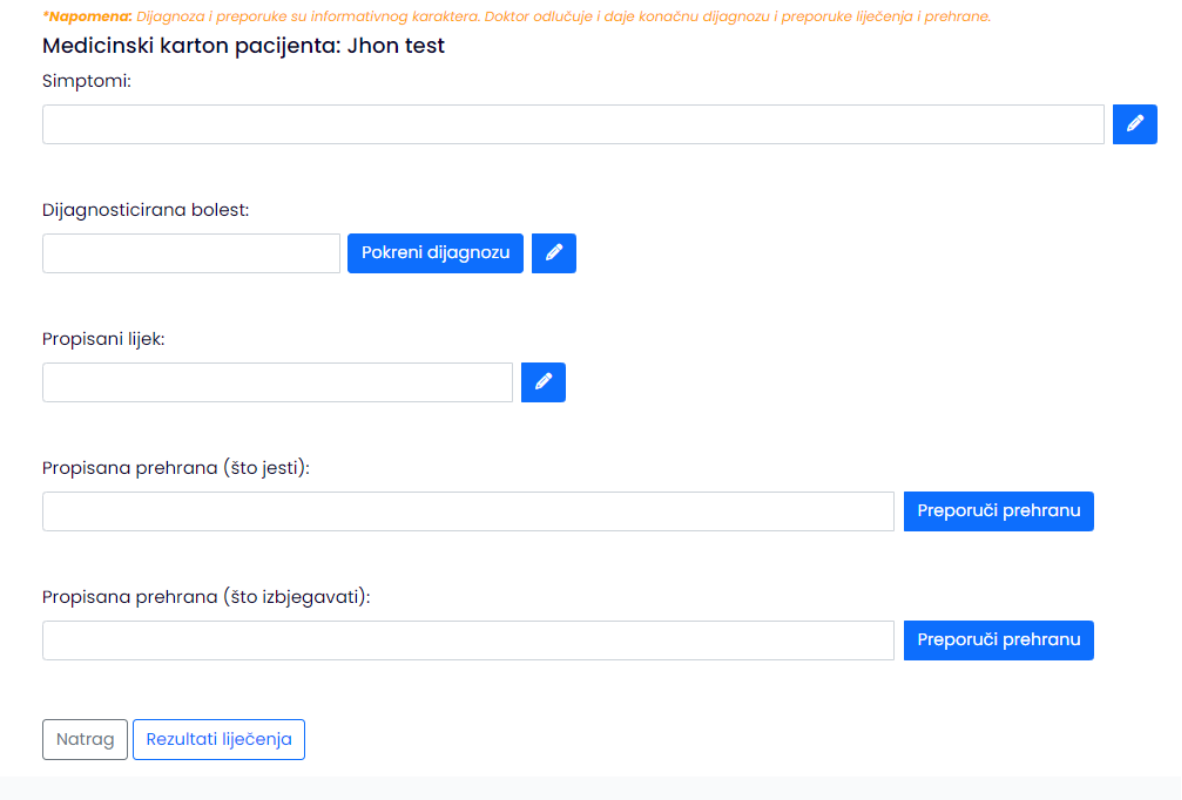
5.2. Testiranje korisničkog sučelja

Kako bi se provjerio ispravan rad korisničkog sučelja, potrebno je prvo definirati testnu specifikaciju po uzoru na koju će se moći izvoditi ručni testovi te zapisivati rezultati testova. U svrhu lakšeg prikaza, unutar rada će biti prikazan manualni test u trenutku kreiranog pacijenta te će se testirati dijagnoza bolesti te preporuke lijeka i prehrane na kreiranom pacijentu. Testna specifikacija se inače zapisuje u poseban dokument, ali u ovome slučaju će biti prikazana kao tablica s potrebnim uputama kako bi se dobilo očekivano ponašanje sustava. Tablicom 5.1 prikazan je jedan od scenarija manualnog testiranja koji će biti objašnjen u radu.

Tablica 5.1. *Prikaz tijeka postupka manualnog testiranja aplikacije*

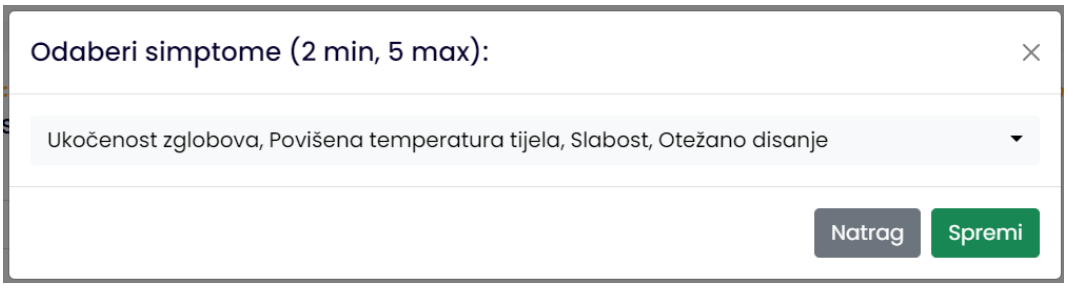
Pred-korak	Koraci	Očekivani rezultat	Stvarni rezultat
1. Doktor mora biti kreiran od strane ADMIN korisnika 2. Doktor treba kreirati pacijenta	1. Otvori medicinski karton pacijenta 2. Klik na dodaj/uredi simptome 3. Odaberi simptome (ukočenost zglobova, Povišena temperatura tijela, Slabost, Otežano disanje) i spremi odabrane podatke 4. Klik na gumb Pokreni dijagnozu 5. Klik na preporuči lijek te odaberi jedan od preporučenih 6. Klik na preporuči prehranu 7. Klik na preporuči prehranu što jesti te odaberi hranu koja je preporučena i spremi podatke 8. Klik na preporuči prehranu što ne jesti i spremi podatke	Svi podaci su prikazani u medicinskome kartonu sa vrijednostima: dijagnosticirana bolest: Sistemski lupus eritematozus, dok su ostali proizvoljno odabrani od strane liječnika	Svi podaci su prikazani u medicinskome kartonu s vrijednostima: dijagnosticirana bolest: Sistemski lupus eritematozus, dok su ostali proizvoljno odabrani od strane liječnika

Prateći napisane korake izvođenja krećemo u manualno testiranje pod uvjetom da su navedeni potrebni koraci iz kolone pred-koraka odrađeni. Klikom na medicinski karton pacijenta otvara se karton koji je prikazan slikom 5.1.

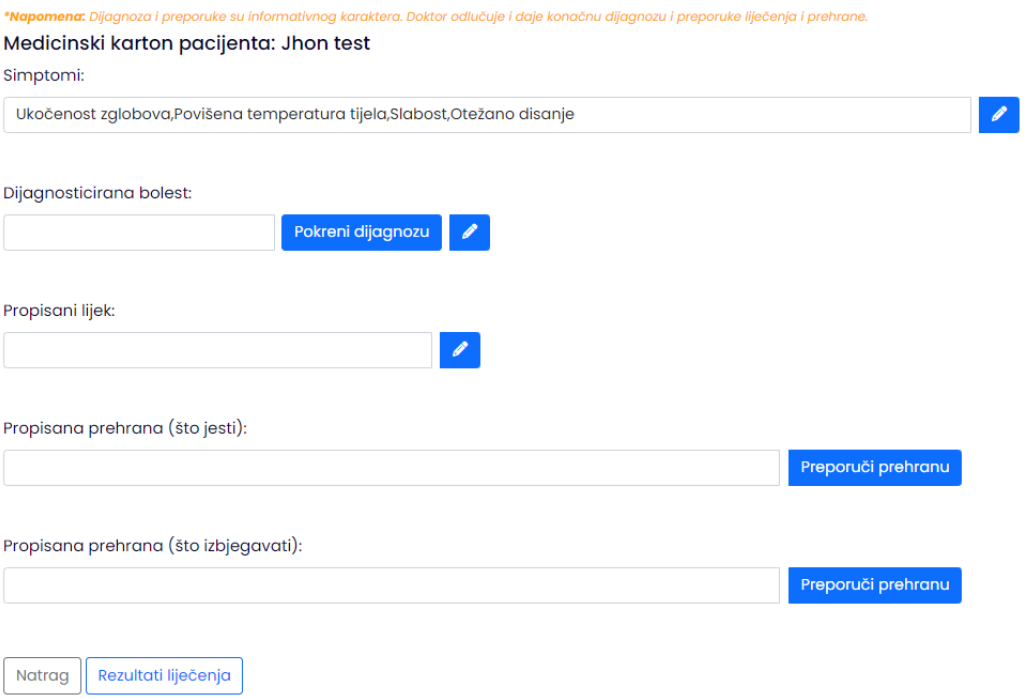


Slika 5.1. *Prikaz kartona pacijenta*

Medicinski karton je prazan te je slijedeći korak kliknuti na gumb Odaberi simptome gdje se liječniku nude simptomi, koje pacijent može imati, za odabir. Odabrani simptomi su prikazani slikom 5.2 čime potvrđujemo da biranje simptoma radi. Klikom na gumb Spremi, očekuje se prikaz tih podataka na medicinskome kartonu što je prikazano slikom 5.3.

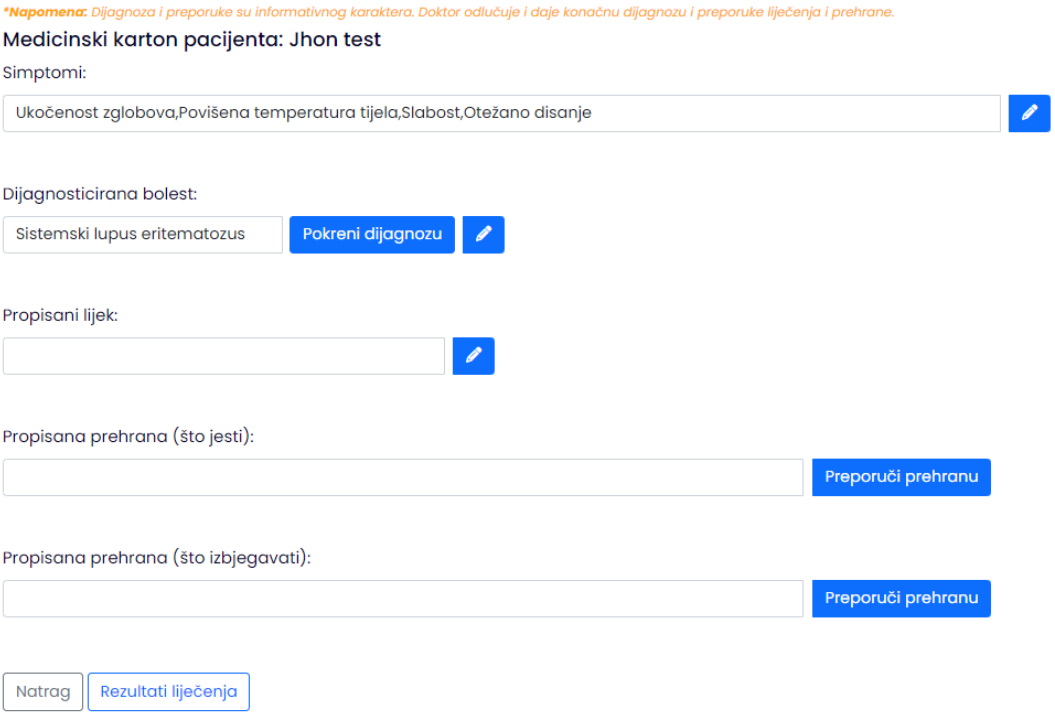


Slika 5.2. *Prikaz odabranih simptoma*



Slika 5.3. *Prikaz simptoma unutar kartona*

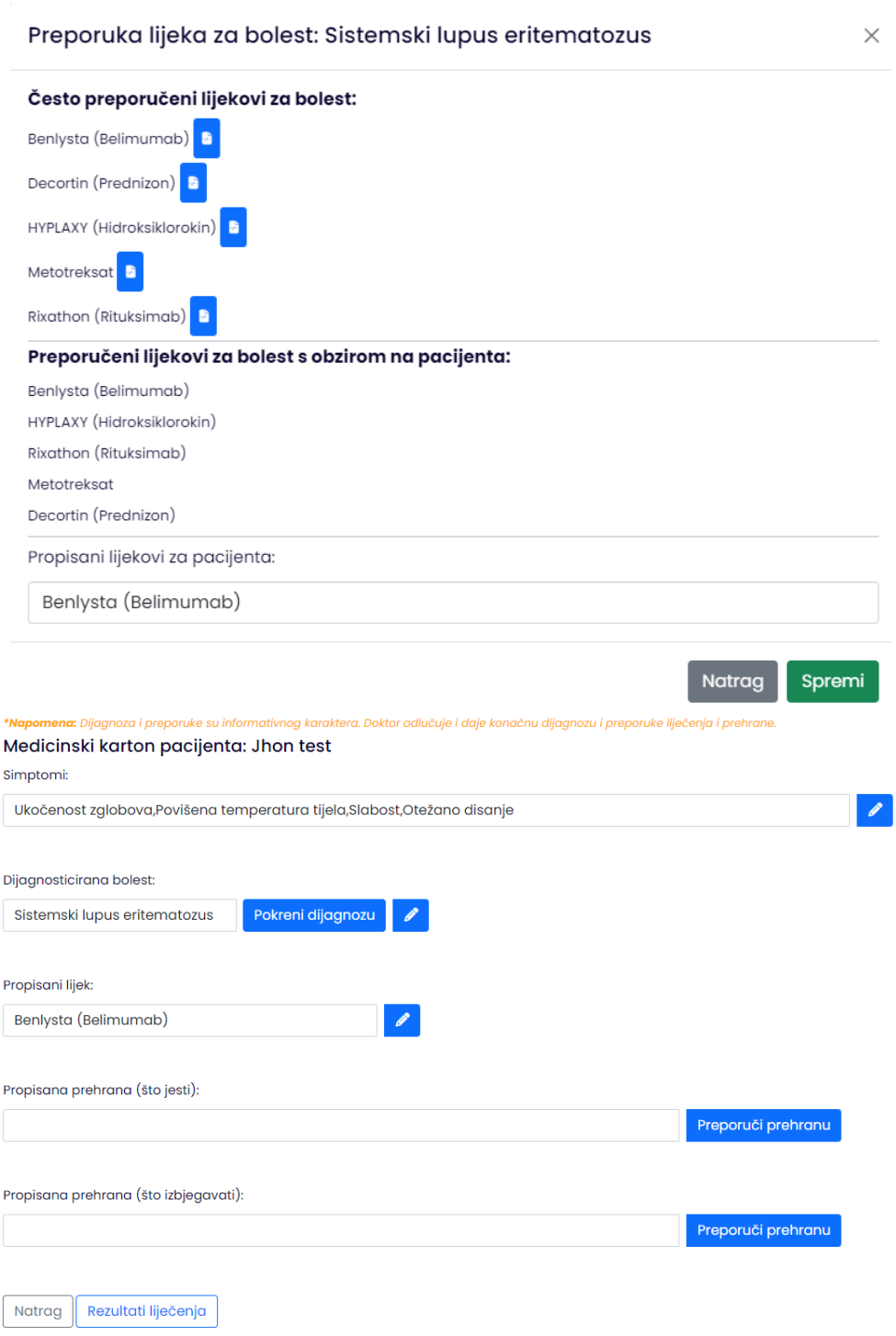
Nakon uspješnog odabira simptoma, slijedi idući korak, a to je klik na pokretanje dijagnoze gdje se kao rezultat očekuje Sistemski lupus što je i prikazano slikom 5.4. Navedenom slikom potvrđuje se ispravna dijagnoza bolesti.



