Izrada i funkcionalno testiranje web aplikacije za praćenje i potporu pacijentima oboljelima od dijabetesa

Filipović, Ena

SVEUČILIŠTE JOSIPA JURJA STORSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij

IZRADA I FUNKCIONALNO TESTIRANJE WEB APLIKACIJE ZA PRAĆENJE I POTPORU PACIJENTIMA OBOLJELIMA OD DIJABETESA

Diplomski rad

Ena Filipović

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 16.09.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Ena Filipović
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina	D-980R, 24.09.2019.
OIB studenta:	83131224191
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Prof.dr.sc. Željko Hocenski
Član Povjerenstva 1:	Prof.dr.sc. Goran Martinović
Član Povjerenstva 2:	izv. prof. dr.sc. Josip Job
Naslov diplomskog rada:	Izrada i funkcionalno testiranje web aplikacije za praćenje i potporu pacijentima oboljelima od dijabetesa
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U diplomskom radu treba analizirati i opisati probleme pacijenata s rizikom obolijevanja i oboljelih od dijabetesa, te utjecaj fizičke aktivnosti na stanje pacijenta i dati osvrt na postojeća web i mobilna rješenja koja omogućuju praćenje stanja i potporu oboljelima. Na temelju navedenoga treba predložiti model i arhitekturu vlastite web aplikacije koja će omogućiti unos i pohranu podataka o pacijentu, njegovom zdravstvenom stanju, te laboratorijske podatke i podatke o fizičkoj aktivnosti s narukvice za prikupljanje podataka. Web aplikacija treba omogućiti analizu navedenih podataka na temelju modela učenja, a pacijentu i liječniku omogućiti njihov prikaz, te stvaranje preporuka i obavijesti vezanih za zdravstveno stanje pacijenta. Nadalje, u radu treba opisati odgovarajuće metodologije funkcionalnog testiranja web aplikacije, odnosno testiranja API-ja, korisničkog sučelja, te integracijskog testiranja. Također, treba izabrati prikladnu metodologiju, okolinu i alat za testiranje. Navedene testove treba provesti za odgovarajuće scenarije testiranja i odgovarajuće skupove podataka, a na temelju rezultata testiranja treba po potrebi unaprijediti programsko rješenje web aplikacije. Tema rezervirana za: Ena Filipović

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 mentora:	16.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 24.09.2020.

Ime i prezime studenta:	Ena Filipović
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-980R, 24.09.2019.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada i funkcionalno testiranje web aplikacije za praćenje i potporu pacijentima oboljelima od dijabetesa**

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 1
1.1. Zadatak diplomskog rada 1
2. PREGLED STANJA U PODRUČJU 3
2.1. Računalne tehnologije i dijabetes 3
2.2. Pregled postojećih rješenja 3
2.3. Pregled stanja u području funkcionalnog testiranja 5
3. DIJABETES 6
3.1. Metabolizam glukoze 6
3.2. Tipovi dijabetesa 6
3.2.1. Dijabetes tipa I 7
3.2.2. Dijabetes tipa II 7
3.3. Simptomi dijabetesa 8
3.4 Komplikacije dijabetesa 9
3.4.1. Akutne komplikacije dijabetesa 9
3.4.2. Kronične komplikacije dijabetesa 10
3.4.3. Smrtnost i broj oboljelih od dijabetesa 13
3.5. Prevencija dijabetesa 14
3.5.1. Čimbenici rizika 14
3.5.2. Postupci prevencije dijabetesa 15
3.6. Utjecaj fizičke aktivnosti na kvalitetu života oboljelih 16
3.6.1. Dijabetes tipa I i fizička aktivnost 16
3.6.2. Dijabetes tipa II i fizička aktivnost 17
4. ARHITEKTURA APLIKACIJE 18
4.1. Zahtjevi na aplikaciju 18
4.2. Idejno rješenje web aplikacije 19
4.3. Struktura aplikacije 20
4.4. Analiza zdravstvenog stanja i rizika obolijevanja od dijabetesa 22
5. PROGRAMSKO RJEŠENJE WEB APLIKACIJE 25
5.1. Korištene tehnologije 25
5.1.1. HTML 25
5.1.2. CSS 25
5.1.3. Razvojni okvir React.js 25
5.1.4. Platforma Node.js 26

5.1.5. Baza podataka MongoDb 26
5.1.6. Programski okvir Express.js 27
5.2. Programsko rješenje aplikacije 27
5.2.1. Baza podataka 27
5.2.2. Registriranje korisnika 33
5.2.3. Prijava i odjava korisnika 34
5.2.4. Prikaz vijesti o dijabetesu 35
5.2.5. Izračun rizika od dijabetesa 37
5.2.6. Pregled pacijenata 39
5.2.7. Unos nalaza 39
5.2.8. Prikaz i odabir liječnika 41
5.2.9. Pregled nalaza i zdravstvenog stanja 42
5.2.10. Unos podataka s pametne narukvice i izračun zdravstvenog stanja 43
6. NAČIN RADA APLIKACIJE 44
6.1. Postupak registriranja korisnika 44
6.2. Postupak prijave korisnika 45
6.3. Sučelje za pregled vijesti 46
6.4. Sučelje za izračun rizika 46
6.5. Sučelje za odabir liječnika 48
6.6. Sučelje za pregled pacijenata i unos nalaza 48
6.7. Sučelje za pregled nalaza 49
7. TESTIRANJE APLIKACIJE 52
7.1. Vrste testiranja 53
7.1.1. Ručno testiranje 53
7.1.2. Automatizirano testiranje 53
7.2. Korištene tehnologije za testiranje programske podrške 53
7.2.1. Cypress 54
7.2.2. Postman 55
7.3. Testni slučajevi 55
7.4. Provođenje automatiziranog testiranja 58
7.4.1 Instalacija alata Cypress 58
7.4.2 Pisanje testova u alatu Cypress 58
7.4.3 Rezultati automatiziranog testiranja 60
7.5. Provođenje testiranja API-ja 62

7.6. Provođenje ručnog testiranja 63
7.6.1. Rezultati ručnog testiranja 65
8. ZAKLJUČAK 67
LITERATURA 68
SAŽETAK 71
ABSTRACT 72
ŽIVOTOPIS 73
PRILOZI 74

1. UVOD

Dijabetes je bolest koja je u zadnje vrijeme postala toliko raširena da se naziva i epidemijom modernog doba. Povezuje se s mnogim bolestima koje smanjuju kvalitetu i zadovoljstvo u ljudskom životu te skraćuju životni vijek. Osim samog zdravlja populacije, dijabetes djeluje negativno i na svjetsku ekonomiju radi porasta broja oboljelih i sve većih troškova na zdravstveni sustav. Iako je jedan oblik te bolesti uzrokovan genetikom i promjena životnih navika ne može utjecati preventivno, taj oblik čini samo 10% slučajeva dijabetesa. Drugi tip dijabetesa veoma lako se prevenira regulacijom prehrane i povećanjem tjelesne aktivnosti, a to, osim na dijabetes, djeluje i na širu paletu zdravstvenih stanja i općenito poboljšava svaku sferu života. Računalne i informacijske tehnologije mogu značajno pomoći u educiranju populacije o problemu dijabetesa te pružiti potporu oboljelima.

Zadatak ovog diplomskog rada je izraditi web aplikaciju koja će služiti kao potpora oboljelima od dijabetesa. Aplikacija će svakoj osobi računati rizik od dijabetesa te na temelju nalaza koje unese liječnik i podataka s pametne narukvice koje unese pacijent, dati pacijentu informacije koje će služiti općenito lakšem praćenju zdravstvenog stanja oboljelih i onih u rizičnim skupinama. Osim same izrade, aplikacija će biti i testirana koristeći alate za automatizirano testiranje te će se na njegom primjeri prikazati funkcionalno testiranje web aplikacija.

U poglavlju 2 predstavljen je pregled uloge računalnih znanosti u području dijabetesa te primjeri web rješenja koja pružaju potporu oboljelima. Osim toga, opisano je i stanje u području testiranja web aplikacija kao znanstvenoj grani čija vrijednost ubrzano raste. Poglavlje 3 sadržava teorijsku podlogu o dijabetesu koja služi za lakše razumijevanje problema koji aplikacija pokušava riješiti. Opisani su njegovi simptomi, načini prevencije i utjecaj te bolesti na ljudsko zdravlje. Zahtjevi na aplikaciju definirani su u poglavlju 4 uz arhitekturu i model aplikacije. Korištene tehnologije koje su odabrane za razvoj rješenja opisane su u poglavlju 5. U istom poglavlju opisan je i razvijeni programski kod te način njegova rada. U poglavlju 6 predstavljen je rad aplikacije s korisničke perspektive. Posljednje poglavlje sadržava opis provedenih postupaka testiranja te rezultate testiranja.

1.1. Zadatak diplomskog rada

U diplomskom radu treba analizirati i opisati probleme pacijenata s rizikom obolijevanja i oboljelih od dijabetesa, te utjecaj fizičke aktivnosti na stanje pacijenta i dati osvrt na postojeća

web i mobilna rješenja koja omogućuju praćenje stanja i potporu oboljelima. Na temelju navedenoga treba predložiti model i arhitekturu vlastite web aplikacije koja će omogućiti unos i pohranu podataka o pacijentu, njegovom zdravstvenom stanju, te laboratorijske podatke i podatke o fizičkoj aktivnosti s narukvice za prikupljanje podataka. Web aplikacija treba omogućiti analizu navedenih podataka na temelju modela učenja, a pacijentu i liječniku omogućiti njihov prikaz, te stvaranje preporuka i obavijesti vezanih za zdravstveno stanje pacijenta. Nadalje, u radu treba opisati odgovarajuće metodologije funkcionalnog testiranja web aplikacije, odnosno testiranja API-ja, korisničkog sučelja, te integracijskog testiranja. Također, treba izabrati prikladnu metodologiju, okolinu i alat za testiranje. Navedene testove treba provesti za odgovarajuće scenarije testiranja i odgovarajuće skupove podataka, a na temelju rezultata testiranja treba po potrebi unaprijediti programsko rješenje web aplikacije.

2. PREGLED STANJA U PODRUČJU

Kako broj oboljelih od dijabetesa užurbano raste, tako su se radi povećane potrebe razvila mnoga ICT i ostala rješenja kako bi educirala pacijente o važnosti prevencije bolesti i načinima kako ju mogu inkorporirati u svoj život.

2.1. Računalne tehnologije i dijabetes

Potražnja i zastupljenost računalnih znanosti u medicini se naglo povećava. Istraživanje provedeno od studenog 2017. do ožujka 2018. godine provelo je internetsku anketu namijenjenu osobama starijim od 18 godina koje boluju od dijabetesa [1]. Tema ankete bila je zastupljenost mobilnih aplikacija koje na neki način pomažu bolesnicima s dijabetesom. Anketu su riješile 1552 osobe koje boluju od dijabetesa tipa 1 i 630 osoba koje boluju od dijabetesa tipa 2. Rezultati su pokazali da više od polovice oboljelih koristi aplikacije za praćenje svoje bolesti te da aplikacije mogu prouzročiti promjene u navikama i mjerenju razine šećera u krvi bolesnika.

Programska rješenja variraju od mobilnih aplikacija, raznih informativnih portala za edukaciju bolesnika o dijabetesu do kalkulatora rizika i korištenja postupaka umjetne inteligencije u dijagnostici i edukaciji. Istraživanje o primjeni umjetne inteligencije u svrhu edukacije o dijabetesu pokazalo je da iako se računalna tehnologija oko umjetne inteligencije veoma brzo razvila, potrebno je još vremena kako bi rezultati usavršili [2]. Ustanovljeno je da je za optimalan rezultat potrebna velika količina ispravnih podataka koja se konstantno unaprjeđuje novim saznanjima te uključivanje pacijenta u dizajn sustava radi što bolje učinkovitosti. Unatoč tome, zaključeno je da bi u budućnosti umjetna inteligencija mogla omogućiti oboljelima veoma snažan sustav potpore kroz usmjeravanje i edukaciju. Iz razloga da razvijena rješenja budu što korisnija oboljelima, razvojni inženjeri sve češće uključuju oboljele u planiranje sustava. Istraživanje provedeno 2007. godine pokušalo je pronaći optimalan dizajn aplikacije za oboljele od dijabetesa [3]. Provodilo se nad 6 odraslih osoba s tipom 1 dijabetesa i 3 osobe s tipom 2 dijabetesa koje su dobile različite inačice aplikacija za potporu oboljelim od dijabetesa te su ocjenjivali svoje zadovoljstvo funkcionalnostima. Rezultati su pokazali da programska rješenja za svakodnevnu podršku oboljelima trebaju sadržavati sučelja za svakodnevni unos glukoze u krvi te podatke o fizičkoj aktivnosti.

2.2. Pregled postojećih rješenja

Najveći broj ICT rješenja koja se bave problemom dijabetesa su razne mobilne aplikacije za vježbanje, praćenje kalorija, nadziranja nalaza i sl. Neke od njih su:

- GDm Health

GDm Health je aplikacija koja služi za pomoć liječnicima kako bi nadzirali dijabetes kod trudnica. Aplikacija i pripadajuća oprema koju pacijenti imaju pomaže u bežičnom praćenju razine glukoze u krvi te kreiranju izvještaja za korištenje u medicinske svrhe za pravovremenu reakciju u slučaju hipoglikemije, hiperglikemije i ostalih stanja koje dijabetes može prouzrokovati.

- mySugar

mySugar je aplikacija za svakodnevni unos izmjerenih razina glukoze u krvi. Pomaže za lakše praćenje napredovanja i kontrole bolesti kod oboljelih od dijabetesa. Osim praćenja glukoze u krvi, u aplikaciju se može unijeti i količina konzumiranih ugljikohidrata i procjena Hba1c razine.

- Low Carb Program

Low Carb Program je aplikacija koja služi u edukativne svrhe kod bolesnika s dijabetesom. Orijentirana je na pacijente koji boluju od dijabetesa tipa II ili su u pred dijabetičnom stanju. Nudi razne recepte i savjete kako bi oboljeli lakše kontrolirali razine glukoze i kolesterola u krvi.

Osim mobilnih aplikacija, postoje državne i druge edukativne web stranice i stranice potpore za oboljele. Najpoznatije od njih su:

- Diabetes.org

Diabetes.org je web stranica kreirana od strane američke zajednice za dijabetes. Nudi razne informacije općenito o samoj bolesti i o svakom tipu bolesti. Osim toga, oboljeli tamo mogu potražiti i psihološku pomoć u obliku članaka i linkova za kontakt s udrugama u blizini.