Slika 5.4. *Prikaz uspješne dijagnoze bolesti*

Nakon uspješne dijagnoze preporučuje se lijek za generiranu bolest što se postiže klikom na gumb za preporučivanje lijeka gdje se otvara prozor sa svim preporučenim lijekovima za

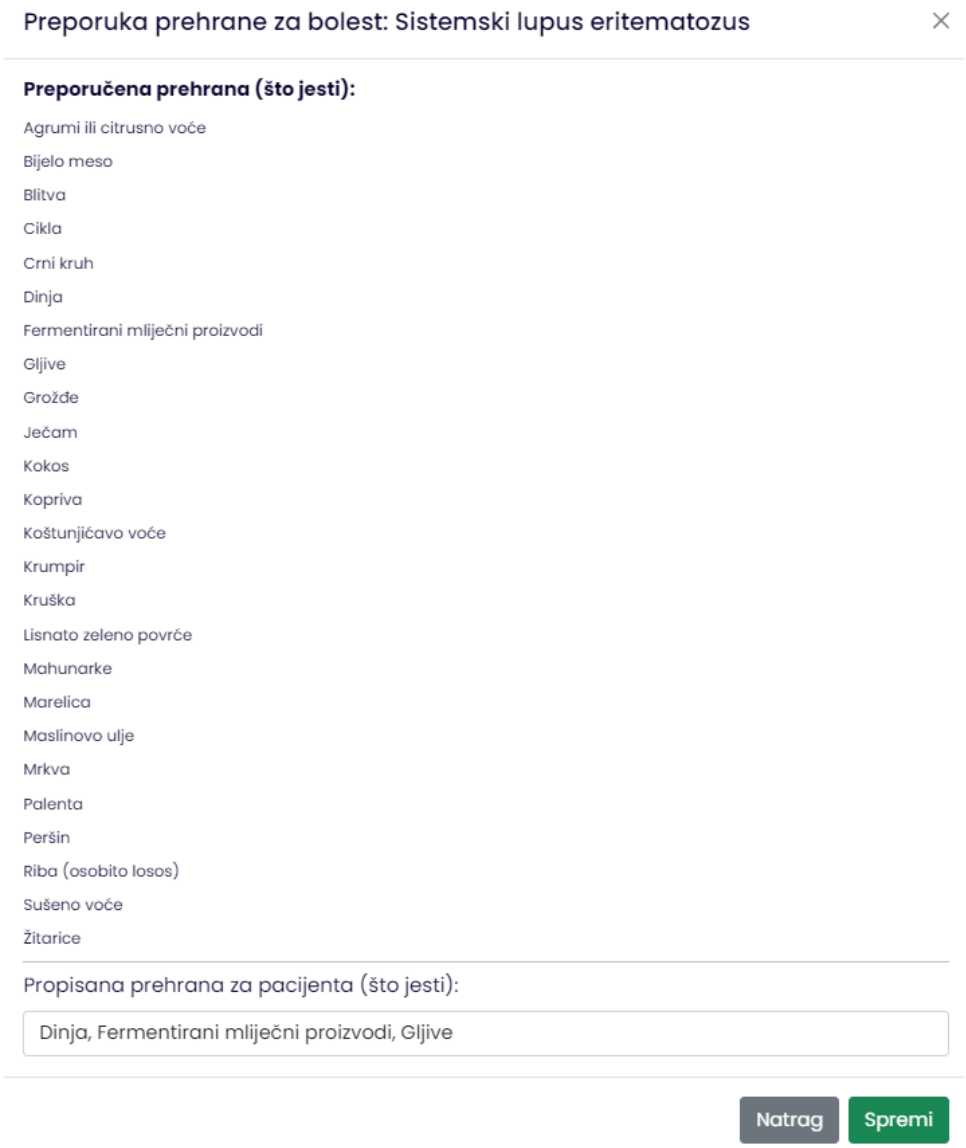
generiranu bolest i s obzirom na pacijentove karakteristike. Liječnik zatim unosi lijekove u polje za propisivanje lijekova te klikom na gumb Spremi prikazuje se medicinski karton pacijenta s unesenim lijekom što je prikazano slikom 5.5.



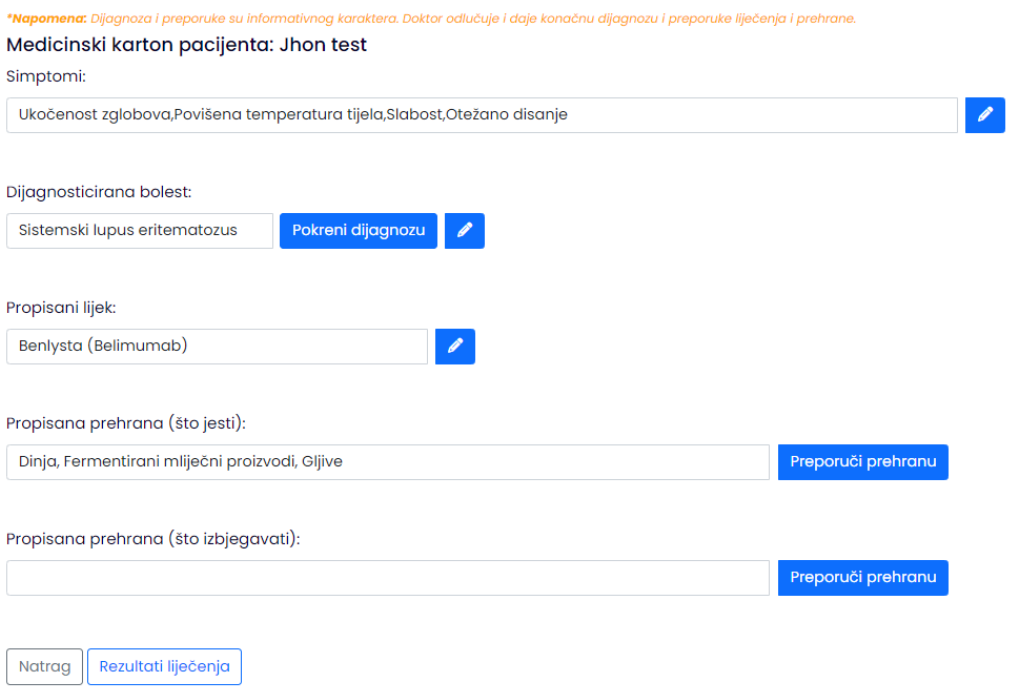
Slika 5.5. *Prikaz uspješnog prikaza preporučenog lijeka*

Zatim u idućem koraku se klika na gumb Preporuči prehranu, nakon čega se otvara prozor gdje će se preporučiti prehrana koju pacijent smije konzumirati. Unesavši namirnice koje pacijent

smije konzumirati, klikom na gumb Spremi, očekuje se prikaz tih namirnica unutar kartona pacijenta. Slika 5.6 prikazuje prozor odabranih prehrambenih namirnica, dok slika 5.7 prikazuje karton s uspješno spremljenim podacima sa slike 5.6.

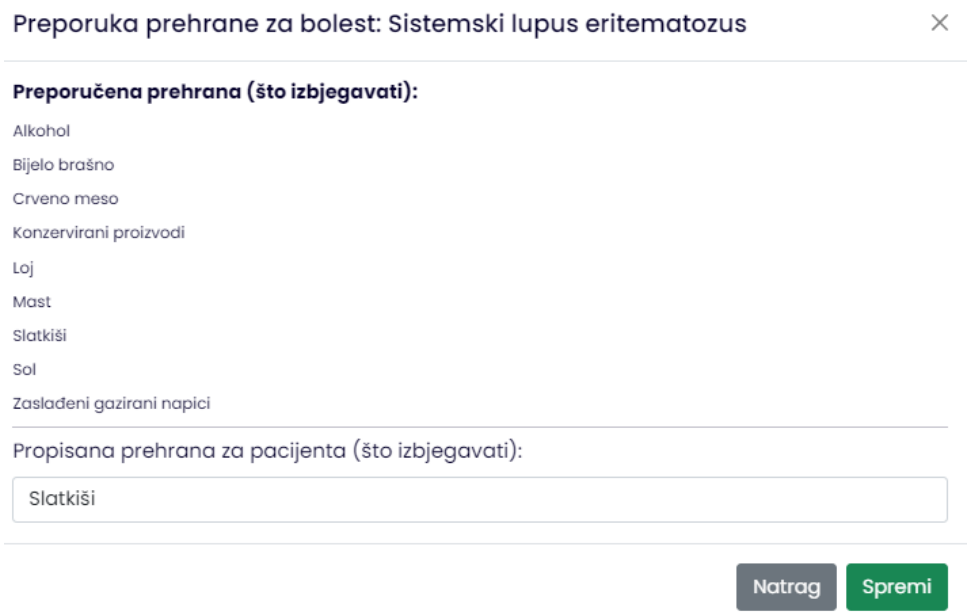


Slika 5.6. *Prikaz odabranih prehrambenih namirnica*

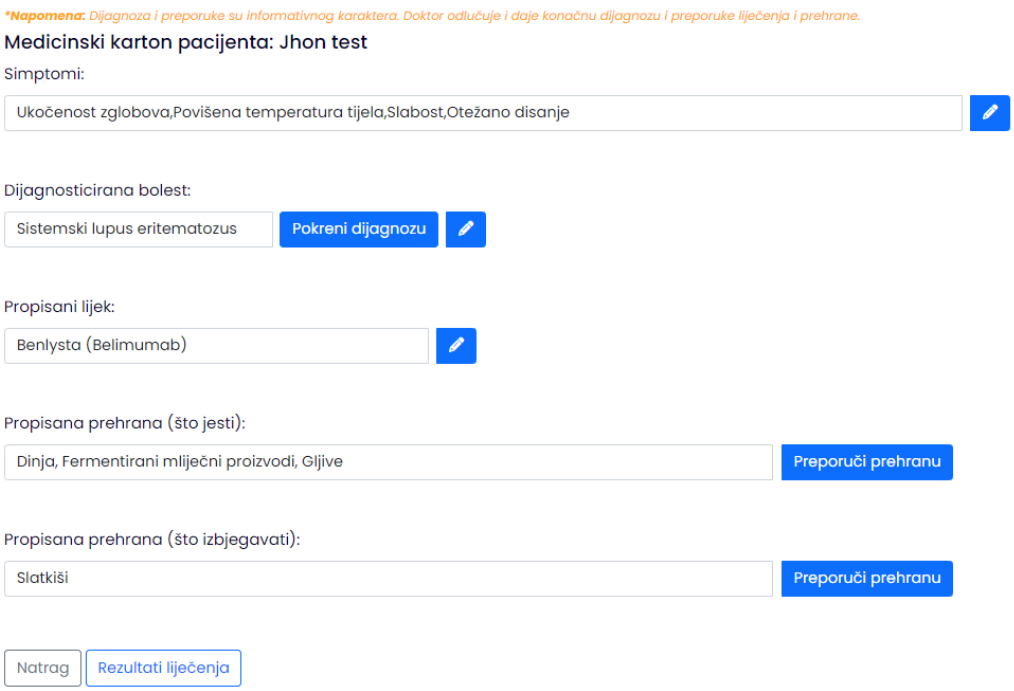


Slika 5.7. *Prikaz uspješnog propisane prehrane*

U zadnjem koraku očekuje se da klikom na preporuči hranu, koju pacijent ne bi trebao konzumirati, imamo slično ponašanje kao i za hranu koju smije konzumirati. Očekuje se uspješno spremanje odabrane hrane te njen prikaz unutar medicinskoga kartona. Slika 5.8 prikazuje odabranu hranu, dok slika 5.9 prikazuje uspješno spremanje podataka te njihov prikaz unutar kartona.



Slika 5.8. *Prikaz odabrane prehrane koju pacijent treba izbjegavati*

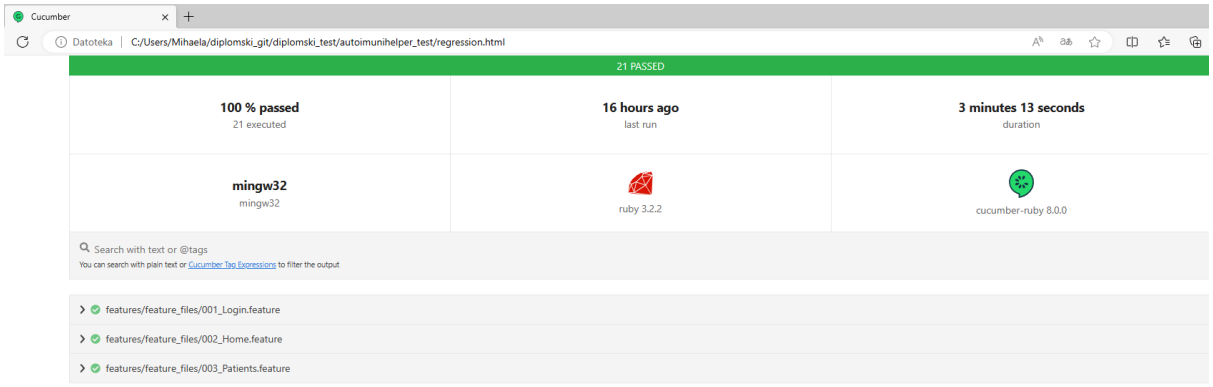


Slika 5.9. *Prikaz prehrane koju pacijent treba izbjegavati unutar kartona*

Izvršivši gore navedene korake iz tablice 5.1, može se zaključiti da sustav radi prema očekivanome ishodu, čime se dolazi do zaključka da sustav radi ispravno.

5.3. Automatizirano testiranje korisničkog sučelja

Svaki sustav treba biti što više pokriven automatiziranim testovima kako bi se osigurala značajno brža provjera ispravnosti rada sustava. S tom namjerom je ovaj sustav pokriven testovima koji provjeravaju ispravnost prijave u sustav, provjeru elemenata na početnoj stranici, kreiranje, izmjenu i brisanje pacijenata kao i prisustvo validacijskih poruka prilikom kreiranja profila pacijenta te se testira ispravnost dijagnoze bolesti i preporuka lijekova i prehrane, kao i ispravnost rada praćenja rezultata tijeka liječenja pacijenta. Izvršenjem svih testova se može generirati html izvješće o prolaznosti svih napisanih testova koji predstavljaju regresiju, a primjer izvršene regresije za ovaj sustav je moguće vidjeti na slikama 5.10 i 5.11.



Slika 5.10. *Prikaz izvješća izvršenih testova*

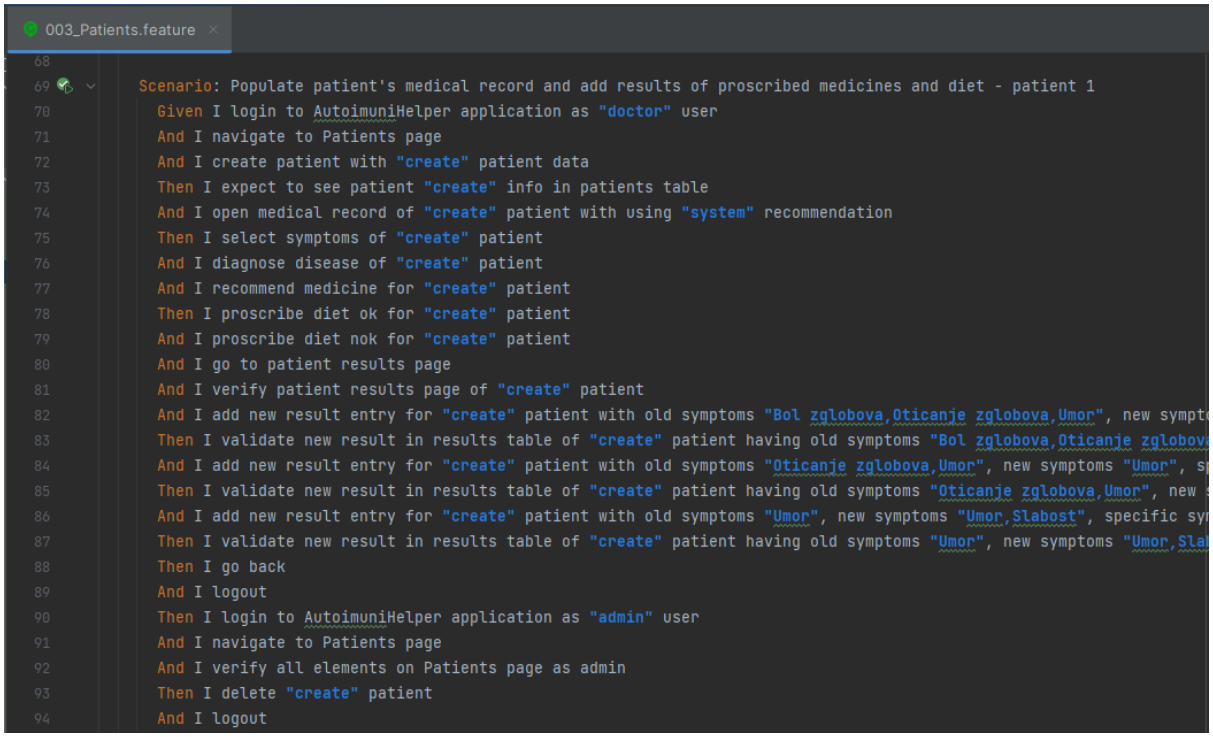
Slika 5.11. *Prikaz testova unutar izvješća za testiranje funkcionalnosti pacijenata*

Kao što je vidljivo na slikama izvješća te prema [31], postoje posebne *feature* datoteke za svaki dio funkcionalnosti sustava gdje su napisani svi scenariji, gdje scenarij predstavlja test, koji se odnose na testiranje te određene funkcionalnosti sustava. Scenariji su pisani Gherkin jezikom koji je lako razumljiv te sadržava naziv scenarija i koraka (engl. *steps*) koji se trebaju izvršiti,

a koji započinju riječima *Given, When, And, Then* ili *But*. Primjer *feature* datoteke za provjeru funkcionalnosti prijave je vidljiv na slici 5.12. Na slici je također vidljivo korištenje *Scenario Outline* i *Examples* dijela čime se omogućava da se scenarij izvrši onoliko puta koliko ima primjera u *Examples* dijelu i to s različitim varijablama datim u *Examples* dijelu. Primjerice, kod scenarija definiranog u četvrtoj liniji, u petoj liniji je definirana varijabla valid_user gdje se pod *Examples* dijelom traži prvi primjer, a to je admin. Tako da se prvi put čitavi scenarij izvršava s admin korisnikom te se, nakon toga, ponovno čitavi scenarij izvršava s drugim primjerom pod *Examples* dijelom, a to je doctor.

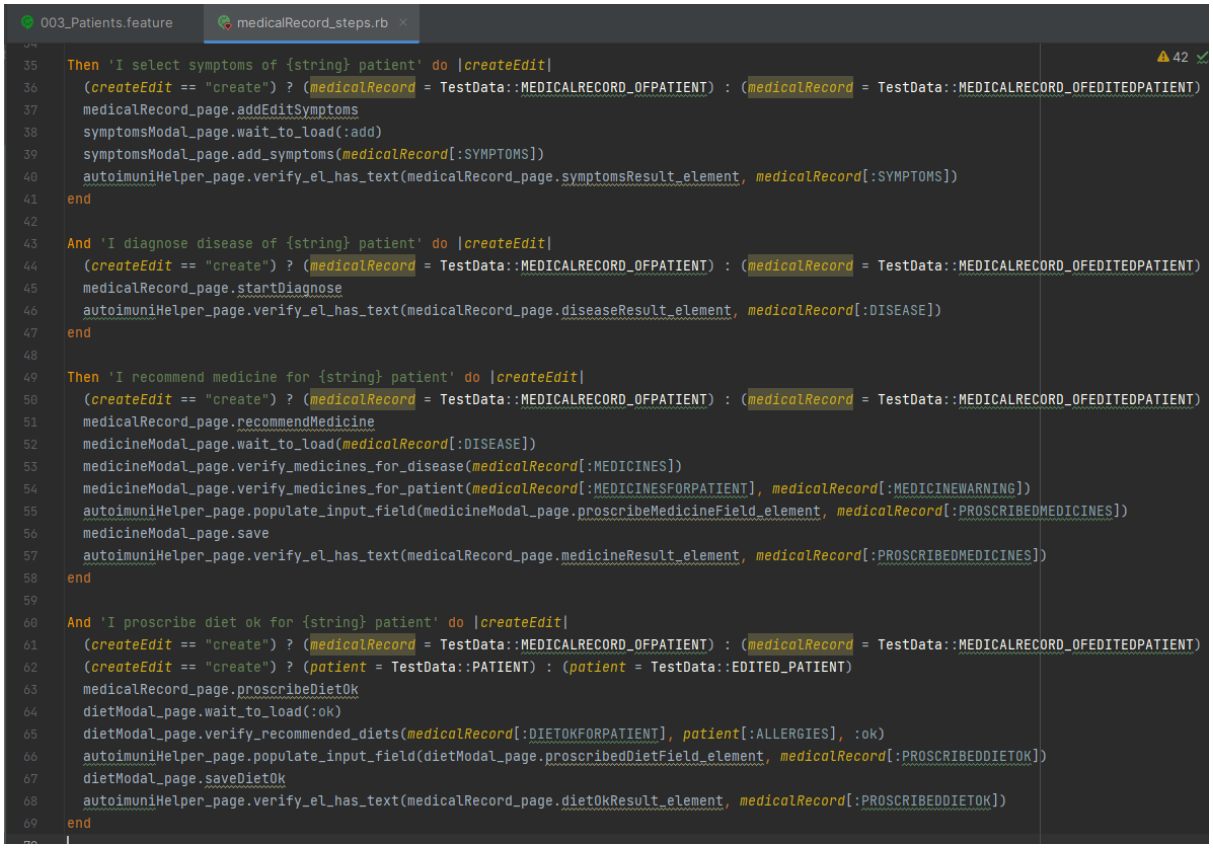
Slika 5.12. *Prikaz feature datoteke za testiranje funkcionalnosti za prijavu u sustav*

Također, scenarij, ukoliko nije definiran kao *Outline*, izvršava se samo jednom, a primjer je moguće vidjeti na slici 5.13. Osim toga, na slici 5.13 je vidljiv test gdje je izvršeno automatizirano testiranje koje obuhvaća test izvršen u ručnom dijelu testiranja, dakle dijagnozu bolesti, propisivanje lijekova i prehrane. Također, ovaj test obuhvaća i kreiranje pacijenta, njegovo brisanje te praćenje rezultata tijeka liječenja.



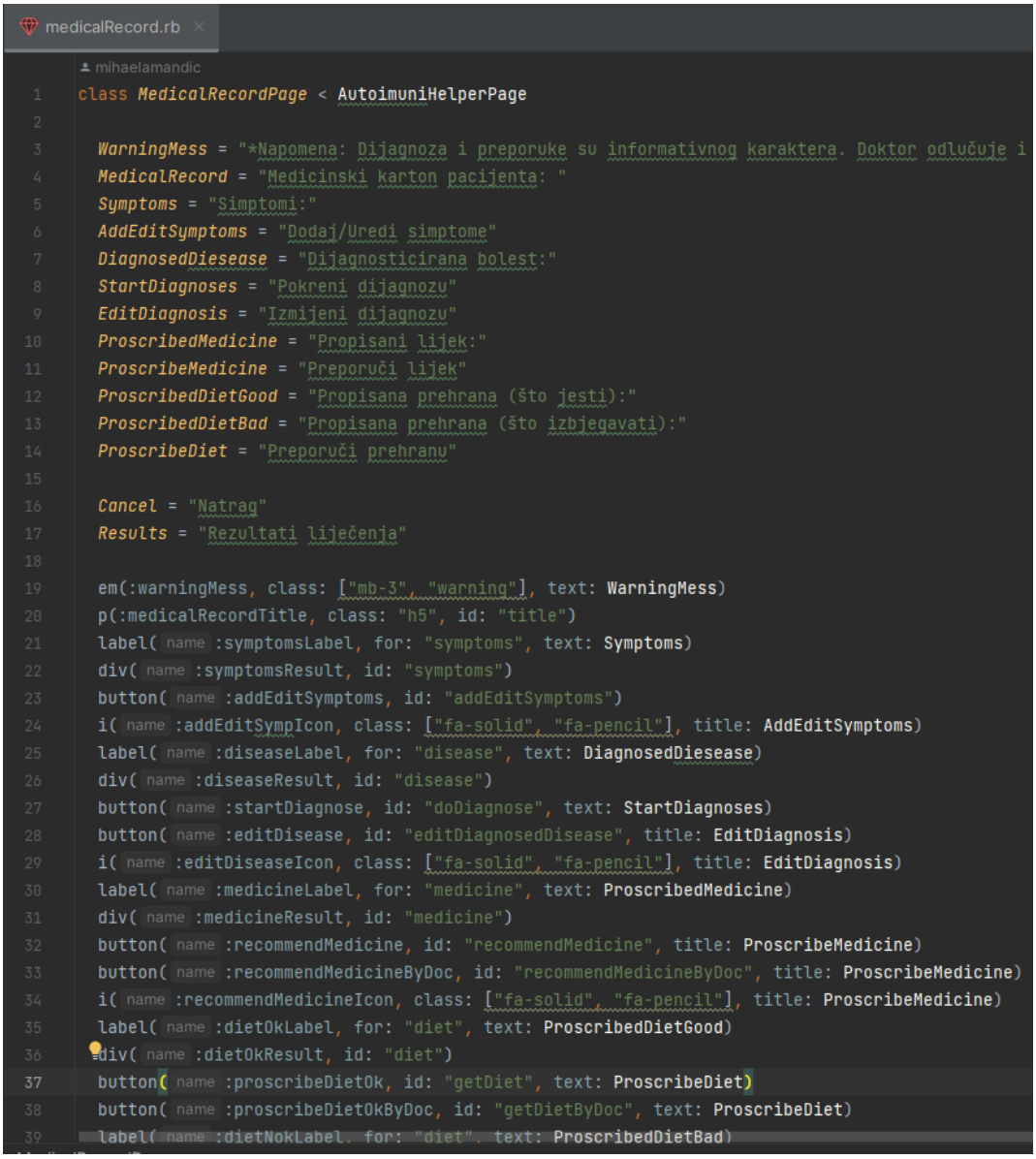
Slika 5.13. *Prikaz scenarija za testiranje dijagnoze bolesti, preporuke lijekova i prehrane te praćenja tijeka liječenja*

Definicija koraka se radi u posebnim datotekama gdje svaka datoteka sadržava korake koji se mogu dogoditi na određenom dijelu funkcionalnosti. Primjerice, u medicalRecord_steps.rb datoteci će biti definirani svi koraci koji su vezani za medicinski karton pacijenta uključujući dodavanje simptoma, dijagnozu bolesti, propisivanje lijekova i prehrane te navigacije koje sadržava medicinski karton stranica što je moguće vidjeti na slici 5.14. Bitno je za napomenuti da se *Given*, *And*, *When*, *Then* i *But* riječi ne moraju podudarati u *feature* datoteci i u datotekama gdje su koraci definirani.



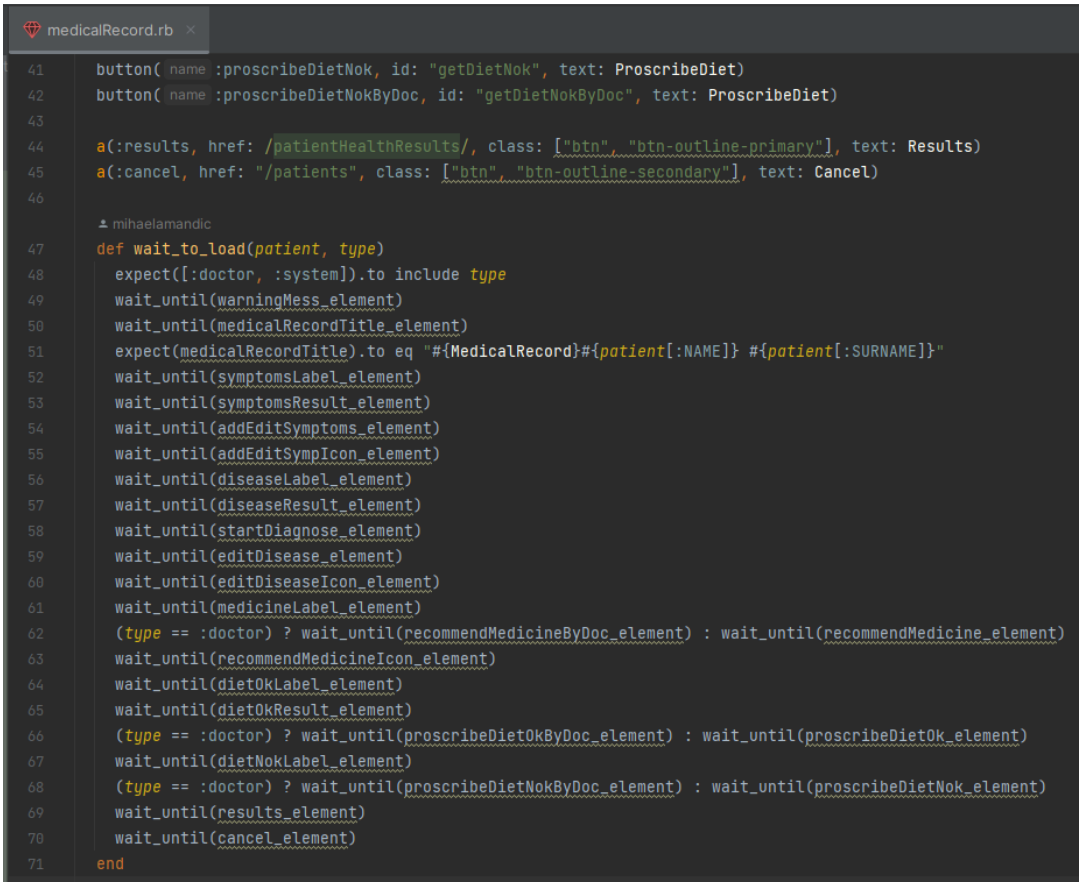
Slika 5.14. *Prikaz definicija koraka*

Unutar koraka se koriste atributi i metode definirane u klasama koje predstavljaju određenu stranicu te, kao što je vidljivo na gornjoj slici 5.14, mogu se učitati u određenu varijablu testni podaci koji će se koristiti primjerice prilikom popunjavanja medicinskoga kartona. Dakle, svaka stranica web sustava bi trebala biti definirana kao zasebna klasa te unutar te klase se trebaju definirati svi elementi vidljivi na stranici koristeći PageObject te se trebaju implementirati sve metode koje se trebaju moći izvršiti prilikom različitih provjera na toj stranici, a primjer je moguće vidjeti na slikama 5.15 te 5.16. Na 5.15 slici je vidljivo kako su na početku definirane varijable koje sadržavaju razne tekstove koji se pojavljuju na stranici, a definirane su na taj način kako bi se olakšala izmjena teksta u testovima, gdje se tekst, ukoliko se promjeni na web sustavu, treba izmijeniti samo unutar varijable, ne treba se mijenjati definicija PageObject elementa. Nadalje, od 19. linije programskoga koda, kreće definicija PageObject elemenata koji su prisutni na web stranici medicinskoga kartona pacijenta.



Slika 5.15. *Prikaz definicije klase koja predstavlja medicinski karton web stranicu*

Na slici 5.16 je vidljiva definicija metode wait_to_load() gdje se provjerava prisutnost svih elemenata na web stranici medicinskoga kartona.



Slika 5.16. *Prikaz definicije metode unutar klase koja predstavlja medicinski karton stranicu*

Ovime se može zaključiti da automatsko testiranje nadmašuje ručno testiranje jer omogućuje bržu i dosljedniju provjeru programskih funkcionalnosti uz smanjenje ljudskih pogrešaka. Automatizirani testovi mogu se lako ponavljati za svaku promjenu u kodu, ubrzavajući razvojni ciklus i osiguravajući visoku razinu kvalitete. S druge strane, ručno testiranje je nužno za situacije koje zahtijevaju ljudsku intuiciju, kreativnost i procjenu, kao i za korisničke sučelja koja zahtijevaju subjektivno vrednovanje.

6. KORIŠTENJE I ISPITIVANJE OSTVARENOG WEB SUSTAVA

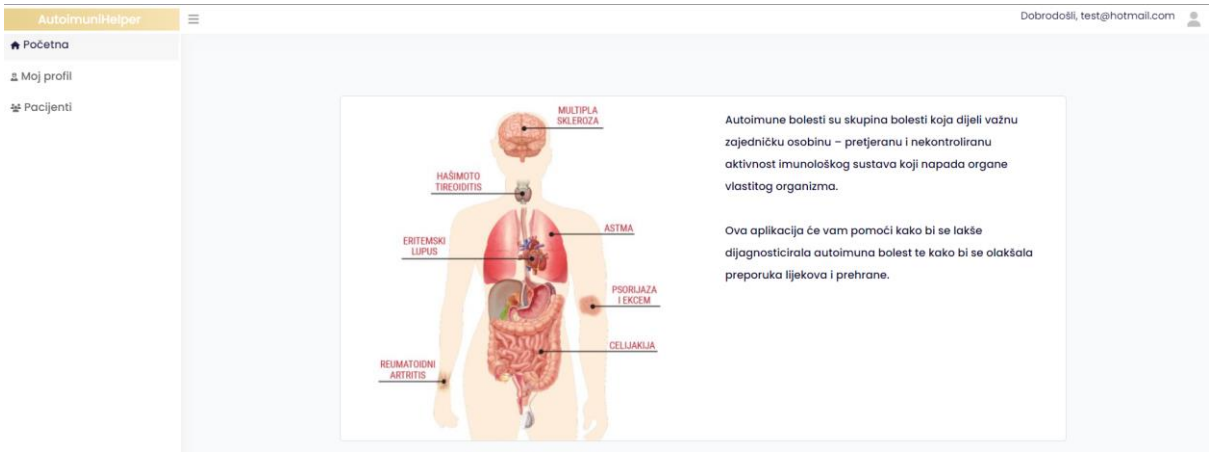
Testiranje i stvarna uporaba razvijenog web sustava imaju ključnu ulogu u osiguravanju kvalitete i uspješnosti web aplikacija. Nakon što je sustav razvijen i implementiran, važno je provesti temeljito ispitivanje kako bi se osiguralo da sustav funkcionira ispravno te da zadovoljava postavljene ciljeve i korisničke zahtjeve. Testiranje i uporaba pomažu u otkrivanju mogućih problema, optimizaciji performansi i osiguravanju pozitivnog korisničkog iskustva.

6.1. Opis načina korištenja

Opis načina korištenja web sustava podrazumijeva detaljni prikaz načina na koje korisnici komuniciraju s različitim aspektima i sučeljem platforme. Ovaj opis prikazuje korake koje korisnici prate kako bi ostvarili svoje zadatke unutar sustava te pruža uvid u različite načine interakcije dostupne korisnicima doprinoseći boljem razumijevanju procesa upotrebe web sustava.