- Diabetes.co.uk

Diabetes.co.uk britanska je web stranica s više od 600 000 članova u zajednici. Povezuje oboljele od dijabetesa te pokušava biti zajednica u kojoj članovi dijele svoja iskustva i time pomažu drugima da se lakše nose s bolešću.

2.3. Pregled stanja u području funkcionalnog testiranja

Osim samog razvoja programskih rješenja, u zadnjem desetljeću počelo se sve više stavljati naglasak na kvalitetu razvijenog sustava. U siječnju 2019. godine napisan je znanstveni članak o važnosti testiranja programske podrške u razvoju sustava. Zaključeno je da je testiranje vrlo važna grana razvoja programske podrške ukoliko se izvršava na pravi način [4]. Potrebno je ne samo izabrati odgovarajuće tehnologije, metodologije i alate, nego i imati jasan cilj što se testiranjem želi postići. Uvođenjem raznih pomagala za oboljele od dijabetesa – inzulinske pumpe, trakice za mjerenje razine glukoze u krvi – pokazala se potreba za testiranjem i u području dijabetesa i medicine. Istraživanje o tehnikama za testiranje medicinske programske podrške pokazalo je da je potreban velik broj različitih ispitanika i korištenih tehnika uz povratnu informaciju korisnika sustava da se sustav za potporu oboljelima od dijabetesa unaprijedi [5].

3. DIJABETES

Dijabetes (lat. *diabetes mellitus*) je kronična bolest vezana uz nepravilno metaboliziranje šećera u organizmu što dovodi do povećane koncentracije glukoze u krvi. Nastaje zbog smanjenog lučenja ili smanjenog djelovanja hormona inzulina koji luči gušterača.

3.1. Metabolizam glukoze

Pojam metabolizam odnosi se na kemijske reakcije i procese koji se odvijaju unutar ljudskog tijela te su nužne za pravilno funkcioniranje organizma. Proces metaboliziranja šećera u ljudskom tijelu gotovo je identičan u zdravih ljudi i onih s dijabetesom, jedina razlika je u djelotvornosti i/ili količini hormona inzulina koji je izlučen u tom procesu [6]. Nakon što čovjek konzumira hranu ljudsko tijelo koristi slinu izlučenu u usnoj šupljini i želučanu kiselinu u želucu kako bi rastvorilo ugljikohidrate. Ugljikohidrati se razlažu na monomere (monosaharide) – glukozu, fruktozu i galaktozu. Glukoza čini 80 posto tvari dobivenih od razlaganja ugljikohidrata te je glavni izvor hrane stanicama organizma. U stanicama se glukoza rastvara u energiju ili pohranjuje kao zaliha energije u obliku polisaharida glikogena. Nakon prerade ugljikohidrata u glukozu, glukoza ulazi u krvožilni sustav. Beta stanice gušterače prepoznaju da se razina šećera podigla u krvi te luče hormon inzulin u krv, ovakvo lučenje inzulina je tzv. prva faza lučenja. Gušterača luči inzulin u dvije faze:

- **1. faza –** lučenje inzulina kao brzi odgovor tijela na povećanje glukoze u krvi
- ***2.*** **faza –** događa se nakon prve faze te se manifestira kroz usporeno lučenje inzulina koje služi kao spremnik hormona na buduće povećanje glukoze u krvi [7]

Nakon što gušterača izluči odgovarajuću količinu inzulina, inzulin omogućava glukozi da uđe u stanice te se tamo koristi kao gorivo ili da se pohrani u mišićima i jetri kao zaliha energije – glikogen. Ako stanicama nije dovoljna količina energije koju su dobile od glukoze, luči se hormon glukagon koji spremljeni glikogen pretvara u glukozu te se krvlju dovodi do odgovarajućih stanica. Ponekad nakon ova oba procesa iskorištavanja glukoze ostane višak glukoze u krvi. Tada tijelo pretvara taj višak šećera u mast. Kod osoba koje boluju od dijabetesa, metabolički problemi se razlikuju ovisno o tipu dijabetesa od kojega pate.

3.2. Tipovi dijabetesa

Dijabetes se dijeli na dva glavna tipa:

- Dijabetes tipa I

- Dijabetes tipa II

3.2.1. Dijabetes tipa I

Dijabetes tipa I poznat je i kao juvenilni dijabetes, jer se najčešće manifestira u dobi manjoj od 30 godina. Osim toga, poznat je i kao dijabetes ovisan o inzulinu pošto u ovom obliku dijabetesa gušterača ne luči dovoljnu količinu hormona inzulina. Ovakva vrsta dijabetesa nastaje radi autoimunog uništavanja beta stanica gušterače koje su odgovorne za lučenje inzulina. Razaranje beta stanica može trajati mjesecima, pa čak i godinama prije dijagnoze. Najčešće se dijagnosticira tek u trenutku kada je količina beta stanica toliko razorena da gušterača više ne luči dovoljnu količinu inzulina za nadzor glukoze. Pacijenti koji boluju od dijabetesa tipa I moraju ubrizgavati točno odmjerene količine inzulina nakon obroka. Ubrizgani inzulin služi kako bi oponašao prvu fazu mehanizma gušterače za lučenje inzulina [8]. Zahtjevno je dozirati ispravnu količinu inzulina kako bi metabolizam glukoze bio sličan onome kod zdravog čovjeka, iz tog razloga sve više pacijenata se okreće korištenju pumpe za ubrizgavanje koja kontinuirano isporučuje inzulin u tijelo kroz plastičnu cjevčicu kanilom povezanu s ljudskom kožom.

Najčešći uzrok dijabetesa tipa I je genetika, iako je znanstveno dokazano da izloženost nekim virusima te okoliš u kojemu je oboljeli prebivao također mogu utjecati na pojavnost ove bolesti [9]. Neizlječiva je autoimuna bolest, stoga oboljeli samo pravilnim korištenjem inzulina i redovitim kontrolama mogu držati svoje zdravlje pod kontrolom. Dijabetes tipa I čini 10 % slučajeva dijabetesa.

3.2.2. Dijabetes tipa II

Dijabetes tipa II poznat je i kao adultni dijabetes, jer se najčešće manifestira u dobi većoj od 30 godina [6]. Osim toga, poznat je i kao dijabetes neovisan o inzulinu pošto u ovom obliku dijabetesa gušterača luči hormon inzulin, ali njegova učinkovitost je mala. Najčešći je oblik šećerne bolesti, pošto mu je glavni uzročnik neuredan način života i pretilost koji rastu u trenutnoj svakodnevici. Javlja se u 90 % slučajeva dijabetesa. Ovaj oblik bolesti manifestira se lučenjem normalne ili prevelike količine inzulina, ali poremećenim metabolizmom istog pošto uslijed promjena u stanicama jetre i gušterače organizam postaje otporan na inzulin. Promjene u stanicama jetre i gušterače događaju se radi prevelikih količina inzulina koje je potrebno preraditi, postaju preopterećene, te se s vremenom istroše do te mjere da ga više ne mogu metabolizirati. Inzulin se kao i u normalnom metabolizmu, veže uz stanice, ali glukoza ne može ući u njih kako bi se pretvorila u energiju ili glikogen. Ovo stanje je također poznato i kao inzulinska rezistencija. Gušterača radi inzulinske rezistencije ispusti sav inzulin koji sadrži, ali njegova količina ne bude dovoljna jer je manje djelotvoran te se tijelo oslanja na inzulin izlučen

u drugoj fazi lučenja koji se usporeno stvara i nije dovoljan da smanji količinu glukoze u krvi. Pacijentima koji boluju od dijabetesa tipa II savjetuju se dijeta i tjelovježba, a ako to ne bude dovoljno za održavajuće stanje glikemije, preporučuju se perioralni lijekovi te u najozbiljnijim slučajevima ubrizgavanje inzulina kao i pacijentima dijabetesa tipa I.

Glavna obilježja i razlike između dijabetesa tipa I i dijabetesa tipa II prikazani su u tablici 3.1.

Tablica 3.1 Glavna obilježja tipova dijabetesa

GLAVNA OBILJEŽJA		
	Tip I	**Tip II**
Dob pri nastupu bolesti	< 30 godina	> 30 godina
Pretilost	Ne	Često
Lučenje inzulina	Ne	Da
Uzrok	Genetika, faktori okoliša	Nezdrav način života, pretilost
Liječenje	Ubrizgavanje odgovarajućih doza inzulina	Promjena navika i načina života, dijeta, u ozbiljnijim slučajevima inzulin
Pojavnost	10 %	90 %

3.3. Simptomi dijabetesa

Simptomi dijabetesa su često prikriveni i rijetko prepoznati pošto, posebice kod tipa II, bolest napreduje veoma polako. Najčešće je dijagnosticirana nakon nekoliko godina blagih simptoma, kada je već uzela maha u organizmu. No, rano otkrivanje bolesti prateći simptome pokazalo se kao jedan od najboljih načina za držati dijabetes pod kontrolom. Najčešći simptomi dijabetesa su:

- Poliurija

Poliurija je najčešće simptom dijabetesa tipa I. Karakterizira ju često mokrenje, posebice noću. Poliurija se dijagnosticira kada bolesnik izlučuje više od 3 litre mokraće na dan te se razlikuje od čestog mokrenja kod zdravih ljudi [10].

- Polidipsija

Polidipsija se najčešće javlja kod bolesnika dijabetesa tipa I. Karakterizira ju pojačan osjećaj žeđi, koju ne utažuje ni velika količina popijene tekućine. Osim žeđi, bolesnici pate i od suhoće usta.

- Polifagija

Polifagija opisuje neutaživu glad ili pojačan apetit koje pacijenti koji boluju od dijabetesa mogu osjećati.

- Gubitak težine

Gubitak težine simptom je dijabetesa tipa I ili dijabetesa tipa II koji dug period vremena nije bio dijagnosticiran i liječen.

- Umor i iscrpljenost
- Zamućen vid
- Trnci u stopalima
- Sporo cijeljenje rana i infekcije

Liječnik može dijagnosticirati oboljenje od dijabetesa na ako je mjerena razina glukoze u krvi oboljelog u dva navrata veća od 7,7 mmol/L ili ako je u bilo koje vrijeme mjerenja razina glukoze veća od 11,1 mmol/L.

3.4 Komplikacije dijabetesa

Komplikacije dijabetesa mogu trajno narušiti zdravlje i kvalitetu života oboljelih. Bolesti koje izazivaju vežu se uz mnoge dijelove ljudskog organizma i mogu završiti smrtnim ishodom. Komplikacije dijabetesa dijele se na:

1. Akutne (kratkotrajne) komplikacije
2. Kronične (dugotrajne) komplikacije

3.4.1. Akutne komplikacije dijabetesa

Akutne (kratkoročne) komplikacije dijabetesa nastaju veoma brzo te izostankom pravovremenog tretmana mogu rezultirati smrću pacijenta. Najčešće akutne komplikacije dijabetesa su:

1. Hipoglikemija

Hipoglikemija je stanje koje karakterizira niska razina glukoze u krvi [11]. Najčešće se pojavljuje kod bolesnika koji primaju inzulinsku terapiju. Opći simptomi hipoglikemije, tzv. andrenergički simptomi počinju se pojavljivati pri razinama glukoze manjim od 3,3 mmol/L te uključuju znojenje, tahikardiju (ubrzan rad srca), bljedilo te anksioznost [12]. Moždani simptomi hipoglikemije, tzv. neuroglukopenijski simptomi počinju se pojavljivati pri razinama glukoze manjim od 2,8 mmol/L kao posljedica poremećenog rada živčanog sustava te uključuju glavobolju, vrtoglavicu, zbunjenost, a u težim slučajevima napadaje i komu. Hipoglikemije se liječi konzumacijom 10-20 grama glukoze u slučaju lakše hipoglikemije ili većim količinama glukoze intravenski u slučaju hipoglikemije koja izaziva simptome u živčanom sustavu.

2. Dijabetička ketoacidoza

Dijabetička ketoacidoza je medicinsko stanje koje se pojavljuje radi manjka inzulina. Događa se u slučaju kada je postojao problem s doziranjem inzulina, kvara opreme za inzulinsku terapiju te raznih infekcija i trauma. Isto se pojavljuje i kod ljudi kojima još nije dijagnosticiran dijabetes tipa I, kao jedan od prvih znakova te bolesti. Radi manjka inzulina, organizam počinje pojačano razgrađivati naslage masti koje se metabolizmom transformiraju u ketokiseline. Ketokiseline se metaboliziraju u velikoj količini te se talože u plazmi i čine organizam kiselim. Simptomi dijabetičke ketoacidoze su karakteristični te se razvijaju postupno kroz vrijeme. Započinju umorom, učestalim lučenjem mokraće i žeđi te se nakon toga razvijaju u mučninu, povraćanje tahikardiju te nerijetko dovode do poremećaja svijesti. Oboljeli od dijabetičke ketoacidoze često imaju zadah koji miriše na aceton. Kako bi se ketoacidoza izliječila, bolesnika je potrebno hidrirati te mu smanjiti razinu glukoze u krvi.

3. Hiperglikemijsko hiperosmolarno stanje

Hiperglikemijsko hiperosmolarno stanje karakteristično je za pacijente starije životne dobi koji imaju dijabetes tipa II. Ponekad je potaknuto iznimnim stresom. Obilježava ga povećana razina glukoze u krvi (hiperglikemija), manjak elektrolita i dehidracija, hiperosmolarnost plazme te poremećaju svijesti. Nastaje nakon duge netretirane hiperglikemije i nedovoljnog unosa tekućine koji rezultira dehidracijom. Liječenje se provodi na sličan način kao i kod dijabetičke ketoacidoze uz dodatak rehidracije pacijenta otopinom natrijevog klorida.

3.4.2. Kronične komplikacije dijabetesa

Kronične komplikacije dijabetesa događaju se radi dugogodišnjeg zanemarivanja dijabetesa, tj. radi konstantne povišene razine glukoze u krvi [13]. Kronične komplikacije najviše pogađaju krvožilni sustav čovjeka zahvaćajući mikrovaskularne i makrovaskularne žile te direktno uzrokuju druge zdravstvene probleme. Mikrovaskularne promjene djeluju na razvijanje tri najčešće degenerativne komplikacije dijabetesa:

1. Dijabetična retinopatija

Dijabetična retinopatija događa se kod dijabetičara koji boluju od dijabetesa nekoliko godina, karakteriziraju ju mikroaneurizme i krvarenja mrežnice, a kasnije i teža stanja poput neovaskularizacije koja opisuje stvaranje novih abnormalnih krvnih žila koje urastaju u rožnicu. Ako se bolest ostavi bez nadzora, može dovesti i do oštećenja i gubitka vida. Dijeli se na:

- Neproliferativna retinopatija je početni stadij bolesti koji izaziva mikroaneurizme mrežnice i ostale rane simptome retinopatije od kojih je jedan edem makule koji uzrokuje

gubitak vida. Simptomi neproliferativne retinopatije nisu uočljivi, ponekad se javlja mutan vid te točkasta krvarenja u mrežnici.

- Proliferativna neuropatija je kasniji i ozbiljniji stadij bolesti koji je obilježen stvaranjem novih abnormalnih krvnih žila na površini mrežnice, pa čak i šarenice što dovodi do teškog i u nekim slučajevima nepovratnog gubitka vida. Simptomi proliferativne retinopatije su mutan vid, crne točke i svjetla u vidnom polju te gubitak vida ukoliko dođe do krvarenja u staklovinu oka.

Rani stadiji retinopatije se liječe pojačanom kontrolom razine glukoze u krvi kako bi se usporio tijek bolesti. U kasnijim stadijima neka stanja se mogu liječiti laserskom tehnologijom, no ako je došlo do gubitka vida, prognoze vraćanja istog su veoma loše.

2. Dijabetična neuropatija

Dijabetična neuropatija događa se radi slabe prokrvljenosti, tzv. ishemije živaca zbog mikrovaskularnih promjena uzrokovanih dijabetesom. Ilustracija dijabetične neuropatije je preuzeta s [13] i prikazana na slici 3.1. Slaba prokrvljenost izaziva oštećenja perifernog živčanog sustava te je najčešća komplikacija šećerne bolesti. Neki oblici neuropatije su bolni i remete normalno funkcioniranje pacijenata, a neki su asimptomatski. Neuropatija se dijeli na:

- Simetrična polineuropatija je oblik neuropatije koja pogađa periferne udove – šake i stopala,a manifestira se gubitkom osjeta dodira, osjećajem trnjenja te dijastazama. Radi smanjenog osjeta bolesnici imaju iskrivljenu percepciju osjeta te radi toga često dolazi do infekcija, lomova kostiju i sl. Neuropatija tankih niti izaziva trnce i bolove, a neuropatija debelih niti slabost i gubitak osjeta. Pregledom stopala prikazanim na slici 3.1. liječnik može dijagnosticirati ovakvu vrstu neuropatije.
- Autonomna neuropatija je opis za bolest oštećenja vlakana autonomnih živaca, često dolazi uz perifernu neuropatiju. Karakterizira ju gubitak kontrole mjehura, erektilna disfunkcija, zatvor (opstipacija) te gubitak osjeta u perifernim udovima.

Simptomi neuropatije mogu se ublažiti, ali ne i ukloniti pošto je to degenerativna bolest. Strogo kontroliranje glukoze u krvi pogoduje usporavanju razvoja bolesti.

Slika 0.1 Pregled osjeta stopala za dijagnozu neuropatije [13]

3. Dijabetična nefropatija

Dijabetična nefropatija je najčešći uzrok zatajenja bubrega i potrebe za medicinskim postupkom dijalize. Bolest je bubrega koja nastaje nakon dugogodišnjeg bolovanja od dijabetesa. Karakteriziraju ju funkcionalne i strukturalne bubrežne promjene, a najčešće je dijagnosticirana otkrićem albumina u mokraći, pošto bubrezi zdravog čovjeka filtriraju albumin prije dolaska u mokraću. Manifestira se u tri stadija:

- Mikroalbuminurija je prva faza dijabetične nefropatije koju karakterizira pojava albumina u mokraću. Ukoliko je otkriveno u ranoj fazi, može se izliječiti.
- Proteinurija je stanje nefropatije koje se događa ukoliko se mikroalbuminija ostavi neliječenom. Liječnik kontrolira konstantno pojavljivanje bjelančevina u urinu jer je to znak nepopravljivog oštećenja bubrega.
- Zatajenje bubrega se događa ukoliko se proteinurija ostavi neliječenom. Bubrezi ne obavljaju svoju funkciju te bolesnici moraju provoditi postupak dijalize ili u krajnjim slučajevima transplantacije bubrega.

Rani simptomi bolesti su vrlo blagi te se najčešće otkrivaju laboratorijskim pretragama na albumin. U kasnijim stadijima bolesti simptomi se pogoršavaju te izazivaju gubitak teka, mučninu i povraćanje, a ako nisu tretirani i potpuno zatajenje bubrega. Pacijenti koji pravilno provodne liječenje imaju iznimno dobru prognozu potpunog izlječenja od bolesti. Kao i kod retinopatije, baza liječenja je strogo kontroliranje glukoze u krvi u svrhu usporavanja tijeka bolesti.

Osim mikrovaskularnih, dijabetes izaziva i makrovaskuarne komplikacije:

1. Ateroskleroza

Ateroskleroza je medicinski pojam za bolest koja pogađa arterije. Stijenke arterija postaju tanje i manje elastične te se uz njihovu stijenku nakuplja masno tkivo koje može prouzročiti začepljenje krvnih žila. Zahvaća sve arterije u ljudskom tijelu, arterije srca, mozga i drugih vitalnih organa. Jedan je od glavnih uzroka srčanih i moždanih udara. Ateroskleroza ne izaziva simptome dok ne dođe do potpunog začepljenja krvne žile. U svrhu prevencije ove bolesti potrebno je strogo kontrolirati glikemiju u krvi te voditi općenito zdrav život.

2. Dijabetična kardiomiopatija

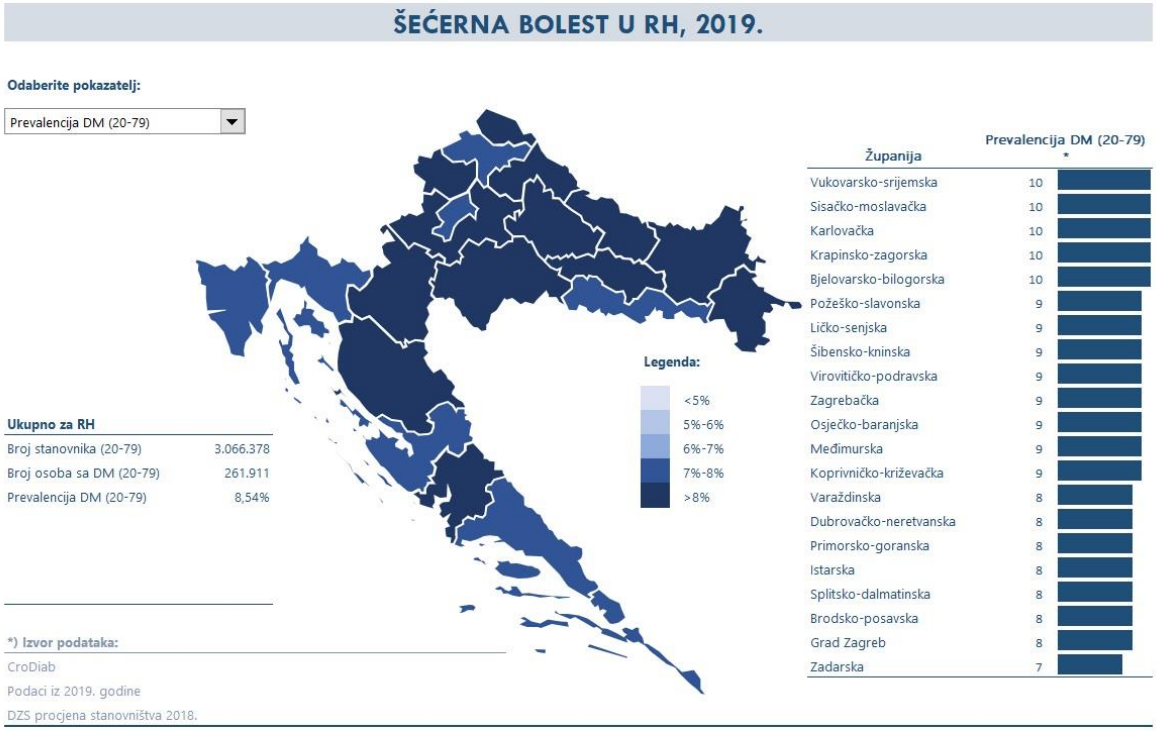
Dijabetička kardiomiopatija je bolest koju karakteriziraju funkcionalne i strukturalne promjene u strukturi miokarda. Vezana je uz dugotrajnu izloženost visokoj razini glukoze u krvi. Ako se ne liječi na vrijeme, može dovesti do zatajenja srca. U početnom stadiju radi hiperglikemije, srce ima manjak opskrbe hranjivim tvarima, pa to dovodi do poremećaja u funkciji. Stanice odumiru i nakupljaju se u miokardu što uzrokuje poremećaj u sistoličkoj i/ili dijastoličkom funkcijom srca radi pogoršanja u kontrakcijama. Kasnije se događa progresija bolesti stanjivanjem krvnih žila, tj. sklerozom koja dovodi do nedostatka kisika i smanjenje funkcije miokarda.

Osim krvožilnih promjena, bolesnici su podložni i raznim gljivičnim i bakterijskim infekcijama te kožnim promjenama, kao što su gangrene, radi slabe prokrvljenosti i neuropatije živaca na perifernim dijelovima tijela.

3.4.3. Smrtnost i broj oboljelih od dijabetesa

Kao što se iz prethodnih poglavlja može zaključiti, dijabetes je ozbiljna bolest koja neliječenjem može dovesti do ozbiljnih zdravstvenih komplikacija, pa čak i smrti. Naziva se epidemijom modernog doba pošto broj oboljelih na svjetskoj razini raste velikom brzinom. Svjetska zdravstvena organizacija (WHO) tvrdi da je broj oboljelih narastao od 108 milijuna dijagnosticiranih u 1980. godini na 422 milijuna u 2014. godini [14]. Čak 8.5 % odraslih osoba na svijetu ima neki oblik dijabetesa. Sa sve većim problemom užurbanog života, nezdrave prehrane i manjka fizičke aktivnosti, dijabetes postaje jedan od vodećih razloga smrtnosti populacije.

U Republici Hrvatskoj je 2019. godine zabilježeno 315 298 osoba s nekim oblikom dijabetesa te taj broj raste proporcionalno s vremenom [16]. Pojavljuje se i problem nepostavljene dijagnoze, pošto istraživanja prikazuju da samo 60 % osoba u RH ima postavljenu dijagnozu dijabetesa. Statistika za dijabetes u RH prikazana je na slici 3.2 te je preuzeta s [16].



Slika 0.2 Prikaz statistike šećerne bolesti u RH 2019. godine [16]

3.5. Prevencija dijabetesa

S obzirom na rastući problem dijabetesa, potrebno je podučiti ljude da proaktivno sudjeluju u unapređenju svoga zdravlja i prevenciji te bolesti. U današnjem digitalnom dobu postoji mnogo računalnih resursa koji mogu poslužiti u svrhu educiranja korisnika o važnosti i načinima prevencije dijabetesa. Pošto je dijabetes tipa I većinom genetski uvjetovan, takav tip dijabetesa ne može se prevenirati. No, tip dijabetesa koji čini 90 % svih slučajeva dijabetesa je tip II, koji je moguće prevenirati uz preinake stila život i svakodnevnih navika.

3.5.1. Čimbenici rizika

Čimbenici rizika za razvoj dijabetesa tipa II su razni te se sastoje od čimbenika uvjetovanih genetikom do onih uvjetovanih stilom života. Faktori rizika su:

- Dob veća od 45 godina
- Indeks tjelesne mase (BMI) – kg/m2 - veći od 30
- Opseg struka (cm) veći od 102 za muškarce i 88 za žene
- Rijetko bavljenje fizičkom aktivnošću
- Ne konzumiranje voća, povrća i žitarica
- Korištenje lijekova za kontroliranje krvnog tlaka
- Povišena razina šećera na ranijim pregledima

- Pušenje
- Polidipsija – neutaživa žeđ
- Poliurija – prekomjerno mokrenje
- Polifagija – neutaživa glad
- Konstantan umor
- Nejasan vid
- Neobjašnjiv gubitak tjelesne mase
- Vaginalne infekcije
- Erektilna disfunkcija
- Problemi sa stopalima
- Rane koje ne zacjeljuju
- Dijabetes u obitelji
- Rođenje djeteta težeg od 4 kg
- Sistolički tlak viši od 135 mmHg
- Dijastolički tlak viši od 90 mmHg
- LDL kolesterol veći od 3.38 mmol/L
- HDL kolesterol niži od 5.27 mmol/L
- Glukoza u krvi viša od 5.72 mmol/L

Na neke od ovih faktora rizika moguće je utjecati, no neki poput godina, genetike i ponašanja u prošlosti ne mogu se promijeniti. No čak i promjenom ovih ostalih navika, rizik za dobivanje dijabetesa će se znatno smanjiti.

3.5.2. Postupci prevencije dijabetesa

Kako bi osoba utjecala na neke od nabrojanih čimbenika dijabetesa mora promijeniti životne navike [15]. Neki od preporučenih savjeta i postupaka prevencije dijabetesa su:

1. Redukcija tjelesne mase – kontroliranjem tjelesne mase kontrolira se i unos hrane u organizam, te se limitira unošenje namirnica koje drastično utječu na podizanje razina glukoze u krvi i stvaranje masnih naslaga
2. Konzumiranje svježih namirnica, voća i povrća, organske hrane koja će opskrbiti tijelo osobe svim važnim mineralima. Potrebno je unositi umjerenu količinu šećera te izbjegavati procesiranu hranu.

3. Prestati pušiti – pušenje je jedan od faktora rizika za inzulinsku rezistenciju i razvijanje dijabetesa tipa II
4. Baviti se fizičkom aktivnošću – mnoga istraživanja dokazuju da tjelesna aktivnost pogoduje zdravlju te smanjuje razine šećera u krvi.