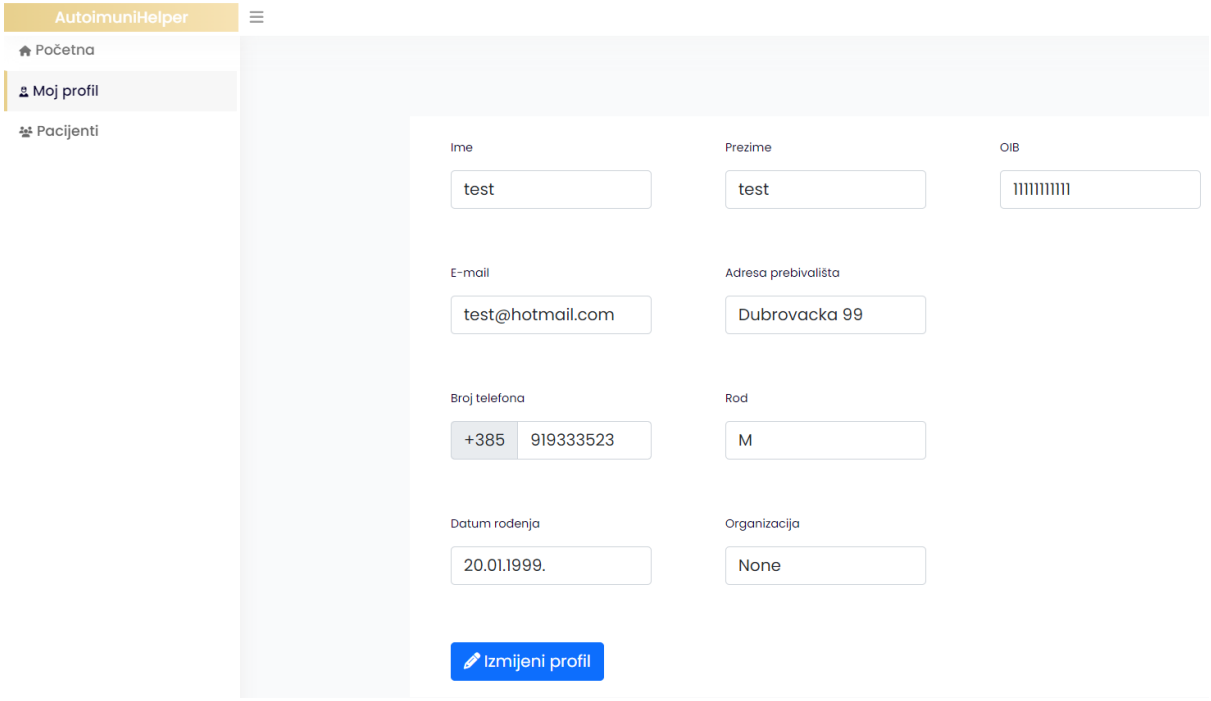
6.1.1. Liječnik kao korisnik

Liječnik je kreiran od strane ADMIN korisnika. Nakon njegovoga stvaranja, liječnik se prijavljuje na web stranicu sa svojim korisničkim podacima te je preusmjeren na početnu stranicu aplikacije koja je prikazana slikom 6.1.



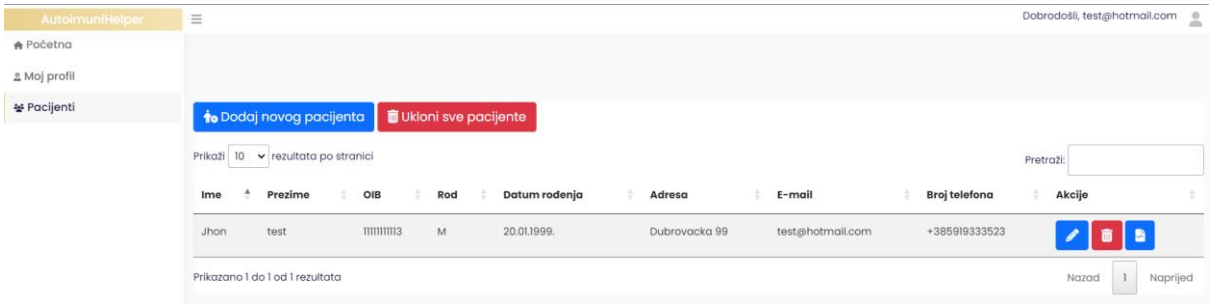
Slika 6.1. *Prikaz početne stranice liječnika*

Prijavom unutar aplikacije, liječnik ima mogućnost pristupa vlastitome profilu gdje su prikazani svi njegovi osobni podaci koje klikom na gumb Izmijeni profil može promijeniti. Slikom 6.2 prikazan je izgled stranice profila liječnika.



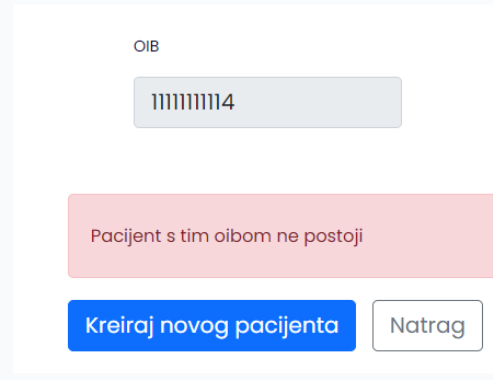
Slika 6.2. *Prikaz profila liječnika*

Liječnik također ima mogućnost odabira izbornika Pacijenti, unutar kojega može kreirati pacijente i obavljati akcije nad njima. Slika 6.3 prikazuje izgled stranice gdje su prikazani pacijenti dodijeljeni određenom liječniku.



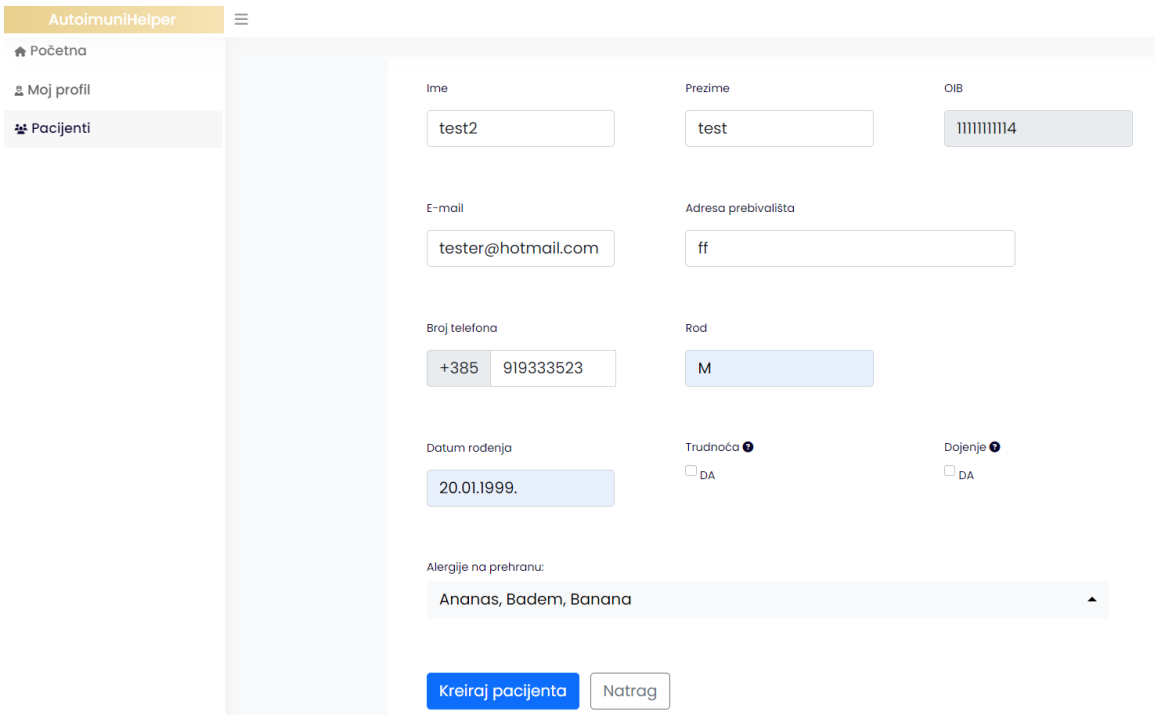
Slika 6.3. *Prikaz stranice pacijenata liječnika*

Klikom na gumb Dodaj novog pacijenta, otvara se prozor koji nudi upisivanje OIB-a pacijenta. Prilikom upisa OIB-a i njegovom potvrdom, sustav provjerava postoji li već u sustavu pacijent s unesenim OIB-om, ako ne postoji, tada liječnik dobiva poruku o statusu pacijenta, što je prikazano slikom 6.4, te nastavlja s kreiranjem pacijenta klikom na gumb Kreiraj novog pacijenta.



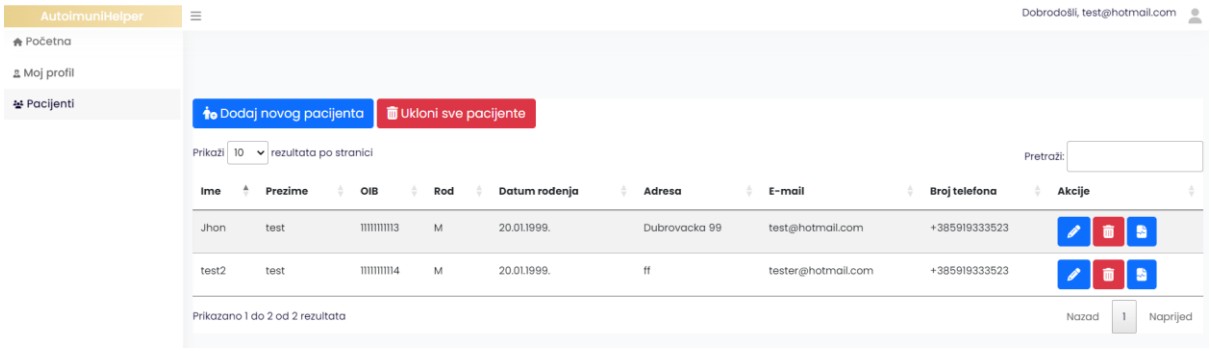
Slika 6.4. *Prikaz unosa OIB-a novoga pacijenta*

Kliknuvši na gumb Kreiraj novog pacijenta, otvara se nova stranica koja nudi unos pacijentovih podataka. Potrebno je unijeti sve tražene podatke. Biranje alergija na prehranu je prikazano kao padajući izbornik koji nudi mnoštvo prehrambenih namirnica na koje je pacijent potencijalno alergičan. Ispunjeni izgled stranice pacijenta prikazan je slikom 6.5.



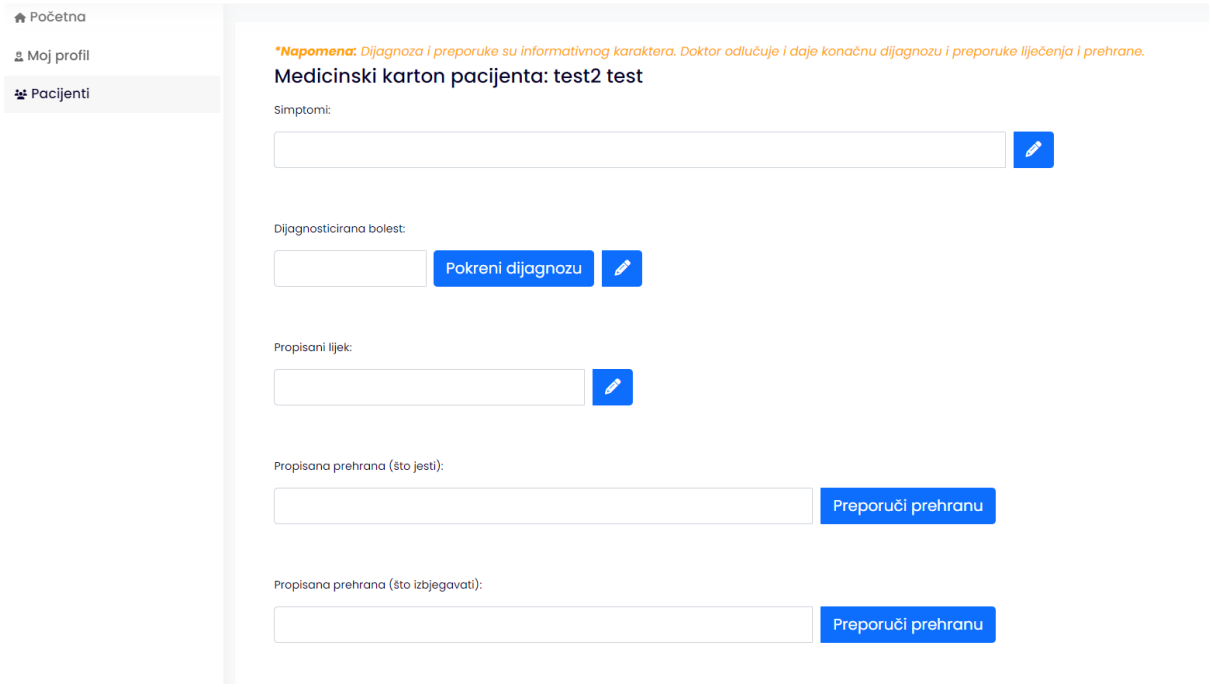
Slika 6.5. *Prikaz ispunjenih podataka pacijenta*

Klikom na gumb Kreiraj pacijenta, pacijent se kreira i prikazuje u tablici pacijenata što je prikazano slikom 6.6.



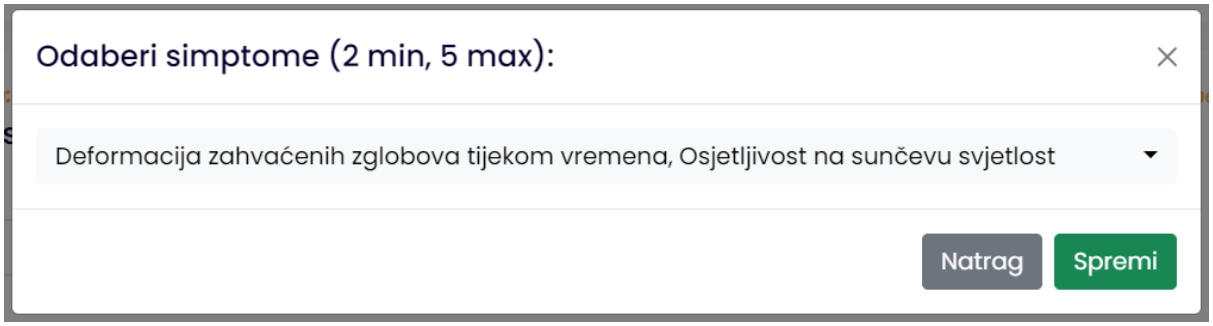
Slika 6.6. *Prikaz kreiranog pacijenta u tablici*

Liječnik zatim ima mogućnost uređivanja kartona pacijenta klikom na prvi gumb u koloni akcije. Klikom na njega otvara se prozor sličan kao kod kreiranja pacijenta gdje liječnik unosi potrebne izmjene. Nakon uspješne izmjene, klikom na gumb Spremi promjene, podaci se ažuriraju unutar tablice. Iduća mogućnost je uklanjanje pacijenta koja uklanja pacijenta s liste praćenja prijavljenog liječnika, no pacijent i dalje postoji u sustavu. Zadnji gumb u koloni akcije predstavlja navigaciju na medicinski karton unutar kojega liječnik može dijagnosticirati bolest, preporučiti lijek i prehranu te pratiti tijek liječenja. Klikom na taj gumb, otvara se medicinski karton pacijenta koji je prikazan slikom 6.7.



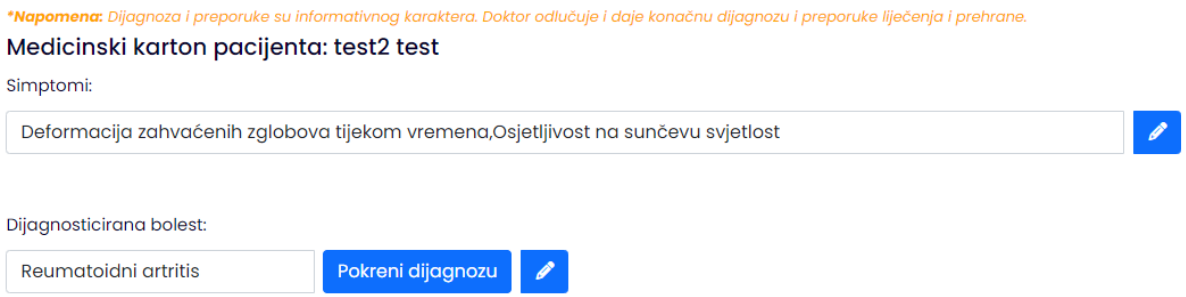
Slika 6.7. *Prikaz medicinskog kartona pacijenta*

Prvo je potrebno odabrati simptome koje pacijent ima. Klikom na gumb za odabiranje simptoma, otvara se padajući izbornik unutar kojega piše kolika je dozvoljena minimalna i maksimalna količina označenih simptoma što je prikazano slikom 6.8.