3.6. Utjecaj fizičke aktivnosti na kvalitetu života oboljelih

Konstitucija čovjeka nalaže da se mora baviti fizičkom aktivnošću. Mišići su stvoreni da se stežu i opuštaju, miču zajedno uz kosti i zglobove. Iz tog razloga sve više stručnjaka tvrdi kako sjedilački život i manjak aktivnosti pogoduje mnogim bolestima modernog doba, pa tako i dijabetesu. Nedostatak gibanja može uzrokovati razne poremećaje u metabolizmu glukoze te ubrzati razvoj bolesti te shodno tome i smrti [17]. Jedan od temeljnih načina liječenja dijabetesa je upravo fizička aktivnost. Istraživanja su dokazala kako vježbanje pozitivno utječe na smanjenje inzulinske rezistencije, regulaciji tjelesne mase pa čak i smanjenu razine glukoze u krvi. Intenzivan trening brže troši zalihe glukoze iz plazme te čini organizam osjetljivijim na inzulin. Iz tog razloga, pacijenti se moraju posavjetovati s izabranim liječnikom kako bi odredili koja vrsta fizičke aktivnosti je adekvatna za njihovo zdravstveno stanje. Radi smanjenja šećera u krvi, fizička aktivnost smanjuje i količinu lijekova koji su oboljelima potrebni te smanjuje rizik za komplikacije dijabetesa opisane u ranijim poglavljima [18].

3.6.1. Dijabetes tipa I i fizička aktivnost

Pošto je dijabetes tipa I uvjetovan genetikom i okolišem, česta zabluda je da fizička aktivnost ne može pomoći u lakšem podnošenju te bolesti i ublažavanju simptoma. Ova vrsta dijabetesa liječi se aktivno inzulinom, a pasivno regulacijom prehrane i tjelovježbom. Pošto je ubrizgavanje inzulina ključno za kontrolu bolesti, cilj fizičke aktivnosti je smanjiti dozu lijekova koje su pacijentima potrebne. Tjelovježbom mišići troše više glukoze nego u normalnim uvjetima te tijelo naglo počinje trošiti glikogen. Što je trening vremenski duži, to se više glukoze i zaliha glikogena potroši iz krvi. U organizmu dolazi do pada razine hormona inzulina te u bolesnika oboljelih od dijabetesa postoji rizik ulaska u stanje hipoglikemije, tj. premalene razine šećera u krvi. Taj rizik pokušava se prevenirati pomnim planiranjem fizičke aktivnosti uz pomoć liječnika. Preporučuje se vježbati nekoliko sati nakon obroka te izbjegavati vježbanje ako je razina glukoze u krvi bolesnika manja od 4.4 mmol/L ili veća od 13.6. mmol/L.

Istraživanje "Učinci tjelesne aktivnosti aerobnog i anaerobnog tipa na smanjenje doze inzulina kod dijabetičara" provelo se nad 30 ispitanika, 17 žena i 13 muškaraca u dobi od 16 do 49 godina te mu je cilj dokazati povoljno utjecanje fizičke aktivnosti na smanjenje doza inzulina

koje bolesnici moraju primati te da će razina inzulina biti smanjena u odnosu na onu koja im je inače potrebna. Dio ispitanika radio je isključivo aerobne treninge poput trčanja, vožnje bicikla i sl., drugi dio ispitanika radio je isključivo anaerobne vježbe bazirane na vježbama s vlastitom težinom, utezima i fokusom na aktiviranje raznih mišićnih skupina, a ostatak ispitanika radio je kombinaciju tih vrsta treninga. Osim toga, ispitanici su mjerili razinu glukoze u krvi prije i poslije treninga. Rezultati istraživanja pokazali su da se razina glukoze u krvi znatno smanjila kao i količina inzulina koju su bolesnici morali uzimati dnevno. Utjecaj tjelesne aktivnosti po dobnim skupinama na smanjenje jedinica inzulina može se vidjeti na tablici 3.2 te je preuzeto iz istraživanja [18].

Tablica 0.2 Utjecaj tjelesne aktivnosti po dobnim skupinama na smanjenje jedinica inzulina [18]

Dobna skupina	Broj ispitanika	Prosječno smanjenje jedinica inzulina (broj)	Prosječno smanjenje jedinica inzulina (%)	Prosječna količina inzulina prije	Prosječno vrijeme treniranja
16-20	8	10,3	24,30	52,9	73,1
21-30	14	8,5	20,30	50,4	61,1
31-40	5	11,6	28,08	53,2	54,0
>40	3	7,8	17,02	56,1	40,0
Ukupno	30	9,4 +-3,9	22,33 +-9,57	52,1+-18,4	61,0+-22,9

3.6.2. Dijabetes tipa II i fizička aktivnost

Glavni način prevencije dijabetesa tipa II upravo je regulacija prehrane i tjelesne aktivnosti. Osobe s poremećenim metabolizmom glukoze i intolerancijom na šećer, vježbanjem mogu smanjiti rizik obolijevanja od ozbiljnijeg oblika bolesti, oni koji već boluju od dijabetesa, vježbanjem mogu utjecati na vlastitu regulaciju glukoze te smanjenje rizik od obolijevanja od težih stanja povezanih s dijabetesom. Fizička aktivnost smanjuje inzulinsku rezistenciju na način da se poboljšava rad receptora inzulina GLUT4 koji su zaslužni za propuštanje glukoze u stanice organizma. Istraživanje objavljeno 2015. godine opisuje promatranje oboljelih od dijabetesa tip II u periodu od 20 godina i utjecaj tjelovježbe na njihovo zdravstveno stanje [19]. Dokazano je da je ukupna smrtnost oboljelih smanjena za čak 39 – 70 % ukoliko su se redovno bavili intenzivnim kardio treninzima i/ili nekim oblikom fizičke aktivnosti koji im je povećao kardiorespiracijsku sposobnost.

U sljedećem poglavlju bit će definirani zahtjevi i arhitektura aplikacije koja bi pružila potporu oboljelima od dijabetesa te educirala društvo o važnosti prevencije.

4. ARHITEKTURA APLIKACIJE

Arhitektura aplikacije obuhvaća definirane zahtjeve na aplikaciju te razradu idejnog rješenja i strukture aplikacije. Osim toga, ovo poglavlje sadržava opisanu analizu rizika koja se vrši u aplikaciji te kriterije po kojima se određuje razred rizika u koji korisnik aplikacije pripada.

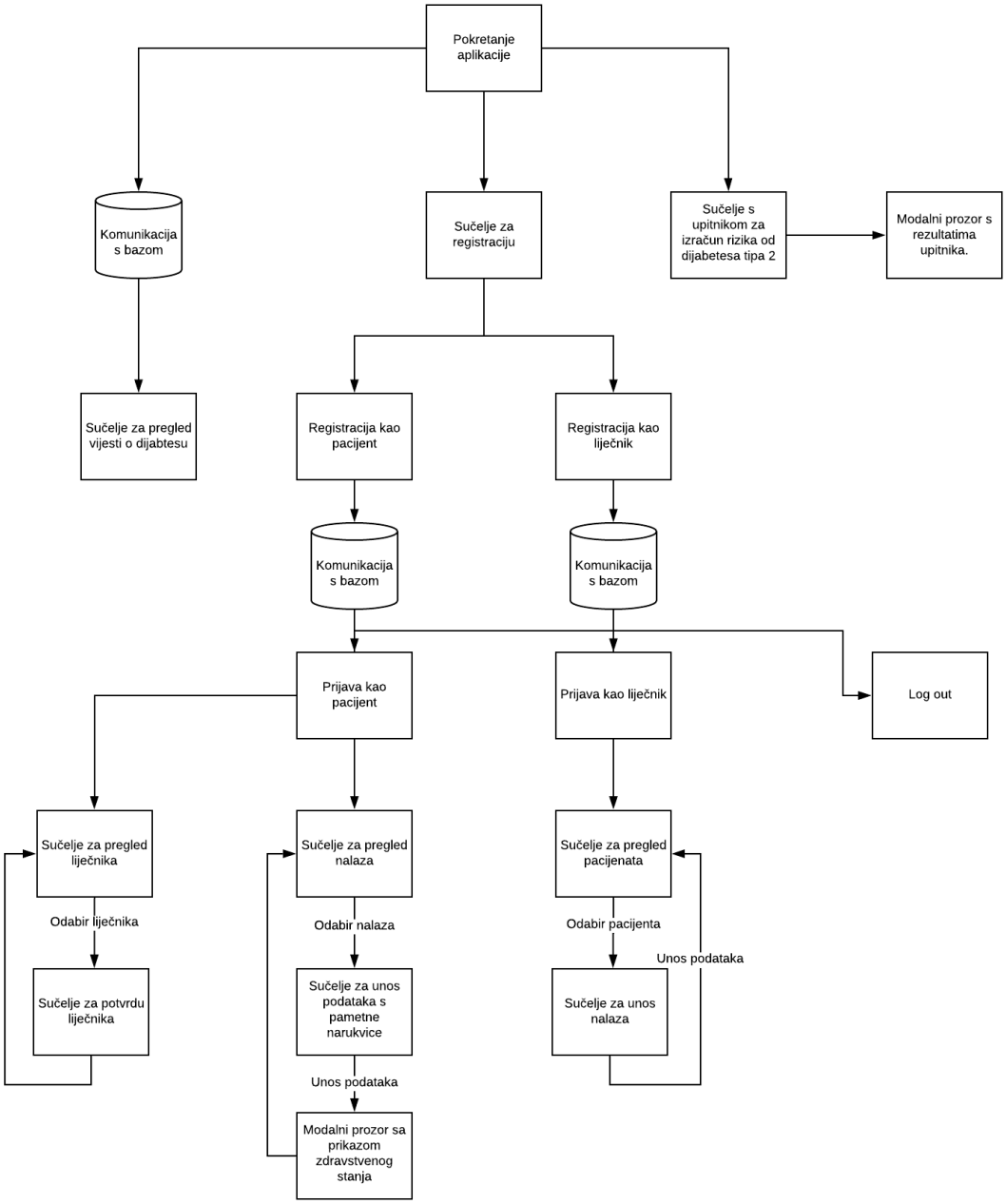
4.1. Zahtjevi na aplikaciju

Kako bi se započelo s razvojem same aplikacije, potrebno je definirati korisničke scenarije i zahtjeve koje ona mora ispunjavati. Ideja aplikacije je sustav koji educira korisnika o dijabetesu i pruža mu mogućnost izračuna vlastitog rizika od dijabetesa tipa 2 popunjavanjem upitnika, a u isto vrijeme ima funkcionalnost bliske suradnje s odabranim liječnikom u svrhu analiziranja pacijentovih nalaza i podataka s pametne narukvice. Uzevši to u obzir, aplikacija je razdijeljena na dva glavna dijela – javni portal i sučelje prijavljenog korisnika. Definirani zahtjevi na svaki od ta dva dijela su sljedeći:

1. Javni portal
    - Korisnik ima pregled vijesti i informacija o dijabetesu
    - Korisnik može ispuniti upitnik za izračun rizika od dijabetesa tipa 2
    - Korisnik se može registrirati kao pacijent
    - Korisnik se može registrirati kao liječnik
2. Sučelje prijavljenog korisnika
    - Korisnik se može prijaviti u aplikaciju u ulozi u kojoj se registrirao
    - Korisnik se može odjaviti iz aplikacije
    - Korisnik u ulozi liječnika ima pregled svojih pacijenata
    - Korisnik u ulozi liječnika ima mogućnost unosa nalaza za svoje pacijente
    - Korisnik u ulozi pacijenta ima mogućnost odabira liječnika
    - Korisnik u ulozi pacijenta ima mogućnost pregleda vlastitih nalaza
    - Korisnik u ulozi pacijenta ima pregled zdravstvenog statusa na osnovi nalaza koje je unio odabrani liječnik
    - Korisnik u ulozi pacijenta ima mogućnost unosa podataka s pametne narukvice
    - Korisnik u ulozi pacijenta nakon unosa podataka s pametne narukvice dobije obavijest o statusu nalaza i unesenih podataka

4.2. Idejno rješenje web aplikacije

Idejno rješenje aplikacije prikazano je na slici 4.1. Predstavlja zamišljeni tok korisnikova korištenja aplikacije definiran putem razrađenih zahtjeva na aplikaciju. Kao što je navedeno u prošlom poglavlju, aplikacija je podijeljena na javni dio i dio za koje je potrebna autentikacija. Neprijavljeni korisnik ima mogućnost pregleda vijesti o dijabetesu te ispunjavanja upitnika rizika. Prijavljeni korisnik ima sve mogućnosti kao i neprijavljeni, ali i dodatne funkcionalnosti komunikacije pacijenta i liječnika.



Slika 4.1. Dijagram tijeka korištenja web aplikacije

4.3. Struktura aplikacije

Za programsko rješenje bilo je potrebno razviti poslužiteljski (engl. *back-end*) i klijentski (engl. *front-end*) dio. Iz tog razloga glavna podjela strukture je na direktorije *client* i *server* što je prikazano na slici 4.2. U *server* direktoriju implementiran je pristup bazi podataka, autentikacija korisnika i održavanje sesije (engl. *session*) te sva poslovna logika aplikacije – izračun rizika, pacijent-liječnik komunikacija i sl. Direktorij je podijeljen na datoteke i poddirektorije:

- Helpers – direktorij s pomoćnim metodama koje se često koriste u aplikaciji, sadržava funkcije za autentikaciju, provjeru korisničke uloge te validaciju na ulaznim podacima
- Models – direktorij koji sadrži shematske prikaze podataka koji će se u tom obliku spremati u bazu podataka
- Passport – Direktorij koji sadrži datoteku za postavljanje biblioteke za autentikaciju
- Routes – Direktorij u kojemu se definira logika aplikacije te putanje REST API zahtjeva koji se koriste u aplikaciji
- Index.js – Ulazna točka aplikacije, file koji se prvi pokreće kod izvršavanja programskog koda
- Roles.js – Datoteka sa šifrarnikom koji sadrži korisničke uloge u aplikaciji
- .env – Datoteka koja sadrži tajne podatke o povezivanju s bazom podataka i kriptiranjem sesije

```
|   index.js|
|   package.json
|   roles.js
+---helpers
|       checkAuth.js
|       validation.js
+---models
|       Activity.js
|       Post.js
|       Test.js
|       User.js
+---node_modules
+---passport
|       setup.js
\---routes
        activity.js
        auth.js
        patients.js
        posts.js
        risk.js
        tests.js
        users.js
```

Slika 4.2 Struktura direktorija poslužiteljske strane aplikacije

Struktura *client* direktorija prikazana je na slici 4.3. Unutar tog direktorija implementiran je klijentski dio aplikacije – prikaz podataka na korisničkom sučelju, izgled web aplikacije te komuniciranje s poslužiteljem. Podijeljen je na:

- Components – Direktorij koji sadrži komponente programskog okvira React koje se koriste na raznim sučeljima aplikacije. Unutar ovog direktorija definira se ponašanje elemenata na korisničkom sučelju i njihov izgled.
- Views – Direktorij koji sadrži komponente koje sadržavaju glavna sučelja aplikacije.
- Style – Direktorij koji sadrži CSS datoteke koje stiliziraju komponente koje su prikazane na korisničkom sučelju.
- Assets – Direktorij koji sadrži logo aplikacije.
- App.js – Ulazna točka klijentskog dijela aplikacije. Unutar datoteke su definirane putanje na kojima se nalazi svaki od dijelova korisničkog sučelja te tema aplikacije.

```
|   App.js
|   index.js
+---components
|       DoctorDialog.js
|       DoctorTable.js
|       HeroImage.js
|       Nav.js
|       PatientDialog.js
|       PatientTable.js
|       Post.js
|       PostList.js
|       RiskResults.js
|       TestResultsDialog.js
|       TestsDialog.js
|       TestTable.js
+---style
|       App.css
|       Dialog.css
|       Form.css
|       HeroImage.css
|       index.css
|       modal.css
|       Nav.css
|       Post.css
|       RiskForm.css
|       Table.css
\---views
        ChooseDoctor.js
        Homepage.js
        Login.js
        MyPatients.js
        MyTests.js
        Register.js
        RiskAssesment.js
```

Slika 4.3 Struktura direktorija klijentske strane aplikacije

4.4. Analiza zdravstvenog stanja i rizika obolijevanja od dijabetesa

Kao što je već spomenuto, na poslužiteljskoj strani aplikacije implementirano je računanje rizika od obolijevanja dijabetesom tipa 2. Izračun dijabetesa preuzet je s upitnika objavljenog na stranicama Hrvatskog zavoda za javno zdravstvo [20]. Upitnik obuhvaća razne parametre kojima se dodjeljuje brojčana vrijednost. Ukupnim zbrojem svih parametara računa se rizik obolijevanja dijabetesom. Parametri i njihovo bodovanje preuzeti su s [20] i prikazani u tablici 4.1.

Tablica 4.1 Analiza parametara za procjenu rizika od dijabetesa [20]

Parametar	**Vrijednost**		**Bodovi**
Dob	Manje od 45 godina		0
	45 – 54 godine		2
	55 – 64 godine		3
	Više od 64 godine		4
Indeks tjelesne mase	Manji od 25 kg/m^2		0
	25 – 30 kg/m^2		1
	Veći od 30 kg/m^2		3
Opseg struka	Muškarci	Manji od 94 cm	0
		94 – 102 cm	3
		Veći od 102 cm	4
	Žene	Manji od 80 cm	0
		80 – 88 cm	3
		Veći od 88 cm	4
Tjelesna aktivnost 30 minuta dnevno	Da		0
	Ne		2
Konzumacija voća i povrća	Svaki dan		0
	Ne svaki dan		1
Povišen šećer u krvi na ranijim pregledima	Da		5
	Ne		0
Uzimanje lijekova za visok tlak	Da		2
	Ne		0
Dijagnoza dijabetesa u obitelji	Ne		0
	Uža obitelj		5
	Šira obitelj		2

Zbroj bodova svih parametara predstavlja razrede rizika od dijabetesa. Rizik od razvijanja dijabetesa u sljedećih 10 godina po bodovima je sljedeći:

1. Nizak rizik - manje od 7 bodova, procjenjuje se da će 1 od 100 osoba razviti dijabetes
2. Blago povišen rizik – između 7 i 11 bodova, procjenjuje se da će 1 od 25 osoba razviti dijabetes
3. Umjereno povišen rizik - između 12 i 14 bodova, procjenjuje se da će 1 od 6 osoba razviti dijabetes
4. Visok rizik - između 15 i 20 bodova, procjenjuje se da će 1 od 3 osoba razviti dijabetes
5. Vrlo visok rizik – više od 20 bodova, procjenjuje se da će 1 od 2 osobe razviti dijabetes

Osim računanja rizika, u aplikaciji se korisniku nakon unesenih podataka s pametne narukvice i podataka liječnika ispisuje status zdravstvenog stanja koje se određuje po rangiranju parametara prikazanih u tablici 4.2.

Tablica 4.2 Analiza zdravstvenog stanja pacijenta po parametrima

Parametar	**Vrijednost**	**Status**
Otkucaji srca [21]	< 60 otkucaja u minuti	Preniska vrijednost
	60 – 100 otkucaja u minuti	Optimalna vrijednost
	> 100 otkucaja u minuti	Previsoka vrijednost
Broj koraka u danu	< 5000 koraka	Preniska vrijednost
	5000 – 7500	Normalna vrijednost
	>7500	Optimalna vrijednost
Prehodana udaljenost	< 3 km	Preniska vrijednost
	> 3 km	Normalna vrijednost
Sistolički tlak [22]	< 120 mmHg	Normalna vrijednost
	120 – 129 mmHg	Povišena vrijednost
	130 – 139 mmHg	Vrijednost koja ukazuje na visok krvni tlak
	140 – 180 mmHg	Vrijednost koja ukazuje na visok krvni tlak
	> 180 mmHg	Vrijednost koja ukazuje na hipertenzivnu krizu
Dijastolički tlak [22]	< 80 mmHg	Normalna vrijednost
	80 – 90 mmHg	Povišena vrijednost
	90 – 120 mmHg	Vrijednost koja ukazuje na visok krvni tlak
	> 120 mmHg	Vrijednost koja ukazuje na hipertenzivnu krizu
Glukoza u krvi [23]	< 3.9 mmol/L	Niska vrijednost
	3.9 – 5.5 mmol/L	Optimalna vrijednost
	> 5.5 mmol/L	Visoka vrijednost
Kolesterol HDL [24]	< 40 mg/dL	Preniska vrijednost
	> 40 mg/dL	Normalna vrijednost
Kolesterol LDL [25]	< 100 mg/dL	Optimalna vrijednost
	100 – 129 mg/dL	Normalna vrijednost

	130 – 159 mg/dL	Granično visoka vrijednost
	160 – 189 mg/dL	Visoka vrijednost
	> 180 mg/dL	Vrlo visoka vrijednost

Analiza rizika i opisani postupci izračuna rizika bit će programski implementirani na način opisan u poglavlju 5.

5. PROGRAMSKO RJEŠENJE WEB APLIKACIJE

5.1. Korištene tehnologije

Kako bi se optimalno implementirali zadani zahtjevi na aplikaciju, potrebno je odabrati tehnologije za izradu web rješenja.

5.1.1. HTML

Korisničko sučelje zasniva se na HTML-u. HTML je kratica za *HyperText Markup Language* te mu ime zapravo opisuje glavnu funkciju - strukturiranje web stranica. Osmislio ga je Tim Berners Lee krajem 1991. godine, a kao programski jezik prvi put je objavljen 1993. godine. Jednostavan je za korištenje te mu je glavna zadaća poslati željene podatke o strukturi web preglednicima Radi na principu oznaka (engl. *tagova*) koji opisuju izgled komponenti na korisničkom sučelju. Dva taga tvore HTML elemente koji zatim grade izgled aplikacije. Svakom elementu mogu se dodati atributi (engl. *attributes*) koji dodatno opisuju taj element – bilo dimenzijama, položajem ili u slučaju slika i linkova putanja do izvornih datoteka (engl. *Source files*).

5.1.2. CSS

CSS dolazi od engleskog naziva *Cascade Style Sheets* te se rabi za oblikovanje i manipulaciju HTML elementima. Stilski je jezik te ne prikazuje sadržaj nego samo izgled i raspored stranice. Osnovna gradivna jedinica mu je tzv. *stylesheet* za pisanje kojega postoje pravila. Svako CSS pravilo sastoji se od selektora i deklaracije. Selektori označavaju dio HTML elemenata koje će biti stilizirani. HTML elementi se mogu dohvatiti na nekoliko načina – bilo putem same vrste elementa – pr. <h1>, <p>. klase koja je atribut nekog od tipova ili identifikatora po kojemu je element označen. Drugi dio CSS pravila je deklaracija u kojoj se zapravo zapisuje sam stil koji se želi primijeniti na HTML element.

5.1.3. Razvojni okvir React.js

React.js je razvojni okvir programskog jezika JavaScript otvorenog tipa koji se koristi za gradnju korisničkog sučelja. Prvi put se pojavio 2013. godine te ga je osmislio Jordan Walke te . Koristi se za razvijanje SPA (engl. *Single Page Application*) web rješenja i mobilnih rješenja. React je jedan od najpopularnijih alata za *front-end* development. Glavni dio tog programskog jezika su komponente. Komponente se mogu primijeniti na elemente u DOM-u (engl. *Document Object Model*) koji su definirani HTML-om.

5.1.4. Platforma Node.js

Node.js je platforma prigodna za izradu aplikacija u stvarnom vremenu. Počiva na programskom jeziku JavaScript i izvršava se izvan preglednika. Kod u Node.js aplikacijama izvršava se asinkrono, što pogoduje brzini izvršavanja. Programski kod 5.1 primjer je koda pisanog u React razvojnom okviru.

```
const passport = require('passport')

module.exports = {
    checkAuthenticated: function (req, res, next) {
        if (req.isAuthenticated()) {
            return next();
        }
        return res.status(401).send('Not logged in');
    },
    checkNotAuthenticated: function (req, res, next) {
        if (req.isAuthenticated()) {
            res.redirect('/');
        }
        next();
    },
    checkRole: function (role) {
        return (req, res, next) => {
            if (req.user.role !== role) {
                return res.status(403).send('No permission');
            }
            next();
        }
    }
};
```

Programski kod 5.1. Primjer programskog koda pisanog u Node.js

5.1.5. Baza podataka MongoDb

MongoDb je sustav za spremanje podataka koji je dokumentno orijentiran. Mongo je noSQL baza podataka, što znači da među tablicama ne postoje relacije. Pomoću MongoDb moguće je manipulirati podacima, pretraživati ih, spremati i koristiti ih na bilo koji željeni način. Podatke spremna u formatu BSON koji je vrlo sličan JSON-u. Sastoji se od zbirki koje sadrže dokumente. Primjer podatka u MongoDb bazi prikazan je na slici 5.1.

Slika 5.1 Primjer podatka u MongoDb

5.1.6. Programski okvir Express.js

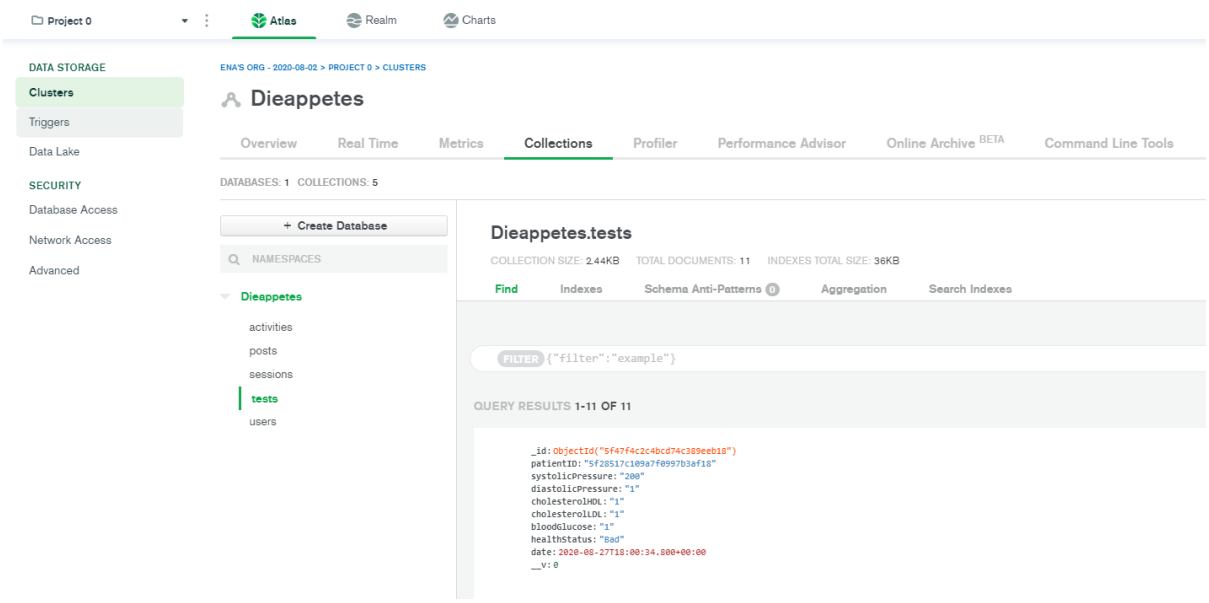
Express.js je programski okvir razvijen za Node.js. Objavljen je 2010. godine te je otvorenog koda (engl. *open source*) i besplatan za korištenje. Dizajniran je za implementaciju web aplikacija i API-ja.

5.2. Programsko rješenje aplikacije

Glavni dijelovi programskog rješenja za potporu oboljelima od dijabetesa obuhvaćaju funkcionalnosti navedene u poglavlju 4.1. Za svako od razvijenih sučelja implementiran je klijentski i poslužiteljski dio. U sljedećim potpoglavljima prikazat će se svaka od glavnih komponenti sustava uz pripadajući programski kod kojim je realizirana.

5.2.1. Baza podataka

Kao što je već spomenuto u poglavlju 5.1., za spremanje podataka unutar aplikacije korištena je MongoDb Atlas baza u oblaku. MongoDb je nerelacijska baza podataka koja se sastoji od zbirki (engl. *collections*) koje sadrže skup dokumenata. Podaci se u MongoDb spremaju u BSON9 formatu. Izgled baze podataka i zbirki unutar nje prikazan je na slici 5.2.