Slika 6.8. *Prikaz prozora odabira simptoma pacijenta*

Odabirom simptoma i klikom gumba Spremi, simptomi se spremaju u karton pacijenta. Liječnik u idućem koraku ima mogućnost dijagnosticirati bolest gdje klikom na gumb Pokreni dijagnozu, prikazuje se dijagnosticirana bolest unutar medicinskoga kartona što je prikazano slikom 6.9, a ukoliko liječnik želi sam dijagnosticirati bolest, tada klikom na gumb s ikonom olovke se otvara prozor s poljem za unos.



Slika 6.9. *Prikaz dijagnosticirane bolesti unutar kartona*

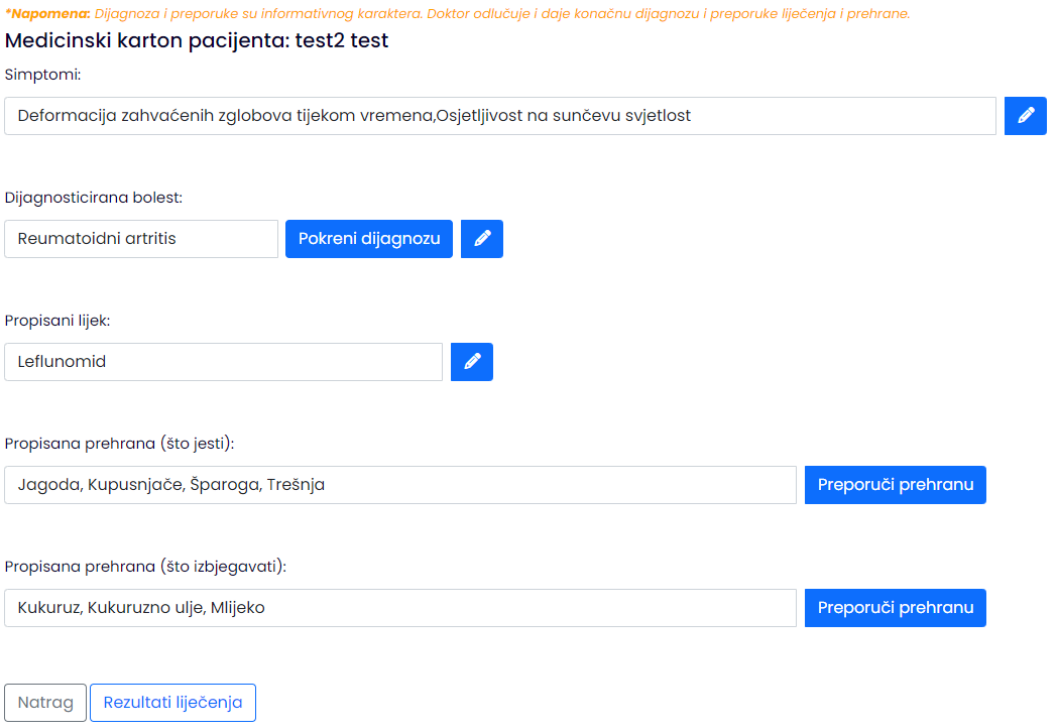
Nakon toga, liječnik dolazi do dijela propisivanja lijeka. Klikom na gumb za preporučivanje lijeka, otvara se prozor koji nudi sve lijekove koji se koriste za dijagnosticiranu bolest i nude se lijekovi specifični za pacijenta na temelju njegovih podataka. Liječnik donosi odluku hoće li propisati neki od generiranih lijekova ili će sam donijeti odluku te propisati neki drugi lijek. Slika 6.10 prikazuje izgled prozora s preporučenim lijekovima za bolest, preporučenim lijekovima s obzirom na bolest i pacijenta, navigacije na upute o lijekovima te poljem za unos propisanog lijeka.

Slika 6.10. *Prikaz prozora odabira preporučenog lijeka*

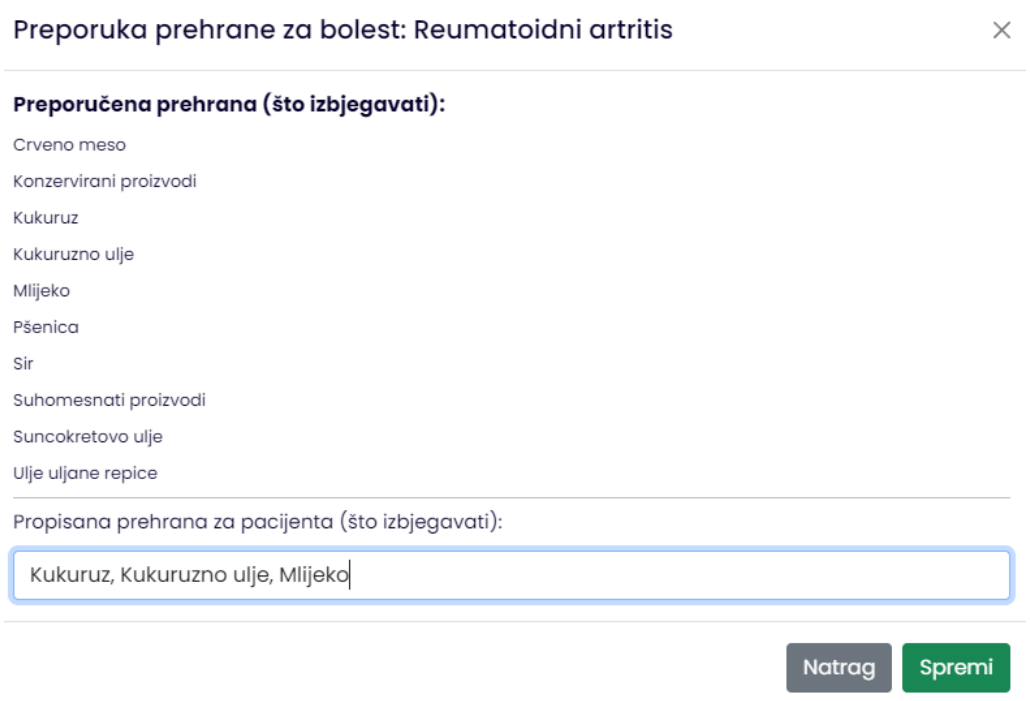
Klikom na gumb Spremi, podatak je spremljen u medicinski karton pacijenta. Slijedeće što liječnik treba odraditi je preporučiti prehranu koju pacijent smije jesti i koju treba izbjegavati. Kliknuvši na gumb Preporuči prehranu, otvara se prozor s preporučenom prehranom na temelju bolesti i alergija koje pacijent ima. Ovdje liječnik ima mogućnost propisati prehranu i klikom na gumb Spremi pohranjuje napisane podatke koji su dalje vidljivi unutar kartona pacijenta. Slika 6.11 prikazuje izgled prozora odabira prehrane koju pacijent treba konzumirati.

Slika 6.11. *Prikaz prozora odabira prehrane koju konzumirati*

Na isti način se odvija i spremanje prehrane koju ne konzumirati. Kliknuvši na gumb Preporuči prehranu koji je smješten pokraj teksta za prehranu koju treba izbjegavati, otvara se identičan prozor kao na slici 6.11 gdje je preporučena hrana koju bi pacijent trebao izbjegavati. Liječnik ondje propisuje prehranu te klikom na gumb Spremi pohranjuje podatak u medicinski karton. U potpunosti ispunjen medicinski karton prikazan je slikom 6.12 dok je slikom 6.13 prikazan prozor odabira prehrane koju ne konzumirati.

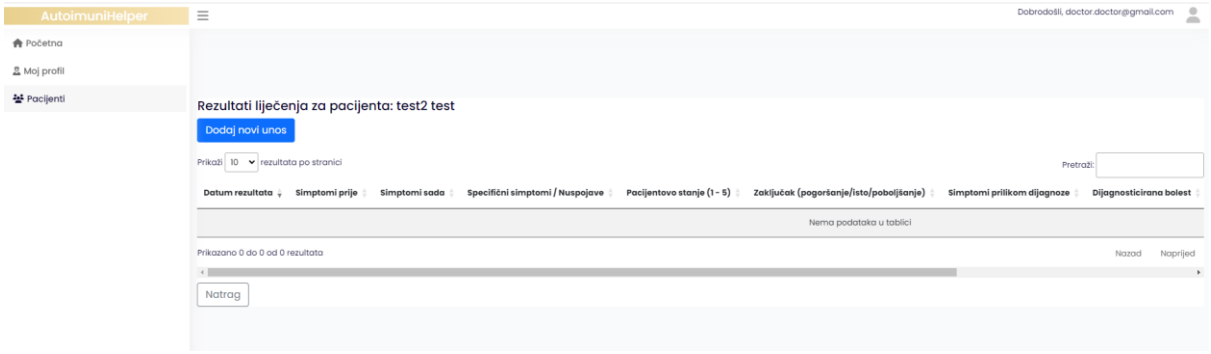


Slika 6.12. *Prikaz medicinskoga kartona pacijenta*



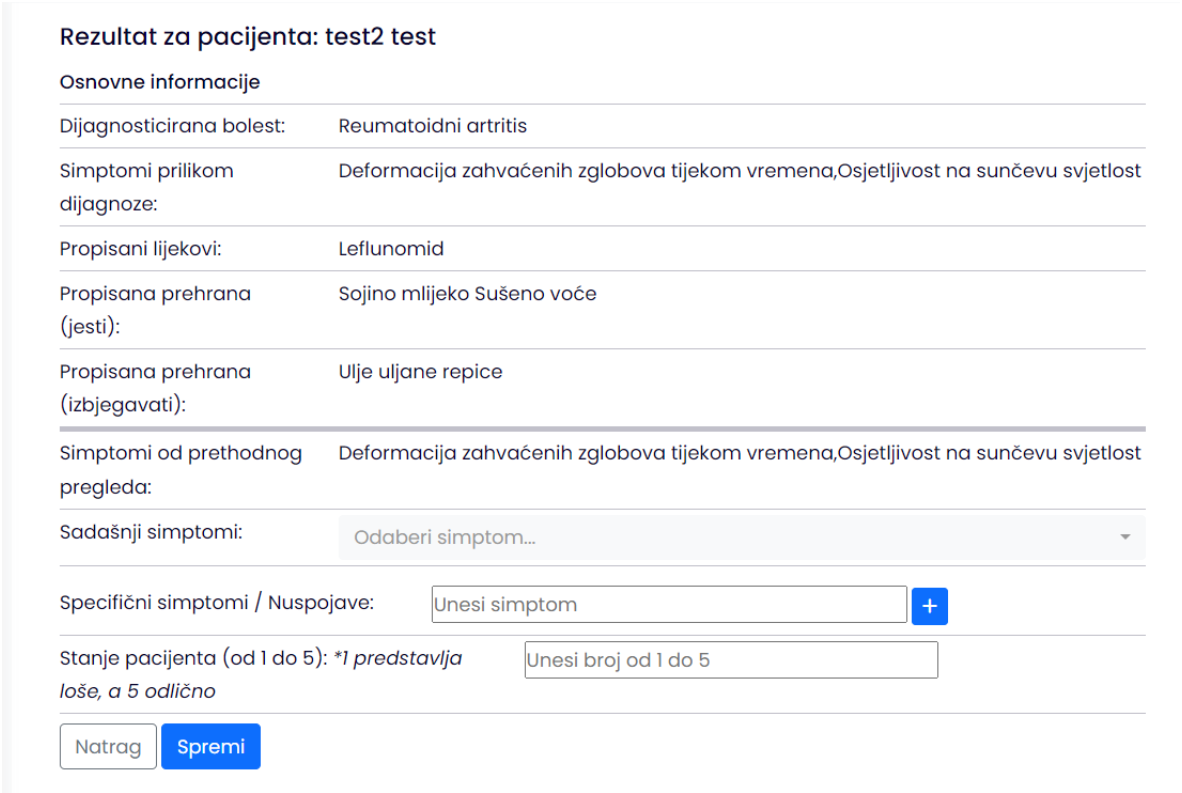
Slika 6.13. *Prikaz prozora odabira prehrane koju pacijent treba izbjegavati*

Liječnik nakon toga ima mogućnost pregleda tijeka liječenja pacijenta. Klikom na gumb Rezultati liječenja unutar medicinskoga kartona, liječnik ima pregled na prijašnje unose stanja pacijenta i mogućnost unosa novog rezultata stanja pacijenta. Slika 6.14 prikazuje izgled opisane web stranice.



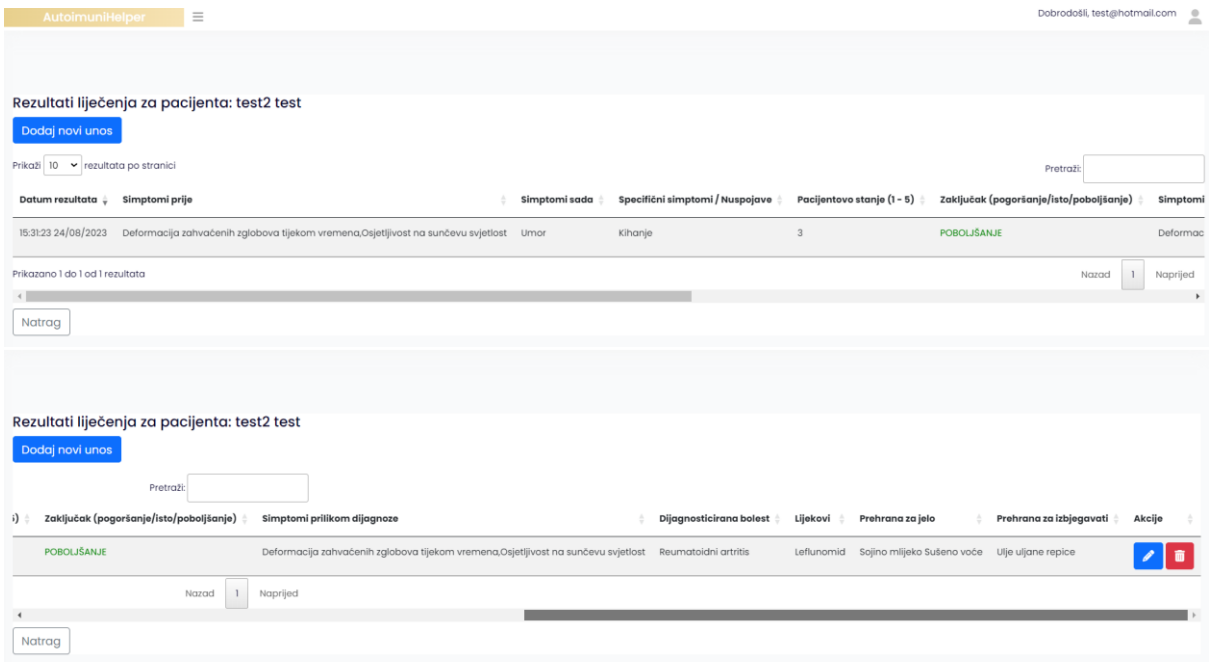
Slika 6.14. *Prikaz stranice tijeka liječenja pacijenta*

Klikom na gumb Dodaj novi unos, otvara se novi prozor, vidljiv na slici 6.15, gdje su prikazane osnovne informacije o pacijentovoj dijagnozi i propisanoj terapiji i prehrani te polja za unos trenutnog stanja pacijenta.



Slika 6.15. *Prikaz rezultata pacijenta*

Liječnik ovdje ima mogućnost unosa pacijentovih trenutnih simptoma specifičnih za autoimune bolesti te unos specifičnih simptoma ili nuspojava koje pacijent ima koristeći propisanu terapiju. Također, liječnik treba unijeti stanje pacijenta u rasponu od 1 do 5 gdje 1 predstavlja loše stanje, dok 5 odlično. Klikom na gumb Spremi, podaci se pohranjuju i pojavljuju unutar tablice rezultata liječenja pacijenta koja je prikazana slikom 6.16.

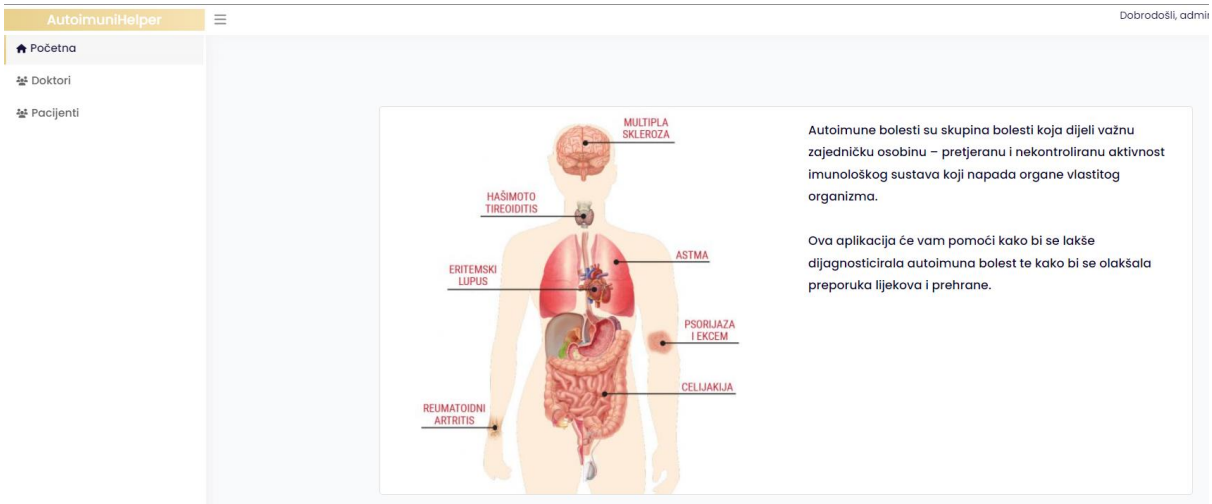


Slika 6.16. *Prikaz popunjene tablice stanja pacijenta*

Također, liječnik ima mogućnost uređivanja ili brisanja već postojećih unosa u tablici.