Slika 5.2 Izgled baze podataka web aplikacije

Povezivanje s bazom podataka ostvareno je putem Mongoose alata te je prikazano je u programskom kodu 5.2. Kako bi aplikacija imala mogućnost povezivanja na bazu podatka, dodan je korisnik koji ima prava na čitanje i pisanje u bazu te se njegovi pristupni podaci šalju uz poveznicu do baze u Mongoose funkciju *connect()*. Pristupni podaci za bazu podataka prikazani su u programskom kodu 5.3.

```
mongoose.connect(process.env.DB_CONN_STRING, { useNewUrlParser: true },
() => {
    console.log("Db")
})
```

Programski kod 5.2. Prikaz povezivanja aplikacije s bazom podataka

```
DB_CONN_STRING=mongodb+srv://sadmin:Pa$$word123@dieappetes.rs1ha.mongodb.net/Dieappetes?retryWrites=true&w=majority
```

Programski kod 5.3. Pristupni podaci za bazu podataka

Zbirke (engl. *collections*) koje sadrže korisničke podatke se tvore kroz programski kod korištenjem modela podataka u obliku Mongoose shema podataka. Kako bi se izradio model podataka potrebno je definirati atribute koje želimo da zbirka podataka ima te njihov tip podatka. Za svaki od podataka moguće je definirati minimalnu i maksimalnu vrijednost te hoće li podatak biti obvezan ili ne. U sklopu aplikacije bilo je potrebno izraditi 5 zbirki kako bi se ostvarile sve zahtijevane funkcionalnosti:

- Activities
- Posts
- Tests
- Users
- Sessions

Zbirka podataka *activities* sadrži podatke s pametne narukvice koje korisnik unosi – brzina otkucaja srca, broj prehodanih koraka, prehodana udaljenost. Osim podataka s pametne narukvice, sprema se i podatak o identifikacijskom broju pacijenta koji unosi te podatke i datumu kada ih je unio. Modelirana je shemom Activity.js prikazanom u programskom kodu 5.4.

```
const ActivitySchema = mongoose.Schema({
    patientID: {
        type: String,
        required: true
    },
    heartRate: {
        type: String,
        required: true
    },
    distance: {
        type: String,
        required: true
    },
    stepsNum: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('Activity', ActivitySchema)
```

Programski kod 5.4. Prikaz Activity.js sheme podataka

Zbirka *posts* sadrži podatke o vijestima o dijabetesu koji su prikazani na sučelju "News". Sadrži podatke o naslovu i opisu vijesti te datumu i vremenu kada je vijest bila kreirana. Modelirana je shemom podataka Post.js prikazanom u programskom kodu 5.5.

```
const PostSchema = mongoose.Schema({
    title: {
        type: String,
        required: true
    },
    description: {
        type: String,
        required: true
    },
    image: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('Posts', PostSchema)
```

Programski kod 5.5.Prikaz Post.js sheme podataka

Zbirka *tests* sadrži podatke o nalazima pacijenta koje unosi liječnik – razini glukoze u krvi, dijastoličkom i sistoličkom krvnom tlaku, HDL i LDL kolesterolu te zdravstvenom statusu koji se računa u ovisnosti o vrijednostima navedenih parametara. Ako je ijedna od vrijednosti u previsokom ili preniskom rangu, zdravstveni status pacijenta postavlja se na “Bad”, ukoliko su sve vrijednosti u normalnom rangu status je “Good”. Osim podataka o nalazima, sprema se i podatak o identifikacijskom broju pacijenta kojemu pripadaju uneseni nalazi i datumu kada ih je liječnik unio. Modelirana je shemom podataka Tests.js prikazanom u programskom kodu 5.6.

```
const TestSchema = mongoose.Schema({
    patientID: {
        type: String,
        required: true
    },
    systolicPressure: {
        type: String,
        required: true
    },
    diastolicPressure: {
        type: String,
        required: true
    },
    cholesterolHDL: {
        type: String,
        required: true
    },
    cholesterolLDL: {
        type: String,
        required: true
    },
    bloodGlucose: {
        type: String,
        required: true
    },
    healthStatus: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('Tests', TestSchema)
```

Programski kod 5.6. Prikaz Tests,js sheme podataka

Zbirka *users* sadrži podatke o korisnicima aplikacije koje su unijeli prilikom registracije. – ime, prezime, e-mail adresa, lozinka. Osim toga, sprema se podatak o korisničkoj ulozi kojoj korisnik pripada – pacijentu ili liječniku, identifikacijskom broju liječnika ukoliko je korisnik pacijent te datumu i vremenu kada je korisnik kreiran. Svi parametri su obvezni osim parametra *doctorId*, taj parametar nije obvezan iz razloga što ga posjeduju samo korisnici s ulogom pacijenta te im se dodjeljuje na sučelju za odabir liječnika. Ostali parametri u bazi podataka se populiraju kada se korisnik registrira. Modelirana je shemom podataka User.js prikazanom u programskom kodu 5.7.

```
const UserSchema = mongoose.Schema({
    firstName: {
        type: String,
        required: true
    },
    lastName: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    password: {
        type: String,
        required: true
    },
    role: {
        type: String,
        required: true
    },
    doctorId: {
        type: String
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('User', UserSchema);
```

Programski kod 5.7. Prikaz User.js sheme podataka

Posljednja zbirka unutar programskog rješenja je zbirka *sessions* koja sadrži podatke o sesijama (engl. *sessions*) korisnika koji su se prijavili u aplikaciju – id, datum i vrijeme kada sesija ističe te same podatke o sesiji. Sesija osigurava da aplikacija uvijek zna koji je korisnik prijavljen te su njegovi podaci uvijek raspoloživi. Za upravljanje sesijom korišten je alat Express.js te je način na koji se postavljaju i spremaju podaci o sesiji prikazan u programskom kodu 5.8.

```
app.use(

    session({
        secret: process.env.SESSION_SECRET,
        resave: true,
        saveUninitialized: true,
        store: new MongoStore({ url: process.env.DB_CONN_STRING, collec-
tion: 'sessions' })
    }));
```

Programski kod 5.8. Prikaz spremanja sesije u bazu podataka

5.2.2. Registriranje korisnika

Funkcionalnost registracije novih korisnika implementirana je na način da se na API */api/auth/register* pošalju podaci koje korisnik unese na korisničkom sučelju. Za ostvarenje ove funkcionalnosti implementiran je API upravljač (engl. *controller*) s HTTP metodom POST. Prije no što se izvrši API, poslužitelj provjerava da korisnik nije prijavljen u aplikaciju. Ako je, API se neće izvršiti. Uneseni podaci se nakon uspješne provjere autentikacije validiraju – provjerava se da su svi podaci uneseni te da unutar aplikacije ne postoji korisnik s unesenom e-mail adresom. Provjera postojeće email adrese vrši pretragom u bazi podataka koristeći izrađeni model podataka *Users.js* i MongoDb funkciju *findOne()* kojoj je predana e-mail adresa korisnika koji se pokušava registrirati. Unutar MongoDb baze podataka se zatim pretražuje zbirka *users* i vraća korisnik koji ima email adresu predanu funkciji. Ako takav korisnik postoji, neće biti moguće dovršiti registraciju korisnika s tim podacima. Iz razloga što se u dijelu koda čeka na odgovor iz baze podataka, API je implementiran asinkrono koristeći sintaksu *async* i *await*. Osim toga, korisniku se kriptira lozinka radi sigurnosti korištenjem bcrypt.js biblioteke koju se također potrebno asinkrono pričekati da izvrši kriptiranje. Ukoliko podaci prolaze sve provjere, kreira se model korisnika koje se zatim sprema u bazu podataka koristeći funkciju *save()*. Ako se dogodi nekakva vrsta greške, API će vratiti statusni kod 400 te grešku u odgovoru na zahtjev, ako operacija prođe uspješno, API vraća statusni kod 200. Odgovor API zahtjeva je u JSON formatu podataka. Registracija korisnika prikazana je programskim kodom 5.9.

```
router.post('/register', checkNotAuthenticated, async (req, res) => {
    const { error } = registerValidation(req.body);
    if (error) return res.status(400).send(error.details[0].message);
    console.log(req.body.role)
    //Check if the user already exists
    const emailExist = await User.findOne({ email: req.body.email })
    if (emailExist) return res.status(400).send('Email already exists');

    //Hash a password
    const salt = await bcrypt.genSalt(10);
    const hashPassword = await bcrypt.hash(req.body.password, salt);
    let userRole = null;
    if (req.body.role === 'doctor') { userRole = req.body.role }
    else {
        userRole = 'patient'
    }

    const user = new User({
        firstName: req.body.firstName,
        lastName: req.body.lastName,
        email: req.body.email,
        password: hashPassword,
        role: userRole
    })

    try {
        const savedUser = await user.save();
        res.status(200).send('Successfully registered');
    } catch (err) {
        res.status(400).send(err);
    }
})
```

Programski kod 5.9. Postupak registriranja korisnika

Korisničko sučelje za registraciju korisnika implementirano je koristeći komponente programskog okvira React. Na klijentskoj strani kreirana je forma za unos podataka koja je povezana s API upravljačem prikazanim programskim kodom 5.9.

5.2.3. Prijava i odjava korisnika

Funkcionalnost prijave korisnika u aplikaciju ostvarena je pomoću alata *passport.js*. Za ostvarenje ove funkcionalnosti implementiran je API upravljačs HTTP metodom POST. S korisničkog sučelja šalju se podaci o e-mailu i lozinki na API */api/auh/login*. Prije no što se izvrši API, poslužitelj provjerava da korisnik nije prijavljen u aplikaciju. Ako je, API se neće izvršiti. Podaci koje je korisnik unio se nakon uspješne provjere autentikacije šalju u *authenticate()* metodu passport.js alata koja provjerava postoji li korisnik s navedenom kombinacijom e-mail adrese i lozinke u bazi podataka. Ukoliko su podaci koje je korisnik unio ispravni, aplikacija mu dopušta ulazak na sučelje za prijavljene korisnike, u slučaju da su neispravni, korisnik je preusmjeren na sučelje za prijavu kako bi se ponovno pokušao prijaviti u aplikaciju. Ako je

kombinacija lozinke i email adrese koju je korisnik unio neispravna, API će vratiti statusni kod 401 te grešku u odgovoru na zahtjev, ako operacija prođe uspješno, API vraća statusni kod 200.

Implementacija prijave korisnika prikazana je u programskom kodu 5.10.

```
router.post('/login', checkNotAuthenticated, (req, res, next) => {

    passport.authenticate('local', {
        successRedirect: '/homepage',
        failureRedirect: '/login',
        passReqToCallback: true
    }, function (err, user, info) {
        console.log(err, user)

        if (err) return res.status(401).json({ errors: err }).redi-
rect('/login');
        if (!user) return res.status(401).json({ message: info })

        req.logIn(user, function (err) {
            console.log(err)
            if (err) { res.status(400).send(err); }
            return res.status(200).redirect('/');
        })
    })(req, res, next);
});
```

Programski kod 5.10. Postupak prijave korisnika

Korisničko sučelje za prijavu korisnika implementirano je koristeći formu za unos podataka koja je povezana s API upravljačem prikazanim u programskom kodu 5.10.

Funkcionalnost odjave korisnika također je napravljena koristeći alat *passport.js*. Za ostvarenje ove funkcionalnosti implementiran je API upravljač s HTTP metodom POST koji koristi funkciju *logOut()*. Implementacija je prikazana programskim kodom 5.11.

```
router.delete('/logout', (req, res) => {
    req.logOut()
    res.redirect('/')
})
```

Programski kod 5.11. Postupak odjave korisnika

5.2.4. Prikaz vijesti o dijabetesu

Funkcionalnost prikaza i dodavanja vijesti o dijabetesu implementirana je kreiranjem API upravljača. Za dodavanje nove vijesti izrađen je API upravljač koji koristi HTTP metodu POST.

API je implementiran asinkrono radi potrebe čekanja da se podaci o vijestima zapišu u bazu podataka i prikazu u odgovoru poslužitelja u slučaju ispravnog spremanja. Podaci koji se šalju

putem API-ja su podaci o naslovu vijesti, opisu vijesti i datumu kada su podaci poslani. Ti podaci se pretvaraju u model Post.js i spremaju u bazu podataka koristeći MongoDb funkciju *save()*. Ukoliko se dogodi greška API će vratiti odgovor u JSON formatu s tekstom greške koja se dogodila. Dodavanje vijesti prikazano je u programskom kodu 5.12.

```
router.post('/', async (req, res) => {
    //res.send(req.user);
    const post = new Post({
        title: req.body.title,
        description: req.body.description,
        image: req.body.image
    });

    try {
        const savedPost = await post.save()
        res.json(savedPost);
    } catch (err) {
        res.json({ message: err })
    }
})
```

Programski kod 5.12. Postupak dodavanja nove vijesti

Osim dodavanja novih vijesti, implementiran je i API za dohvat svih dokumenata u zbirci *posts* u bazi podataka koristeći HTTP metodu GET. API je implementiran asinkrono radi potrebe čekanja podataka iz baze podataka o vijestima koje se trebaju dohvatiti i prikazati na korisničkom sučelju. Nakon dohvata svih zapisa iz baze, podaci se šalju u JSON formatu kao odgovor API-ja. Ukoliko dohvat podataka nije bilo uspješan iz bilo kojeg razloga, poslužitelj će vratiti odgovor koji sadrži opis greške koja se dogodila. Funkcionalnost dohvata svih vijesti prikazana je u programskom kodu 5.13.

```
router.get('/', async (req, res) => {

    try {
        const posts = await Post.find().limit(5);
        res.json(posts);

    } catch (err) {
        req.json({ message: err })
    }
});
```

Programski kod 5.13. Postupak dohvata svih vijesti

Korisničko sučelje za prikaz vijesti implementirano je dohvatom podataka o svim vijestima koristeći programski kod 5.12. Podaci na njemu se dohvaćaju koristeći fetch API metodu programskog okvira React. Metodi se prosljeđuje URL API upravljača te se dohvaćeni podaci

parsiraju u JSON format podataka i spremaju u React stanje (engl. *state*). Dohvat podataka s API-ja u prikazan je u programskom kodu 5.14.

```
componentDidMount() {
        fetch('/api/')
            .then(res => {
                return res.json();
            })
            .then(data => {
                this.setState({ data: data })
            })
            .catch(err => {
                console.log('err', err);
            });
    }
```

Programski kod 5.14. Dohvat podataka o vijestima na klijentskoj strani aplikacije

5.2.5. Izračun rizika od dijabetesa

Način izračuna rizika od dijabetesa opisan je u poglavlju 4.4. Kako bi funkcionalnost bila uspješno razvijena implementiran je API upravljač koji koristi HTTP metodu POST za kreiranje izračuna rizika. Implementirana je logika bodovanja koristeći jednostavna *if-elseif-else* grananja te je kreiran API */api/risk/calculate* koji dodjeljuje definirane bodovne vrijednosti svakom unesenom podatku te inkrementira varijablu *score* za određeni broj bodova. Kako bi rizik bio uspješno izračunan potrebno je unijeti podatke o godinama, spolu, masi, visini, genetskim predispozicijama za dijabetes, visokom krvnom tlaku, povećanim šećerom u krvi na nekom od ranijih pregleda, opsegu struka, zdravoj prehrani i svakodnevnoj fizičkoj aktivnosti. Bodovi zbrojeni u *score* varijabli se zatim uspoređuju s predefiniranim rangovima objašnjenima u poglavlju 4.4. se na korisničko sučelje šalje informacija o riziku od obolijevanja od dijabetesa tipa 2 ovisno o zbroju. Programski kod 5.15 prikazuje funkcionalnost izračuna rizika.

```
router.post('/calculate',(req, res) => {
    let score = null;
    console.log(req.body.physicalActivity);

    if (req.body.age < 45) score = score;
    else if (req.body.age >= 45 && req.body.age <= 54) score = score +
2;
    else if (req.body.age >= 55 && req.body.age <= 64) score = score +3;
    else score = score + 4

    if ((req.body.sex === 'female' && req.body.waistCirc < 80) ||
(req.body.sex === 'male' && req.body.waistCirc < 94)) score = score;
    else if ((req.body.sex === 'female' && req.body.waistCirc >= 80 &&
req.body.waistCirc <= 88) || (req.body.sex === 'male' && req.body.waist-
Circ >= 94 && req.body.waistCirc <= 102)) score = score + 3;
    else score = score + 4;

    if (req.body.physicalActivity === 'yes') score = score;
    else score = score + 2;

    if (req.body.healthyDiet === 'yes') score = score;
    else score = score + 1;

    if (req.body.bloodPressure === 'yes') score = score + 2;
    else score = score;

    if (req.body.bloodGlucose === 'yes') score = score + 5;
    else score = score;

    if (req.body.heritage === 'closeFamily') score = score + 5;
    else if (req.body.heritage === 'extendedFamily') score = score + 2;
    else score = score;

    let BMI = Math.round(req.body.weight / Math.pow(req.body.height, 2)
* 10000);

    if (BMI < 25) score = score;
    else if ((BMI >= 25) && (BMI <= 30)) score = score + 1;
    else score = score + 3;

    if (score < 7) res.status(200).json('Your diabetes risk score is: '
+ score + ' Low risk - 1 out of 100 people will get diabetes type 2 in
the next 10 years.');
    else if (score >= 7 && score <= 11) res.status(200).json('Your dia-
betes risk score is: ' + score + ' Slightly elevated risk - 1 out of 25
people will get diabetes type 2 in the next 10 years.');
    else if (score >= 12 && score <= 14) res.status(200).json('Your dia-
betes risk score is: ' + score + ' Moderately elevated risk - 1 out of 6
people will get diabetes type 2 in the next 10 years.');
    else if (score >= 15 && score <= 20) res.status(200).json('Your dia-
betes risk score is: ' + score + ' High risk - 1 out of 3 people will
get diabetes type 2 in the next 10 years.');
    else res.status(200).json('Your diabetes risk score is: ' + score +
' Very high risk - 1 out of 2 people will get diabetes type 2 in the
next 10 years.');
});
```

Programski kod 5.15 Postupak izračuna rizika

Korisničko sučelje za izračun rizika realizirano je kreiranjem forme povezane s API upravljačem prikazanim u programskom kodu 5.15. na kojoj korisnik može unijeti potrebne podatke. Osim forme, kreiran je i modalni prozor za prikaz rezultata upitnika koji se prikaže korisniku nakon što unese podatke i izračuna rizik.

5.2.6. Pregled pacijenata

Funkcionalnosti pregleda pacijenata imaju samo korisnici s ulogom liječnika. Funkcionalnost je ostvarena pomoću API upravljača koji koristi HTTP metodu GET za dohvaćanje pacijenata. API je implementiran asinkrono radi čekanja dohvata korisnika iz baze podataka. Prije no što se izvrši API, poslužitelj provjerava da je korisnik prijavljen u aplikaciju u ulozi liječnika. Ako nije, API se neće izvršiti. U bazi podataka se pretražuje zbirka *users* te pronađu svi korisnici kojima je parametar *patientID* jednak identifikacijskom broju trenutno prijavljenog liječnika. API zatim vraća popis s pronađenim pacijentima. Ukoliko se dogodila greška, API vraća grešku u JSON formatu. Programski kod 5.16 prikazuje funkcionalnost dohvata vlastitih pacijenata.

```
router.get('/mypatients', checkAuthenticated, checkRole(ROLE.DOCTOR),
async (req, res) => {

    try {
        const users = await User.find({ doctorId: req.user._id });
        res.json(users);

    } catch (err) {
        req.json({ message: err })
    }
});
```

Programski kod 5.16. Postupak dohvata pacijenata

Na korisničkom sučelju se popis pacijenata dohvaćen s API-ja prikazuje u tablici u kojoj su navedeni podaci o pacijentima – ime, prezime, e-mail i identifikacijski broj.

5.2.7. Unos nalaza

Korisnici s ulogom liječnika trebali bi imati mogućnost unosa nalaza za svakog od svojih pacijenata. Funkcionalnost je ostvarena pomoću API upravljača koji koristi HTTP metodu POST za unos nalaza. API je implementiran asinkrono radi čekanja dohvata korisnika iz baze podataka. Prije no što se izvrši API, poslužitelj provjerava da je korisnik prijavljen u aplikaciju u ulozi liječnika. Ako nije, API se neće izvršiti. Funkcionalnost unosa nalaza je implementirana na način da se na API šalje podatak o identifikacijskom broju pacijenta kojeg je liječnik

odabrao za unos nalaza te podaci o nalazima koje je liječnik unio – *patientId, systolicPressure, diastolicPressure, bloodGlucose, cholesterolHDL* i *cholecterolLDL*. Podaci se provjeravaju te ako su uspješno prošli provjeru, spremaju se u bazu podataka u zbirku *tests* Ako se nisu uspješno spremili u bazu podataka, API će vratiti grešku u JSON formatu i zdravstveni status. Zdravstveni status se računa na način opisan u poglavlju 4.4 te je implementiran koristeći *if-else-elseif* grananje. Ako je ijedan parametar van normalnih granica, zdravstveni status pacijenta je „*Bad*“, inače je „*Good*“. Funkcionalnost unosa nalaza prikazana je programskim kodom 5.17.

```
router.post('/test', checkAuthenticated, checkRole(ROLE.DOCTOR), async
(req, res) => {
    try {
        const exist = await User.find({ _id: req.body.patientID });
        console.log(exist);
    } catch (err) {
        console.log(req.user._id)
        return res.status(400).json({ message: err })
    }
    try {
        let health_Status = '';
        if (req.body.systolicPressure > 180 || req.body.diastolicPres-
sure > 120 || req.body.bloodGlucose > 9 || req.body.cholesteroHDL > 60
|| req.body.cholesterolLDL > 190) {
            health_Status = 'Bad';
            console.log(health_Status)
        }
        else {
            health_Status = 'Good';
            console.log(health_Status)
        }
        console.log(health_Status)

        const test = new Test({
            patientID: req.body.patientID,
            systolicPressure: req.body.systolicPressure,
            diastolicPressure: req.body.diastolicPressure,
            cholesterolHDL: req.body.cholesterolHDL,
            cholesterolLDL: req.body.cholesterolLDL,
            bloodGlucose: req.body.bloodGlucose,
            healthStatus: health_Status
        });
        console.log(test)
        const savedTest = await test.save()
        return res.status(200).send(savedTest)
    } catch (error) {
        return res.status(400).json({ message: error })
    }
})
```

Programski kod 5.17 Postupak unosa nalaza

5.2.8. Prikaz i odabir liječnika

Korisnici u ulozi pacijenta imaju mogućnost odabira liječnika ukoliko ga već nisu prethodno odabrali. Ova funkcionalnost implementirana je putem dva API upravljača. Prvi API upravljač koristi HTTP metodu GET za dohvat podataka o svim korisnicima u ulozi liječnika u bazi podataka. Prije no što se izvrši API, poslužitelj provjerava da je korisnik prijavljen u aplikaciju u ulozi pacijenta. Ako nije, API se neće izvršiti. API pretražuje zbirku *users* u bazi podataka tako što MongoDb funkciji *find()* predaje parametar koji označava ulogu korisnika. Ukoliko je pretraga baze podataka uspješna, API vraća popis korisnika koji zadovoljavaju uvjetima pretrage. Ako je pretraga bila neuspješna, API vraća grešku u JSON formatu. Funkcionalnost dohvata liječnika prikazana je programskim kodom 5.18.

```
router.get('/alldoctors', checkAuthenticated, checkRole(ROLE.PATIENT),
async (req, res) => {

    try {
        const doctors = await User.find({ role: ROLE.DOCTOR });
        res.json(doctors);

    } catch (err) {
        req.json({ message: err })
    }
});
```

Programski kod 5.18. Postupak dohvata liječnika

Drugi API upravljač koristi HTTP metodu POST za odabir liječnika. Na korisničkom sučelju se dohvaćeni liječnici prikazuju u tablici te korisnik klikom na gumb pored željenog liječnika pokreće API za odabir liječnika. API je implementiran na način da se ažuriraju podaci o pacijentu u bazi podataka na način da se prijavljenom korisniku doda identifikacijski broj odabranog liječnika kao vrijednost atributa *doctorId* koristeći MongoDb funkciju *updateOne()*. Funkciji se kao parametre predaje identifikacijski broj pacijenta kojeg je potrebno ažurirati koji je preuzet iz sesije prijavljenog korisnika te unutar *$set* parametra postavlja vrijednost koju je potrebno izmijeniti. Ukoliko je ažuriranje uspješno prošlo, API vraća ažuriranog pacijenta u JSON formatu. Implementacija ove funkcionalnosti prikazana je u programskom kodu 5.19.

```
router.post('/doctor/choose', checkAuthenticated, checkRole(ROLE.PA-
TIENT), async (req, res) => {
    try {
        const updatedDoctorInfo = await User.updateOne(
            { _id: req.user._id },
            { $set: { doctorId: req.body.doctorId } }
        )
        console.log("a", updatedDoctorInfo);
        res.json(updatedDoctorInfo);

    } catch (err) {
        return res.status(400).send({ message: err })
    }
});
```

Programski kod 5.19. Postupak odabira liječnika

5.2.9. Pregled nalaza i zdravstvenog stanja

Korisnici u ulozi pacijenta imaju mogućnost pregleda nalaza koje je za njih unio njihov odabrani liječnik. Implementirani API upravljač koristi HTTP metodu GET za dohvat nalaza prijavljenog korisnika iz baze podataka. Prije no što se izvrši API, poslužitelj provjerava da je korisnik prijavljen u aplikaciju u ulozi pacijenta. Ako nije, API se neće izvršiti. Ova funkcionalnost implementirana je na način da API */api/tests/mytests* vraća informaciju o svim nalazima prijavljenog pacijenta tako što pretražuje zbirku *tests* u bazi podataka koristeći funkciju *find()* te ih sortira po datumu kreiranja od najnovijeg prema najstarijem. Sortiranje se vrši korištenjem funkcije *sort()* koja ka parametar prima atribut po kojemu se sortira i broj koji označava kakva je vrsta sortiranja – uzlazno ili silazno. Pretraživanje baze podataka funkcionira na način da se filtriraju svi nalazi koji kao atribut *patientId* imaju vrijednost identifikacijskog broja trenutno prijavljenog pacijenta koji je preuzet iz sesije. Ukoliko se dogodila greška, ista će biti vraćena u JSON formatu kao odgovor API-ja. Funkcionalnost prikaza nalaza prikazana je u programskom kodu 5.20.

```
router.get('/mytests', checkAuthenticated, checkRole(ROLE.PATIENT),
async (req, res) => {
    try {
        const tests = await Test.find({ patientID: req.user._id
}).sort({ "date": -1 });
        res.json(tests);
    } catch (err) {
        res.json({ message: err })
    }
});
```

Programski kod 5.20. Postupak dohvata pacijentovih nalaza

Na korisničkom sučelju se dohvaćeni nalazi prikazuju u tablici te korisnik klikom na gumb pored željenog nalaza otvara modalni prozor za unos podataka s pametne narukvice.

5.2.10. Unos podataka s pametne narukvice i izračun zdravstvenog stanja

Pacijent za svaki nalaz koji mu je odabrani liječnik unio ima mogućnost unosa podataka s pametne narukvice u svrhu kompletnog pregleda zdravstvenog stanja. API upravljač koristi HTTP metodu POST za unos podataka s pametne narukvice izračun zdravstvenog stanja. Prije no što se izvrši API, poslužitelj provjerava da je korisnik prijavljen u aplikaciju u ulozi pacijenta. Ako nije, API se neće izvršiti. Nakon unosa podataka s pametne narukvice, podaci odabranog nalaza i podaci s pametne narukvice rangiraju se na način opisan u poglavlju 4.4. Kreiran je JSON objekt *message* koji služi za skup poruka koje se šalju kao rezultat izračuna zdravstvenog stanja. Izračun se provodi koristeći jednostavna *if-elseif-else* grananja gdje svaka od grana programa doda drugačiju poruku u *message* varijablu. Nakon što je status svakog od parametara izračunat, *message* varijabla se šalje kao odgovor API-ja. Ukoliko je došlo do pogreške, API vraća grešku u JSON formatu. Implementacija funkcionalnosti izračuna zdravstvenog stanja prikazana je u programskom kodu 5.21.

```
if (req.body.systolicPressure < 120) message.sysbloodPressure = "Your
blood pressure is normal."
    else if (req.body.systolicPressure >= 120 &&
req.body.systolicPressure < 130) message.sysbloodPressure = "Your
systolic blood pressure is elevated."
    else if (req.body.systolicPressure >= 130 &&
req.body.systolicPressure < 140) message.sysbloodPressure = "Your
systolic blood pressure is high. You could suffer from Hypertension
stage 1."
    else if (req.body.systolicPressure >= 140 &&
req.body.systolicPressure < 180) message.sysbloodPressure = "Your
systolic blood pressure is high. You could suffer from Hypertension
stage 2."
    else message.sysbloodPressure = 'You may be in a hypertensive
crisis. Consult your doctor immediately'
```

Programski kod 5.21 Primjer rangiranja sistoličkog krvnog tlaka

Sve funkcionalnosti definirane u poglavlju 4 su implementirane na klijentskoj i poslužiteljskoj strani te će način rada aplikacije iz perspektive korisnika biti opisan u sljedećem poglavlju.