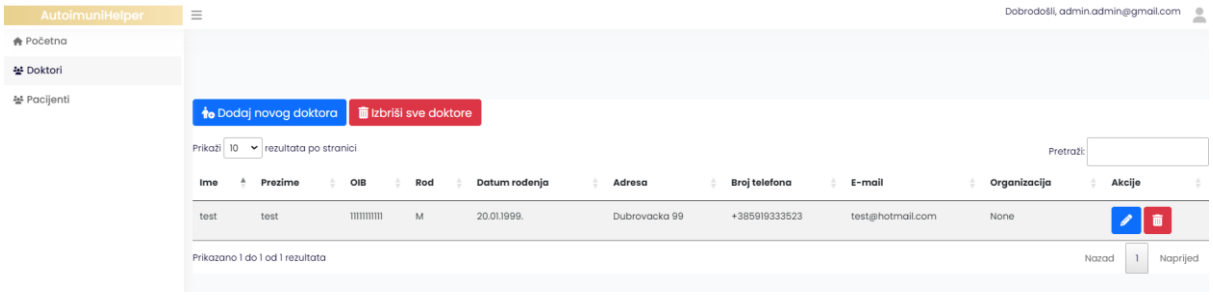
6.1.2. Administrator kao korisnik

ADMIN je tip korisnika koji je kreiran od strane sustava te se prilikom pokretanja sustava provjerava postoji li ADMIN u bazi te, ukoliko ne postoji, tada se kreira. Prijavivši se u sustav, ADMIN je prebačen na početnu stranicu web aplikacije koja je prikazana slikom 6.17.



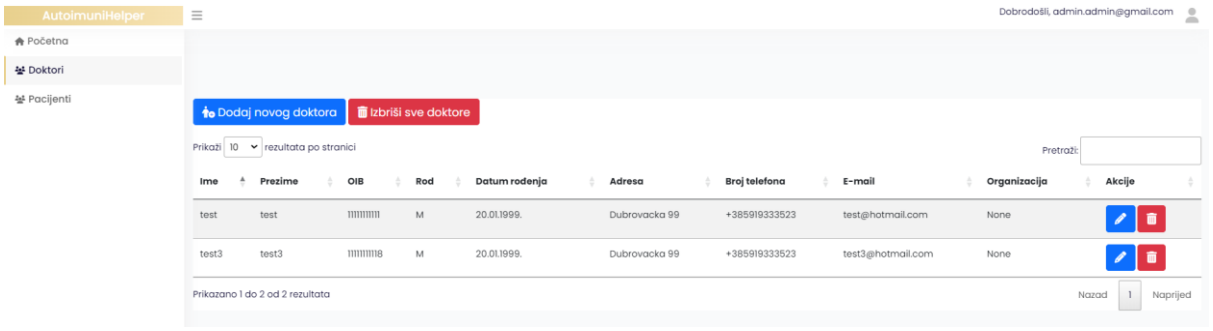
Slika 6.17. *Prikaz početne stranice ADMIN korisnika*

ADMIN svojim ulaskom u sustav ima mogućnost pregleda i kreiranja liječnika te brisanje i arhiviranje postojećih pacijenata. Klikom na izbornik Doktori, prikazuje se tablica s postojećim liječnicima u sustavu te gumb za kreiranje novoga liječnika i brisanje svih liječnika. Slika 6.18 prikazuje izgled stranice Doktori.



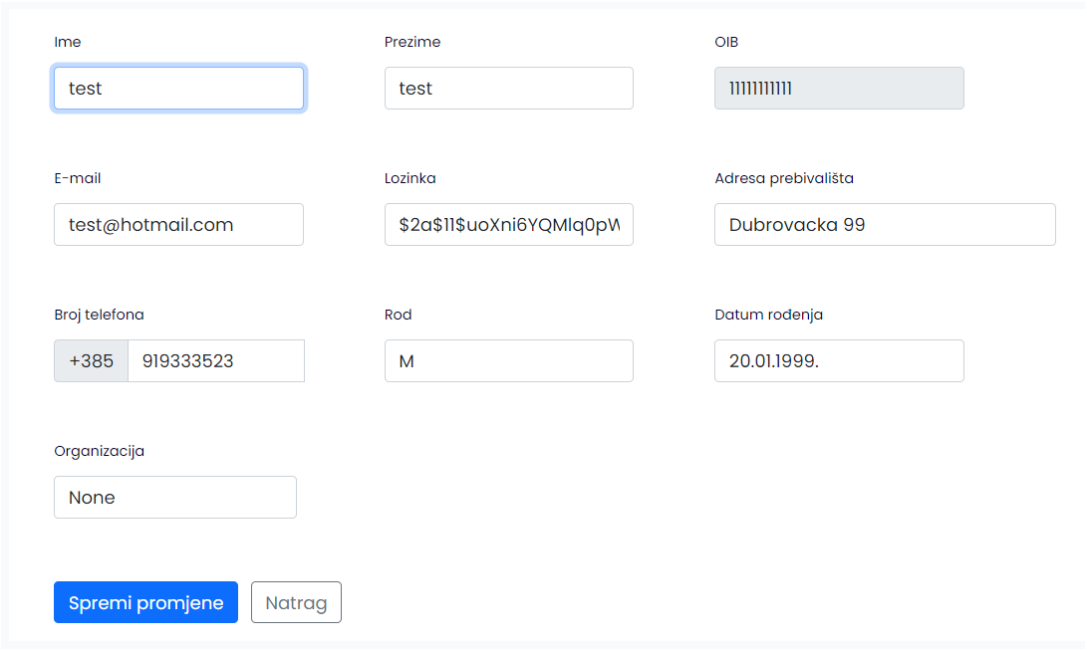
Slika 6.18. *Izgled stranice postojećih liječnika*

Klikom ADMIN-a na gumb Dodaj novog doktora, otvara se stranica unutar koje je potrebno unijeti podatke za novoga liječnika. Nakon unosa svih parametara, klikom na gumb Spremi, podaci se pohranjuju te je novokreirani liječnik prikazan u tablici što se vidi na slici 6.19.



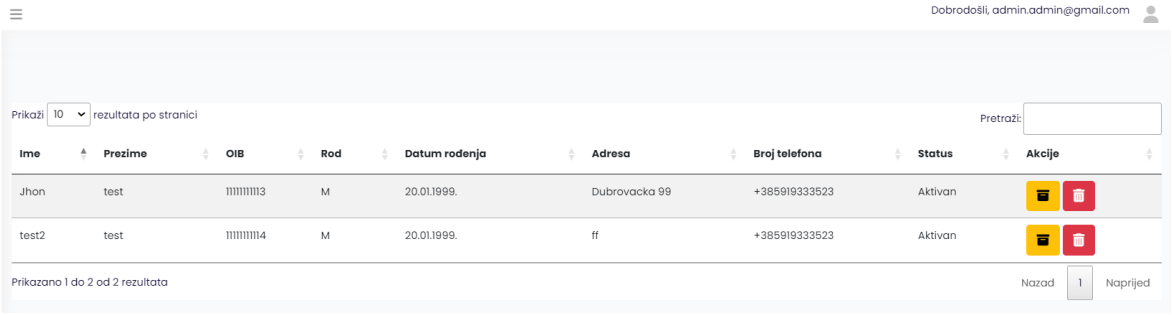
Slika 6.19. *Tablica kreiranih liječnika*

Unutar kolone akcije, ADMIN ima mogućnost uređivanja liječničkih podataka kao i njegovo brisanje iz sustava. Klikom na uređivanje podataka, otvara se identičan prikaz stranice kao i pri kreiranju liječnika, ali ovoga puta unutar polja su već uneseni podaci koje je moguće mijenjati što je prikazano slikom 6.20. Klikom na gumb Spremi promjene, podaci se ažuriraju unutar tablice liječnika. Također postoji gumb Izbriši sve doktore, čime se brišu iz sustava svi postojeći liječnici.



Slika 6.20. *Prozor uređivanja profila liječnika*

Odabirom na izbornik Pacijenti, otvara se stranica svih postojećih pacijenata unutar sustava što je prikazano slikom 6.21. Klikom na prvi gumb unutar kolone akcije, pacijenta je moguće arhivirati nakon čega se pacijent više ne može dodijeliti niti jednom doktoru, a klikom na drugi gumb unutar iste kolone, moguće ga je obrisati.



Slika 6.21. *Tablica svih postojećih pacijenata u sustavu*

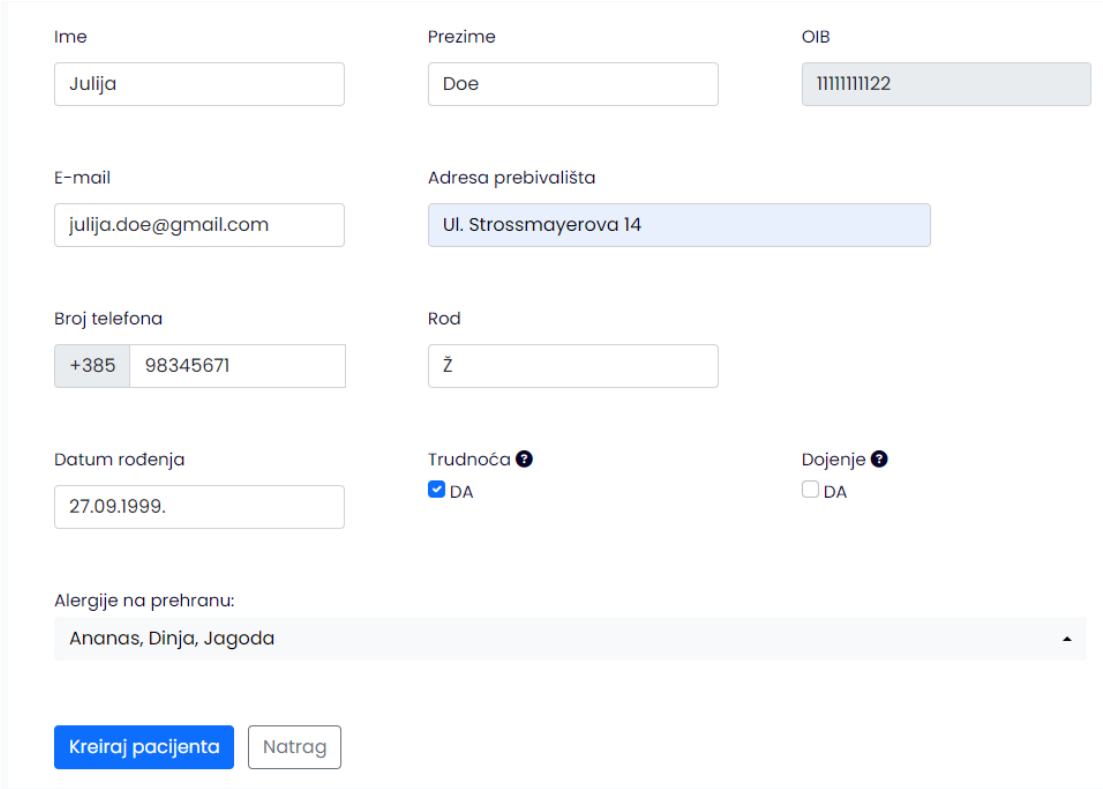
6.2. Uvjeti ispitivanja

Ispitivanje web aplikacije predstavlja ključan korak tijekom njenog razvoja. Ovim procesom se osigurava precizna dijagnoza autoimunih bolesti te preporuka odgovarajućih lijekova i prehrane prema individualnom pacijentovom stanju. Također, ovaj sustav je lako proširiv gdje je potrebno samo dodati nove autoimune bolesti zajedno s njihovim simptomima, godinama kada se najčešće pojavljuju te definirati učestalost pojavljivanja kod muškaraca ili žena te je potrebno dodati nove lijekove i prehranu u bazu podataka. Navedene promjene u bazi neće zahtijevati promijene u sustavu. U nastavku će biti prikazana dva scenarija ispitivanja

aplikacije gdje će se na jednom primjeru testirati dijagnoza bolesti i preporuka lijekova i prehrane kod trudnica, a u drugom primjeru kod djece.

6.2.1. Primjer korištenja 1

Kod prvog primjera testiranja kreirati će se ženski pacijent koji ima 23 godine, trudan je i ima alergiju na prehrambene namirnice: jagoda, ananas i dinja. Profil pacijenta je vidljiv na slici 6.1.

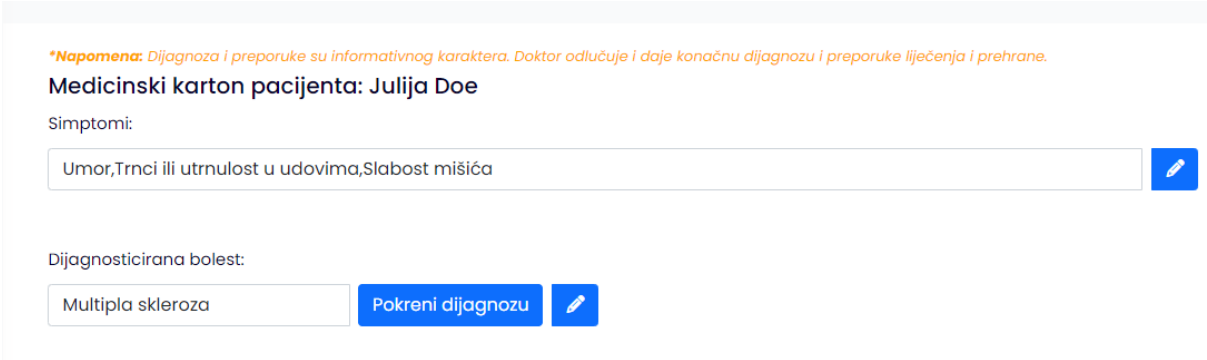


Slika 6.1. *Profil pacijenta kod prvog slučaja testiranja*

Pacijentu će se dodati simptomi: umor, slabost mišića, trnci ili utrnulost u udovima te se očekuje da sustav vrati bolest multipla skleroza. Računanje bolesti se radi na način da se za svaki simptom koji pacijent u listu spreme bolesti te se računa koja se bolest najčešće pojavljuje. U ovom slučaju se najčešće pojavljuje bolest multipla skleroza te se završava računanje bolesti. Na slici 6.2 se mogu vidjeti označeni simptomi te bolesti na koje simptomi upućuju, a na slici 6.3 se može vidjeti ispravnost rezultata dijagnoze bolesti.

B	C
SIMPTOMI	BOLEST
Bol zglobova	Reumatoidni artritis, Sistemski lupus eritematozus, Psorijaza, Sjögrenov sindrom
Oticanje zglobova	Reumatoidni artritis, Sistemski lupus eritematozus, Sjögrenov sindrom
Ukočenost zglobova	Reumatoidni artritis, Psorijaza
Umor	Reumatoidni artritis, Sistemski lupus eritematozus, Multipla skleroza , Hashimotov tireoiditis, Tip 1 dijabetes, Celiakija, Crohnova bolest
Povišena temperatura tijela	Reumatoidni artritis,Sistemski lupus eritematozus, Crohnova bolest
Gubitak apetita	Reumatoidni artritis
Deformacija zahvaćenih zglobova tijekom vremena	Reumatoidni artritis
Slabost	Sistemski lupus eritematozus
Kožni osip (osip u obliku leptira na licu)	Sistemski lupus eritematozus
Osjetljivost na sunčevu svjetlost	Sistemski lupus eritematozus
Problemi s bubrezima	Sistemski lupus eritematozus
Bol u prsima	Sistemski lupus eritematozus
Otežano disanje	Sistemski lupus eritematozus
Poteškoće pri hodanju ili problemi s koordinacijom	Multipla skleroza
Trnci ili utrnulost u udovima	Multipla skleroza
Slabost mišića	Multipla skleroza

Slika 6.2. *Baza za dijagnozu bolesti*



Slika 6.3. *Prikaz ispravne dijagnoze bolesti*

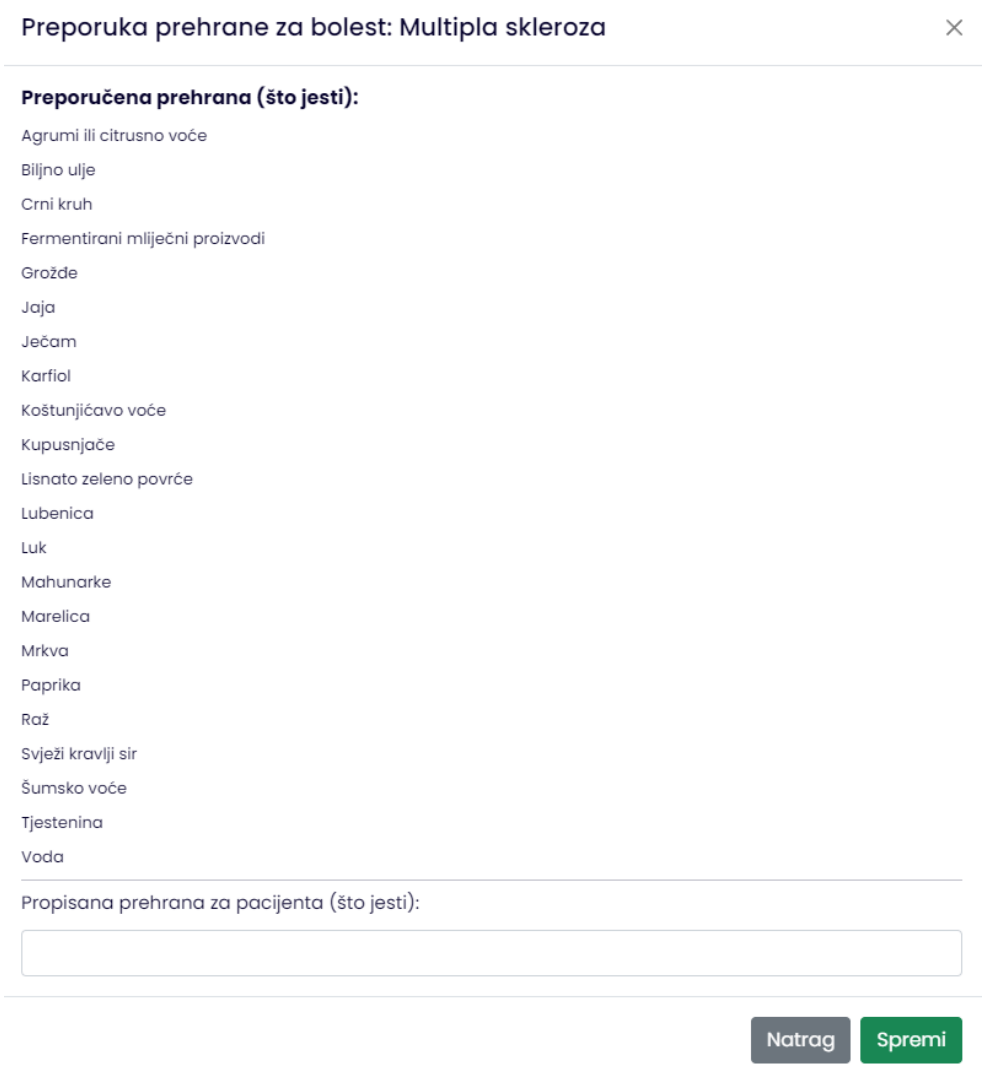
Nadalje, lijekove koje sustav treba preporučiti kao često preporučene za dijagnosticiranu bolest su: AVONEX (Interferon beta-1a), Betaferon (Interferon beta-1b), Extavia (Interferon beta-1b), Glatirameracetat Pliva, Tysabri (Natalizumab), Arlesias (Fingolimod), Inzolfi (Fingolimod), Aregalu (Teriflunomid), Tergio (Teriflunomid). Od navedenih lijekova, preporučeni lijekovi za pacijenta će biti samo Extavia (Interferon beta-1b), Glatirameracetat Pliva, Tysabri (Natalizumab), AVONEX (Interferon beta-1a) i Betaferon (Interferon beta-1b) jer su lijekovi Arlesias (Fingolimod), Inzolfi (Fingolimod), Aregalu (Teriflunomid) i Tergio (Teriflunomid) zabranjeni kod trudnica. Na slici 6.4 je prikazana baza s vidljivim lijekovima i pravilima za trudnoću gdje 0 predstavlja da trudnica smije piti lijek, 1 da nikako ne smije piti lijek gdje ti lijekovi neće biti preporučeni liječniku, a 2 da se ne preporuča lijek te u tome slučaju se ispisuje obavijest liječniku prilikom preporuke lijeka. Ispravnost preporuke lijekova od strane sustava je vidljiva na slici 6.5.

LIJEK	TRUDNOĆA(0 - S
Amgevita (Adalimumab)	2
Apidra (inzulin glulizin)	0
Aregalu (Teriflunomid)	1
Arlesias (Fingolimod)	1
Atenolol Pliva	0
Athyrazol (Metimazol)	2
AVONEX (Interferon beta-1a)	0
BELODERM krema (Betametazon)	0
BELODERM mast (Betametazon)	0
Benlysta (Belimumab)	2
Betaferon (Interferon beta-1b)	0
Betaloc ZOK (Metoprolol)	2
Betametazon salicilatna kiselina Belupo	2
Budosan	2
Ciklosporin Alkaloid	2
CORTEF (Hidrokortizon)	2
Decortin (Prednizon)	2
Diprogenta krema	1
Diprogenta mast	1
Enbrel (Etanercept)	2
Entyvio (Vedolizumab)	2
Euthyrox (Levotiroksin (T4 hormon))	0
Extavia (Interferon beta-1b)	2
Flixabi (Infliksimab)	2
Gastal	2
Glatirameracetat Pliva	2
HEMANGIOL (Propranolol)	1
Humalog (inzulin lispro)	0
HYPLAXY (Hidroksiklorokin)	2
Imuran (Azatioprin)	1
Inzolfi (Fingolimod)	1
Tergio (Teriflunomid)	1
Tresiba (inzulin degludek)	2

Slika 6.4. *Baza lijekova s ograničenjima prilikom trudnoće*

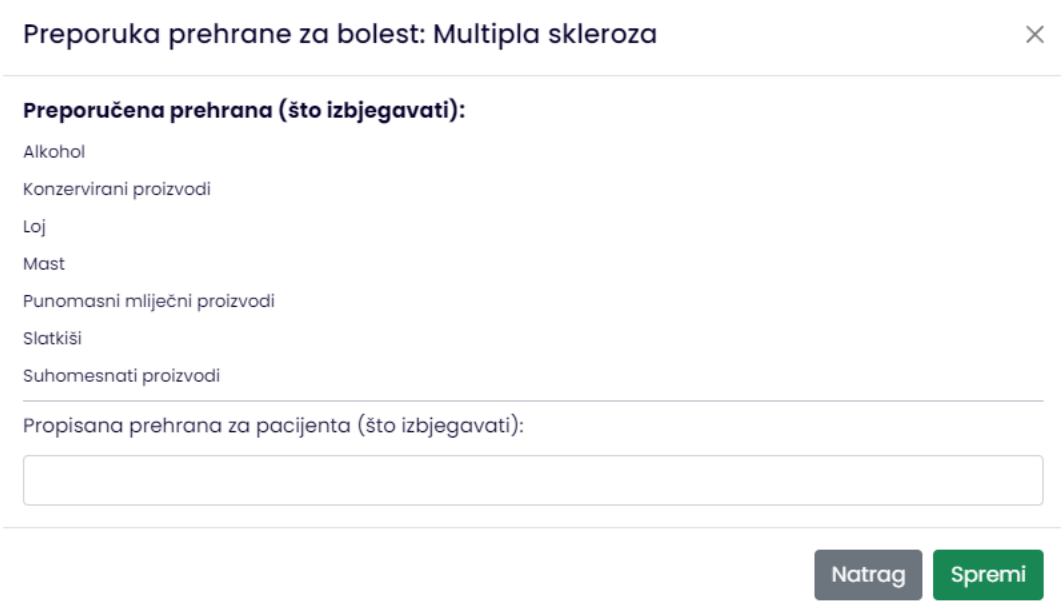
Slika 6.5. *Preporučeni lijekovi s obzirom na bolest multipla skleroza i pacijenta*

Očekivana preporuka prehrane koja se treba jesti kod multiple skleroze je slijedeća: agrumi ili citrusno voće, ananas, biljno ulje, crni kruh, dinja, fermentirani mliječni proizvodi, grožđe, jagoda, jaja, ječam, karfiol, koštunjićavo voće, kupusnjače, lisnato zeleno povrće, lubenica, luk, mahunarke, marelica, mrkva, paprika, raž, šumsko voće, svježi kravlji sir, tjestenina i voda, no budući da pacijent ima alergiju na ananas, dinju i jagode, te namirnice neće biti preporučene što je vidljivo na slici 6.6.



Slika 6.6. *Preporuka prehrane s obzirom na bolest i pacijenta*

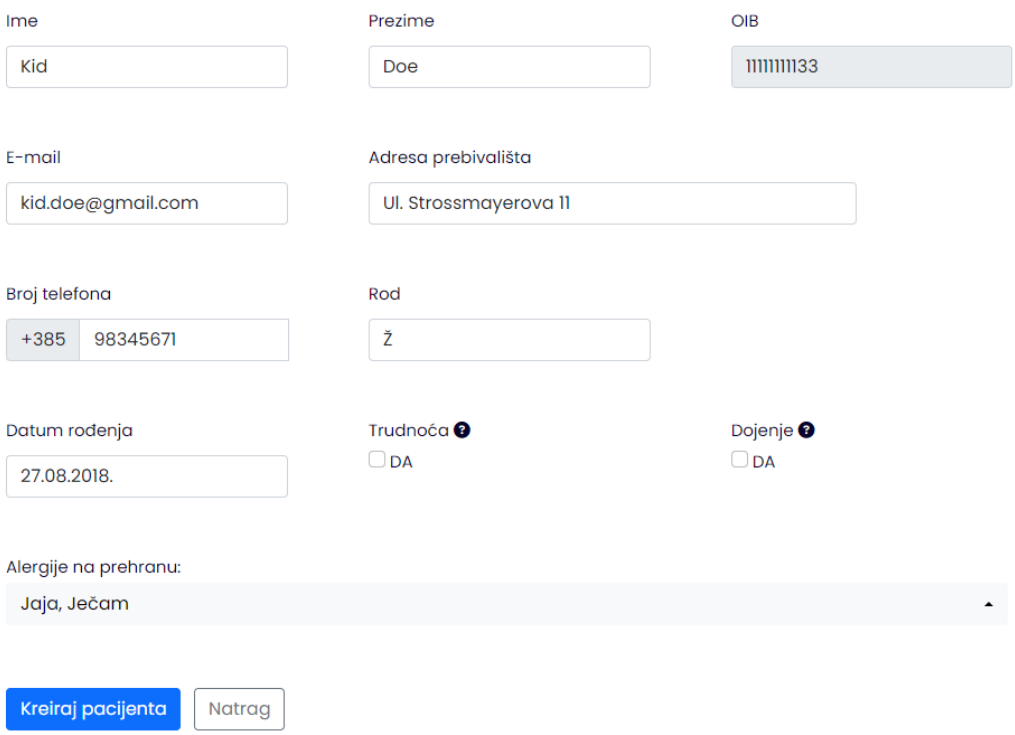
Očekivana preporuka prehrane za izbjegavati kod multiple skleroze je slijedeća: alkohol, konzervirani proizvodi, loj, mast, punomasni mliječni proizvodi, slatkiši te suhomesnati proizvodi, a na slici 6.7 je vidljivo da sustav ispravno radi i u ovome slučaju.



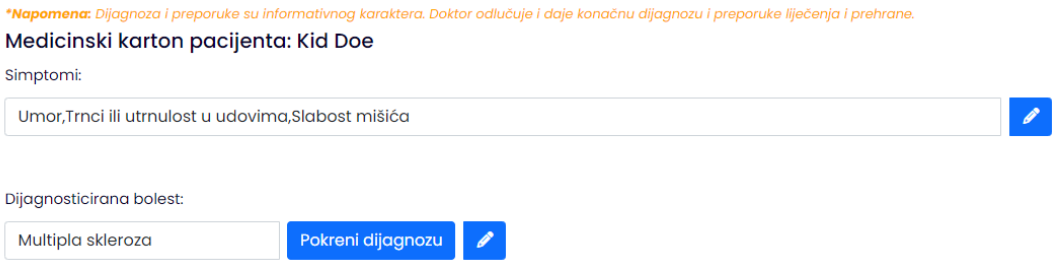
Slika 6.7. *Preporuka prehrane za izbjegavati kod bolesti multiple skleroze*

6.2.2. Primjer korištenja 2

Kod drugog primjera testiranja kreirati će se pacijent koji ima 5 godina, žensko je te ima alergije na jaja i ječam. Odabrati će se isti simptomi kao i u prethodnom primjeru te se očekuje ista dijagnoza bolesti. Budući da je ovaj pacijent dijete te nije trudno i ima različitu alergiju na prehranu, očekuju se drugačiji rezultati preporuke lijekova i prehrane. Profil pacijenta je vidljiv na slici 6.8, a na slici 6.9 je vidljiva ista dijagnoza bolesti.



Slika 6.8. *Profil pacijenta korištenog u drugom ispitivanju*

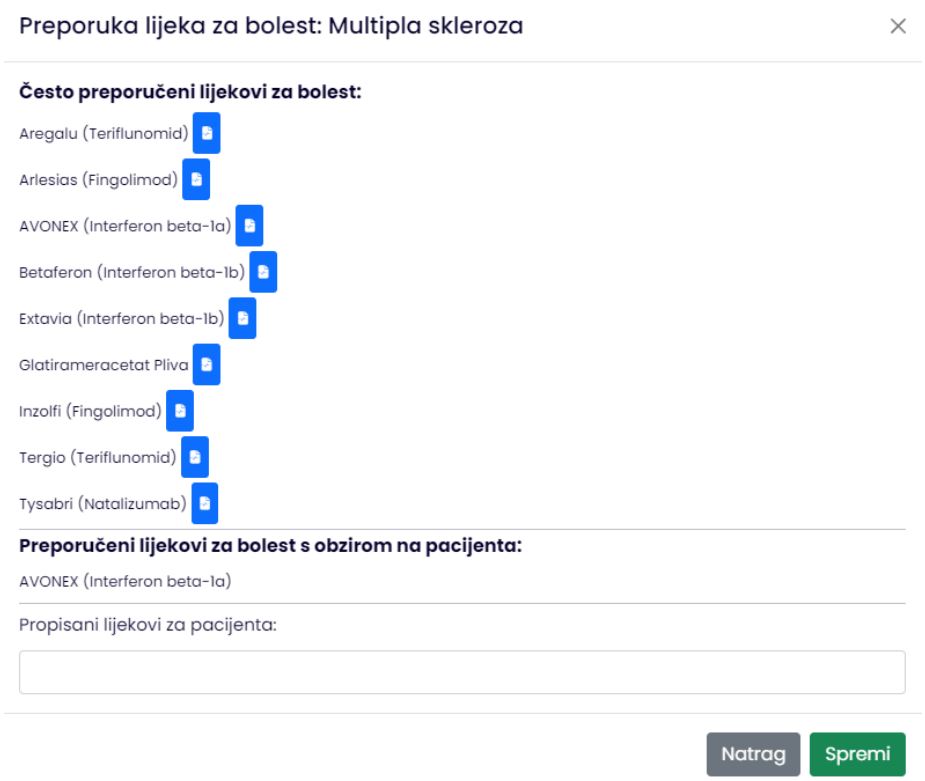


Slika 6.9. *Prikaz dijagnoze bolesti za drugo ispitivanje*

Lijekovi koje će sustav preporučiti kao često preporučene za multiplu sklerozu će biti isti kao i u prvom ispitivanju, no sustav će drugačije preporučiti lijekove s obzirom na pacijenta. U bazi, što se može vidjeti na slici 6.10, lijekovi su ograničeni s obzirom na dob te ukoliko je pacijent jednako star ili stariji od ograničenja dobi, tada se lijek preporuča. S obzirom da pacijent nije trudan niti doji, jedini uvjet je dob prilikom filtriranja lijekova za bolest multiplu sklerozu te se očekuje da će sustav preporučiti samo lijek AVONEX (Interferon beta-1a), gdje je ispravnost preporuke vidljiva na slici 6.11.

LIJEK	TRUDNOĆA(0 - S	DOJENJE(	DOB - ograničenja
Amgevita (Adalimumab)	2	0	6
Apidra (inzulin glulizin)	0	0	0
Aregalu (Teriflunomid)	1	1	10
Arlesias (Fingolimod)	1	1	10
Atenolol Pliva	0	0	14
Athyrazol (Metimazol)	2	2	3
AVONEX (Interferon beta-1a)	0	0	0
BELODERM krema (Betametazon)	0	0	0
BELODERM mast (Betametazon)	0	0	0
Benlysta (Belimumab)	2	2	0
Betaferon (Interferon beta-1b)	0	0	12
Betaloc ZOK (Metoprolol)	2	1	6
Betametazon salicilatna kiselina Belupo	2	2	0
Budosan	2	1	12
Ciklosporin Alkaloid	2	2	0
CORTEF (Hidrokortizon)	2	2	0
Decortin (Prednizon)	2	2	0
Diprogenta krema	1	2	0
Diprogenta mast	1	2	0
Enbrel (Etanercept)	2	1	0
Entyvio (Vedolizumab)	2	2	0
Euthyrox (Levotiroksin (T4 hormon))	0	0	0
Extavia (Interferon beta-1b)	2	0	12
Flixabi (Infliksimab)	2	0	6
Gastal	2	2	12
Glatirameracetat Pliva	2	2	18
HEMANGIOL (Propranolol)	1	2	0
Humalog (inzulin lispro)	0	0	0
HYPLAXY (Hidroksiklorokin)	2	1	0
Imuran (Azatioprin)	1	1	0
Inzolfi (Fingolimod)	1	1	10
Tergio (Teriflunomid)	1	1	10
Tresiba (inzulin degludek)	2	2	1
Tysabri (Natalizumab)	2	1	18

Slika 6.10. *Baza lijekova s ograničenjima za dob*



Slika 6.11. *Preporuka lijekova s obzirom na bolest i pacijenta*

Nadalje, kod preporuke prehrane koja se smije jesti kod multiple skleroze, preporučiti će se prehrana navedena u prvom ispitivanju, osim prehrambenih namirnica jaja i ječma na koje je pacijent alergičan. Preporuka prehrane je vidljiva na slici 6.12. Prehrana koja se preporuča izbjegavati je jednaka kao i u prvom slučaju jer je tu jedini uvjet za preporuku dijagnosticirana bolest koja je u prvom i drugom ispitivanju ista.

Slika 6.12. *Preporuka prehrane za pacijenta u drugom slučaju ispitivanja*

7. ZAKLJUČAK

Razvijena web aplikacija za potporu u dijagnosticiranju bolesti i generiranje preporuka za terapiju i prehranu predstavlja značajan korak prema personaliziranoj medicinskoj podršci. Kroz integraciju informacijske tehnologije i medicinske dijagnostike ova aplikacija ima potencijal značajno poboljšati pristup pacijentima i olakšati posao zdravstvenim stručnjacima. Koristeći arhitekturu Model-View-Controler (MVC), aplikacija je programski ostvarena kako bi omogućila jasno odvajanje funkcionalnosti i poboljšala održivost. Također, za razvoj korišten je Spring Boot kojim je omogućeno lakše, brže postavljanje, konfiguriranje i pokretanje aplikacije. Kako bi se što pravilnije dijagnosticirale autoimune bolesti, proučene su osobine koje upućuju na neke najčešće autoimune bolesti čime se došlo do zaključka da su prilikom dijagnoze bolesti najvažniji simptomi, zatim spol i dob pacijenta. Korišten je višekriterijski pristup preporuke lijeka i prehrane gdje se na osnovu više različitih kriterija došlo do preporučenih podataka. Tijekom razvoja, aplikacija je pažljivo testirana kako bi se osigurala njena funkcionalnost i pouzdanost. Provedeno je ručno testiranje kako bi se provjerile osnovne funkcionalnosti i interakcije unutar sustava. Uz to, izvršeno je automatsko testiranje korištenjem alata Cucumber. Ovaj sveobuhvatan pristup testiranju doprinio je ispravnom funkcioniranju i stabilnosti aplikacije.

Razvijena aplikacija predstavlja koristan alat za podršku dijagnostici i upravljanju autoimunim bolestima. Ovo web rješenje ima za cilj unaprijediti medicinsku praksu kroz personalizirane preporuke za terapiju i prehranu te dijagnozu bolesti omogućavajući potporu i pomoć zdravstvenim stručnjacima. Nadalje, budući da na mnoštvo autoimunih bolesti utječe genetika, ovo web rješenje je moguće unaprijediti praćenjem genetske predispozicije pacijenta te je moguće proširiti sustav s informacijama za dijagnozu više autoimunih bolesti, proširiti i poboljšati bazu simptoma, lijekova i prehrane. U konačnici, važno je za naglasiti da je ovaj sustav samo potpora te da ne zamjenjuje liječnike, već liječnik donosi odluku o dijagnozi bolesti i propisivanju terapije i prehrane.

LITERATURA

[1] M. P. Holsapple, N. E. Kaminski, Immune System, Encyclopedia of Toxicology (Second Edition), 2005, pp. 573-596, dostupno na: https://www.sciencedirect.com/science/article/abs/pii/B0123694000005147, pristup: [27.6.2023.]

[2] N.R. Rose, Autoimmune Diseases , International Encyclopedia of Public Health (Second Edition), 2017, pp. 192-195, dostupno na: https://www.sciencedirect.com/science/article/abs/pii/B9780128036785000291, pristup: [27.6.2023.]

[3] K. Bieber, et al., Autoimmune pre-disease, Autoimmunity Reviews, Vol. 22, Issue 2, February 2023, 103236, dostupno na: https://www.sciencedirect.com/science/article/pii/S1568997222002063, pristup: [27.6.2023.]

[4] S.M. Keestra, V. Male, G.D. Salali, Out of balance: the role of evolutionary mismatches in the sex disparity in autoimmune disease, Medical Hypotheses, Vol. 151, June 2021, 110558, dostupno na: https://www.sciencedirect.com/science/article/pii/S0306987721000761, prtistup: [27.6.2023.]

[5] S.J. Enna, D.B. Bylund, xPharm: The Comprehensive Pharmacology Reference, Elsevier, 2008, dostupno na: https://www.sciencedirect.com/referencework/9780080552323/xpharm-the-comprehensive-pharmacology-reference#book-info, pristup: [27.6.2023.]

[6] W. Hou, G. Xu, H. Wang, Treating Autoimmune Disease with Chinese Medicine, Churchill Livingstone, 2011, dostupno na: https://www.sciencedirect.com/book/9780443069741/treating-autoimmune-disease-with-chinese-medicine, pristup: [27.6.2023.]

[7] A.A. Bodemer, Chapter 73 - Psoriasis, Integrative Medicine (Fourth Edition), 2018, pp. 726-738, dostupno na: https://www.sciencedirect.com/science/article/abs/pii/B9780323358682000736, pristup: [27.6.2023.]

[8] P. Brito-Zerón, S. Retamozo, M. Ramos-Casals, Sjögren syndrome Síndrome de Sjögren, Medicina Clínica (English Edition), Vol. 160, Issue 4, 24 February 2023, pp. 163-171, dostupno na: https://www.sciencedirect.com/science/article/abs/pii/S2387020623000372, pristup: [27.6.2023.]

[9] M.K. Houtchens, S.J. Khoury, Women and Health (Second Edition), Academic Press, 2013, dostupno na: https://www.sciencedirect.com/book/9780123849786/women-and-health, pristup: [27.6.2023.]

[10] R.A. DeLellis, Y.E. Nikiforov, Diagnostic Surgical Pathology of the Head and Neck (Second Edition), Saunders, 2009, dostupno na: https://www.sciencedirect.com/book/9781416025894/diagnostic-surgical-pathology-of-the-head-and-neck, pristup: [12.8.2023.]

[11] V. Miceli, P.G. Conaldi, Diabetes (Second Edition), Oxidative Stress and Dietary Antioxidants, Academic Press, 2020, dostupno na: https://www.sciencedirect.com/book/9780128157763/diabetes, pristup: [12.8.2023.]

[12] V. Nehra, et al., Encyclopedia of Human Nutrition (Second Edition), Academic Press, 2005, dostupno na: https://www.sciencedirect.com/referencework/9780122266942/encyclopedia-of-human-nutrition, pristup: [13.8.2023.]

[13] F. Alexander, Pediatric Surgery (Sixth Edition), Mosby Hardcover, 2006, dostupno na: https://www.sciencedirect.com/book/9780323028424/pediatric-surgery, pristup: [13.8.2023.]

[14] Pliva zdravlje, Priručnik bolesti, dostupno na: https://www.plivazdravlje.hr/bolest-clanak/bolest/34/Hipertireoza.html, pristup: [13.8.2023.]

[15] AMN Healthcare, The Most Difficult to Diagnose Autoimmune Diseases, [Mrežno], dostupno na: https://www.staffcare.com/locum-tenens-blog/news/most-difficult-autoimmune-diseases-to-diagnose/, pristup: [27.6.2023.]

[16] Cleveland Clinic, Immunosuppressants, dostupno na: https://my.clevelandclinic.org/health/drugs/10418-immunosuppressants, pristup: [13.8.2023.]

[17] Todaysdietitian, The Magazine for Nutrition Professionals, dostupno na: https://www.todaysdietitian.com/newarchives/110211p36.shtml , pristup: [13.8.2023.]

[18] K.C. Cara, et al., Commonalities among Dietary Recommendations from 2010 to 2021 Clinical Practice Guidelines: A Meta-Epidemiological Study from the American College of Lifestyle Medicine, Advances in Nutrition, Vol. 14, Issue 3, May 2023, pp. 500-515, dostupno na: https://www.sciencedirect.com/science/article/pii/S2161831323002764, pristup: [13.8.2023.]

[19] Geeks for Geeks, Functional vs Non Functional Requirements, dostupno na: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/, pristup: [13.8.2023.]

[20] A. Deluka-Tibljaš, B. Karleuša, N. Dragičević, Pregled primjene metoda višekriterijske analize pri donošenju odluka o prometnoj infrastrukturi. Građevinar, 65(7), str. 619-631, 2013, dostupno na: https://hrcak.srce.hr/file/156408, pristup: [13.8.2023.]

[21] Geeks for Geeks, Introduction to Spring Boot, dostupno na: https://www.geeksforgeeks.org/introduction-to-spring-boot/?ref=lbp, pristup: [13.8.2023.]

[22] Testing Master, Manual Testing Material, dostupno na: http://testingmasters.com/wp-content/uploads/ManualTestingMaterial.pdf, pristup: [13.8.2023.]

[23] The Ultimate Guide to Automation Testing, dostupno na: https://testguild.com/wp-content/uploads/2018/07/UltimateGuildToAutomation.pdf, pristup: [13.8.2023.]

[24] W3schools, dostupno na: https://www.w3schools.com/, pristup: [13.8.2023.]

[25] AWS, What Is Java?, dostupno na: https://aws.amazon.com/what-is/java/, pristup: [13.8.2023.]

[26] K. Sagar, Programming Tools, dostupno na: https://www.linkedin.com/pulse/programming-tools-sagar-khurana, pristup: [13.8.2023.]

[27] Geeks for Geeks, Spring Boot – Thymeleaf with Example, dostupno na: https://www.geeksforgeeks.org/spring-boot-thymeleaf-with-example/, pristup: [13.8.2023.]

[28] AWS, What's The Difference Between MySQL And PostgreSQL?, dostupno na: https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/, pristup: [13.8.2023.]

[29] Callicoder, Spring Boot, PostgreSQL, JPA, Hibernate RESTful CRUD API Example, dostupno na: https://www.callicoder.com/spring-boot-jpa-hibernate-postgresql-restful-crud-api-example/ , pristup: [13.8.2023.]

[30] Baeldung, Spring Boot Tutorial – Bootstrap a Simple Application, dostupno na: https://www.baeldung.com/spring-boot-start, pristup: [13.8.2023.]

[31] Cucumber, dostupno na: https://cucumber.io/docs/cucumber/api/?lang=java#after, pristup: [13.8.2023.]

[32] MVC Framework - Introduction, dostupno na: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm, pristup: [13.8.2023.]

[33] B. Garcia, F. Ricca, J.M. del Alamo, M. Leotta, Enhancing Web Applications Observability through Instrumented Automated Browsers, Journal of Systems and Software, Volume 203, September 2023, 111723, dostupno na: https://www.sciencedirect.com/science/article/pii/S0164121223001188, pristup: [18.8.2023.]

[34] Aila Health, Google Play, Aila Health, dostupno na: https://play.google.com/store/apps/details?id=com.ailahealth&hl=bs&gl=US, pristup: [2.9.2023.]

[35] Mymee, Google Play, Mymee Inc., dostupno na: https://play.google.com/store/apps/details?id=com.mymee.app&hl=en_US, pristup: [2.9.2023.]

[36] Grand View Research, mHealth Market Size, Share & Trends Analysis Report By Component, By Services (Monitoring Services, Diagnosis Services), By Participants (Mobile Operators, Devices Vendors), By Region, And Segment Forecasts, 2023 - 2030, dostupno na: https://www.grandviewresearch.com/industry-analysis/mhealth-market, pristup: [2.9.2023.]

[37] K. Fan, Y. Zhao, Mobile health technology: a novel tool in chronic disease management, Intelligent Medicine, Vol. 2, Issue 1, February 2022, pp. 41-47 dostupno na: https://www.sciencedirect.com/science/article/pii/S2667102621000346, pristup: [2.9.2023.]

[38] J. Peng , E.C. Jury, P. Dönnes, C. Ciurtin, Machine Learning Techniques for Personalised Medicine Approaches in Immune-Mediated Chronic Inflammatory Diseases: Applications and Challenges, 2021, dostupno na: https://www.scienceopen.com/document_file/0b55bbb0-20db-4d98-a283-8a5ca6cb6597/PubMedCentral/0b55bbb0-20db-4d98-a283-8a5ca6cb6597.pdf, pristup: [2.9.2023.]

SAŽETAK

U ovom diplomskom radu ostvaren je web sustav koji omogućuje potporu u dijagnosticiranju najčešćih autoimunih bolesti te preporuku prikladnih lijekova i prehrane s obzirom na osobine pacijenta i njegove dijagnosticirane bolesti. Opisani su izazovi i pristupi korišteni prilikom dijagnoze autoimunih bolesti te propisivanja lijekova i prehrane. Korištenjem višekriterijskog filtriranja omogućeno je stvaranje točnijih preporuka prema većem broju pacijentovih osobina. Aplikacija se temelji na arhitekturi Model-View-Controler, a razvijena je u programskom okviru Spring Boot. Za dinamički prikaz HTML-a koristi se Thymeleaf, za oblikovanje sučelja CSS i Bootstrap, a Javu za implementiranje logike aplikacije. Podaci se pohranjuju u PostgreSQL bazu podataka. Također, korištena su razvojna okruženja Visual Studio Code, IntelliJ IDEA, DBeaver i GitHub. Testiranje ostvarenog sustava obavljeno je na razini korisničkog sučelja putem ručnog i automatiziranog testiranja. Automatizirano testiranje je ostvareno korištenjem Ruby programskog jezika i korištenjem Cucumber alata. Testiranjem je potvrđeno ispunjenje funkcionalnih i nefunkcionalnih zahtjeva na sustav.

Ključne riječi: autoimune bolesti, stvaranje preporuka, testiranje, višekriterijski postupak, web sustav.

ABSTRACT

Development and Testing of a Web Application with a Recommendation System to Support Diagnosis, Treatment and Diet in Autoimmune Diseases

In this master's thesis, a web system has been developed to provide support for diagnosing the most common autoimmune diseases and recommending appropriate medications and diets based on the patient's characteristics and diagnosed diseases. The challenges and approaches used in the diagnosis of autoimmune diseases and the prescription of medications and diets are described. By using multicriteria filtering, more accurate recommendations can be generated based on a larger number of patient characteristics. The application is based on the Model-View-Controller architecture and was developed using the Spring Boot framework. Thymeleaf is used for dynamic HTML rendering, CSS and Bootstrap for interface styling, and Java for implementing application logic. Data is stored in a PostgreSQL database. Additionally, development environments such as Visual Studio Code, IntelliJ IDEA, DBeaver, and GitHub were utilized. Testing of the implemented system was carried out at the user interface level through manual and automated testing. Automated testing was achieved using the Ruby programming language and the Cucumber tool. Testing confirmed the fulfillment of both functional and non-functional requirements of the system.

Keywords: autoimmune diseases, recommendation generation, testing, multicriteria approach, web system.

ŽIVOTOPIS

Mihaela Mandić rođena je 29. rujna 1999. godine u Osijeku, Hrvatska. Osnovnu školu pohađala je u Donjem Miholjcu u Osnovnoj školi "August Harambašić". Nakon toga je upisala opću gimnaziju u Srednjoj školi Donji Miholjac koju je završila 2018. godine. Upisala je Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Računarstvo, gdje je završila preddiplomski studij. Akademske godine 2020./2021. upisala je prvu godinu diplomskog sveučilišnog studija računarstva, smjer Programsko inženjerstvo. Krajem 2020. godine započela je raditi kao *tester* u tvrtki Ericsson Nikola Tesla d.d.

PRILOZI

Prilog 1. Diplomski rad u DOCX obliku

Prilog 2. Diplomski rad u PDF obliku

Prilog 3. Programski kod web aplikacije za dijagnozu autoimunih bolesti i preporuku prikladnih lijekova i prehrane