6. NAČIN RADA APLIKACIJE

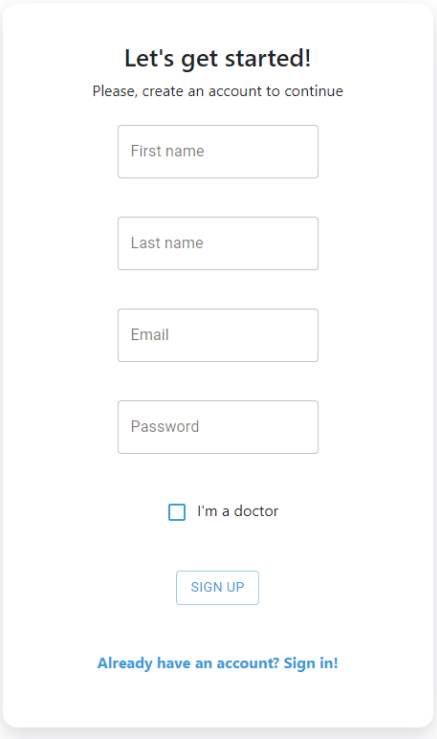
Kao što je već spomenuto u poglavlju 4.1, aplikacija za potporu pacijentima oboljelima od dijabetesa sastoji se od javnog portala i sučelja prijavljenog korisnika. Javni portal dostupan je prijavljenim i neprijavljenim korisnicima i sastoji se od sljedećih sučelja:

1. Sučelje za pregled vijesti
2. Sučelje za izračun rizika
3. Sučelje za registraciju
4. Sučelje za prijavu

Sučelje prijavljenog korisnika dodatno se dijeli na sučelje korisnika s ulogom liječnika i sučelje korisnika s ulogom doktora. Korisnik s ulogom liječnika vidi sva sučelja od kojih se sastoji javni portal uz dodatak sučelja za pregled pacijenata. Korisnik s ulogom pacijenta vidi sva sučelja javnog portala uz dodatna sučelja za odabir liječnika i pregled nalaza.

6.1. Postupak registriranja korisnika

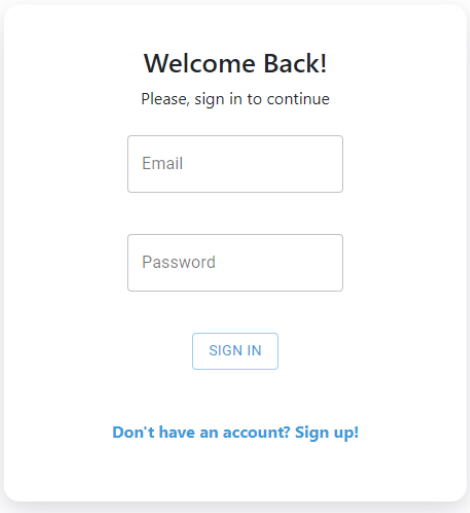
Kako bi korisnik mogao početi koristiti dio portala za prijavljene korisnike, potrebno je da se registrira. Klikom na "Login/Register" u navigaciji aplikacije dolazi do sučelja za prijavu u kojemu postoji poveznica na sučelje za registraciju. Na sučelju za registraciju korisnik mora popuniti polja sa svojim osobnim podacima – ime, prezime, e-mail, lozinka te oznaka želi li se registrirati kao liječnik ili kao pacijent. Ukoliko je *checkbox "I'm a doctor*" označen, korisnik će u sustav biti registriran kao liječnik, ukoliko nije, korisnik će biti registriran kao pacijent. Ako korisnik nije popunio sva obvezna polja, registracija se smatra neuspješnom te korisnik nije registriran. Sučelje za registraciju prikazano je na slici 6.1.



Slika 6.1 Izgled forme za unos podataka za registraciju

6.2. Postupak prijave korisnika

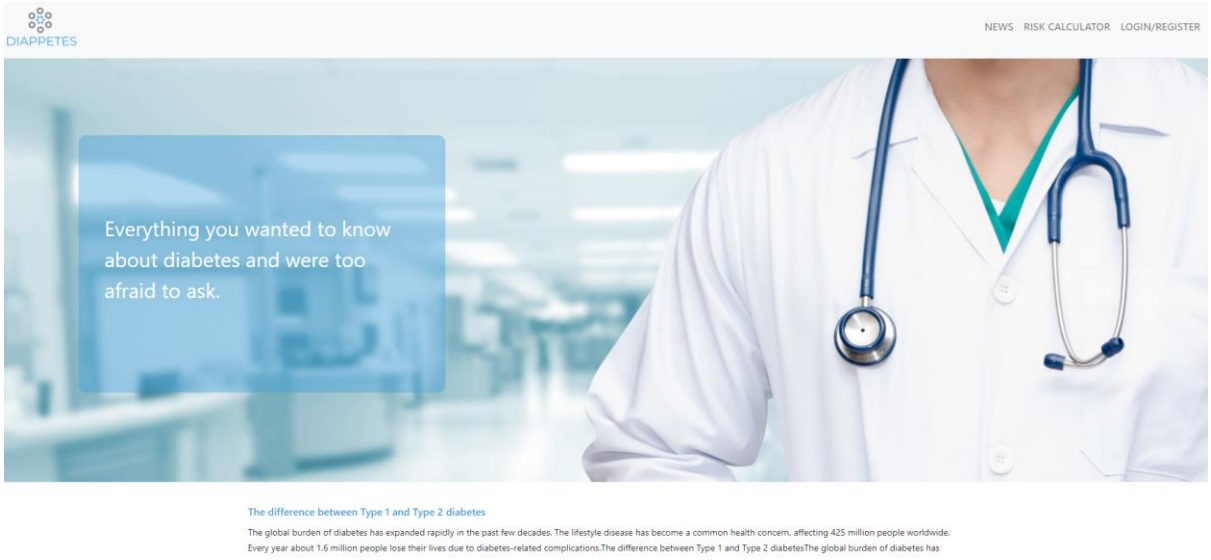
Nakon registracije, korisnik ima mogućnost prijaviti se u sustav. Do sučelja za prijavu dolazi se klikom na poveznicu "*Login/Register*" u navigaciji aplikacije. Nakon toga, korisniku se prikazuje forma s poljima za unos e-mail adrese i lozinke. Ukoliko su unesena e-mail adresa i lozinka ispravni, korisnik je prijavljen u sustav te mu se prikazuju dodatna sučelja ovisno o korisničkoj ulozi koju posjeduje. Sučelje za prijavu korisnika prikazano je na slici 6.2.



Slika 6.2 Izgled forme za unos podataka za prijavu

6.3. Sučelje za pregled vijesti

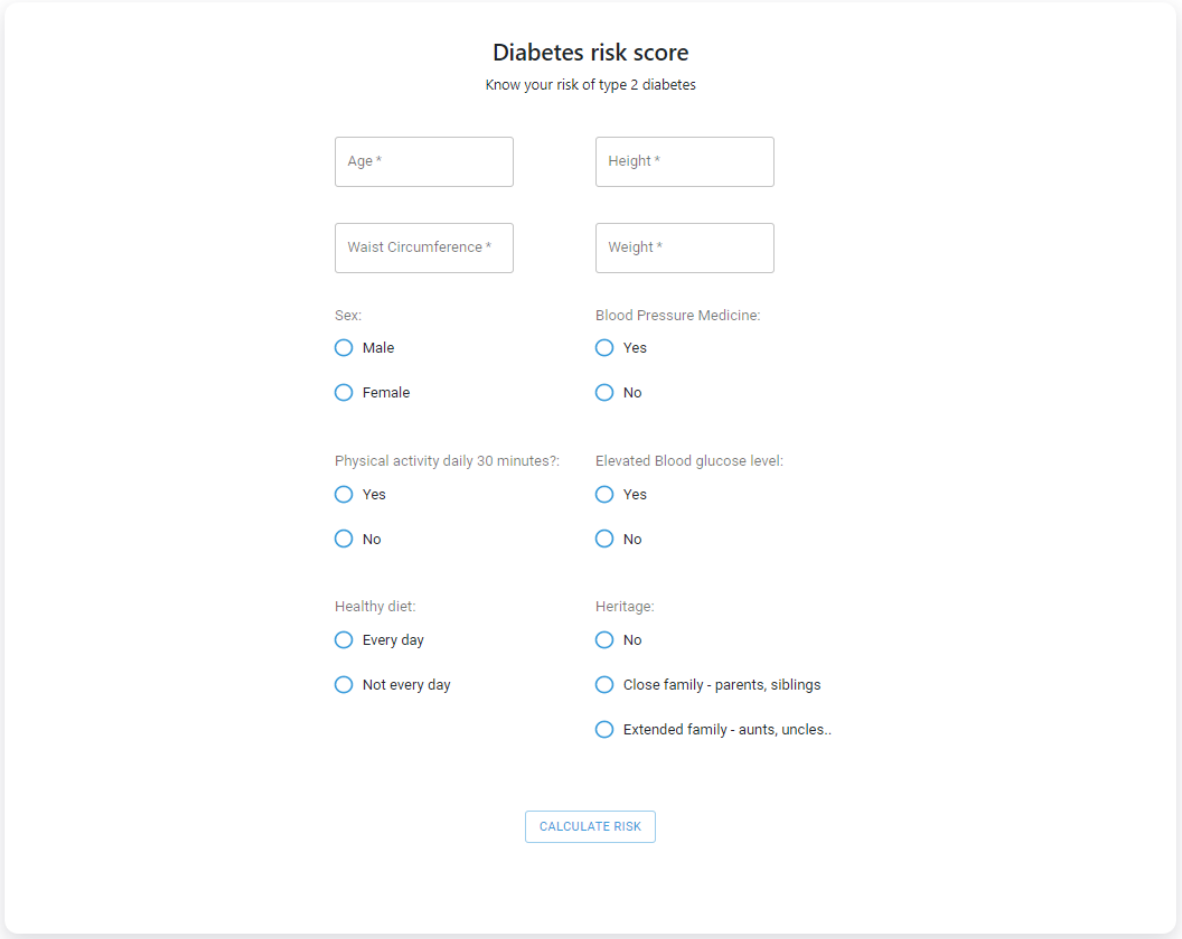
Sučelje za pregled vijesti dostupno je i prijavljenim i neprijavljenim korisnicima. Na tom sučelju nalaze se kratke obavijest o novostima u prevenciji dijabetesa, načinima za lakši život s tom bolešću i svim ostalim relevantnim informacijama. Do ovog sučelja dolazi se odabirom poveznice "*News*" u navigaciji aplikacije. Izgled sučelja za pregled vijesti prikazan je na slici 6.3.



Slika 6.3 Sučelje za pregled vijesti

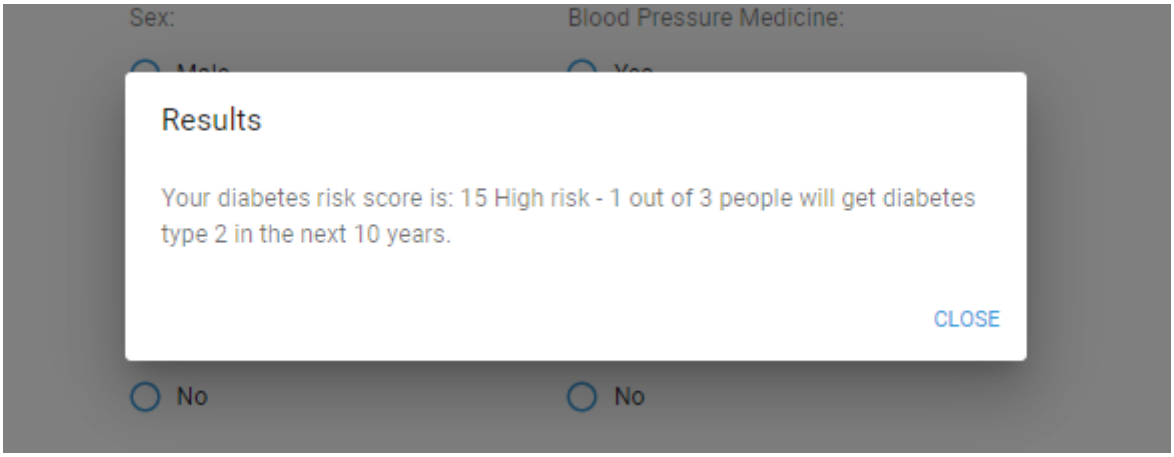
6.4. Sučelje za izračun rizika

Sučelje za izračun rizika dostupno je i prijavljenim i neprijavljenim korisnicima. Do ovog sučelja dolazi se odabirom poveznice "*Risk calculator*" u navigaciji aplikacije. Dolaskom na sučelje, korisnik ima pregled forme za unos podataka kako bi si mogao izračunati rizik od dijabetesa tipa 2 u narednih 10 godina. Korisnik mora popuniti sva polja kako bi mu se uspješno izračunao rizik. Izgled sučelja za izračun rizika prikazan je na slici 6.4.



Slika 6.4 Izgled forme za izračun rizika

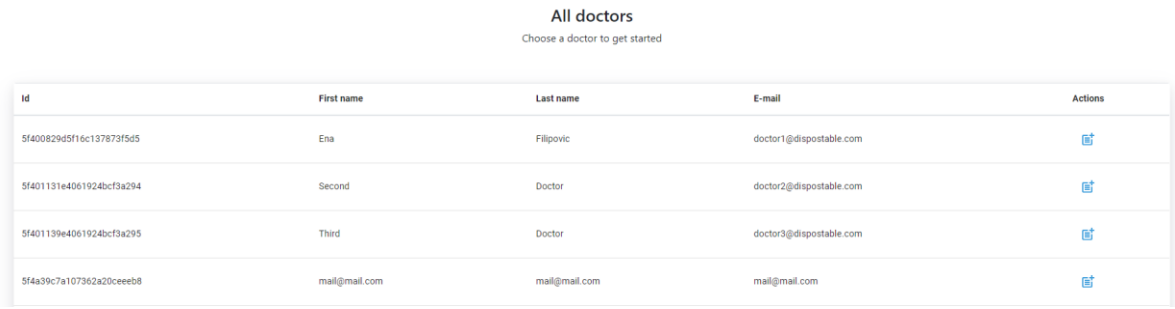
Nakon što korisnik popuni sva polja na formi i klikne na gumb "*Calculate risk*", pojavit će mu se modalni prozor s rezultatima upitnika. Izgled modalnog prozora prikazan je na slici 6.5.



Slika 6.5 Modalni prozor s rezultatima izračuna rizika

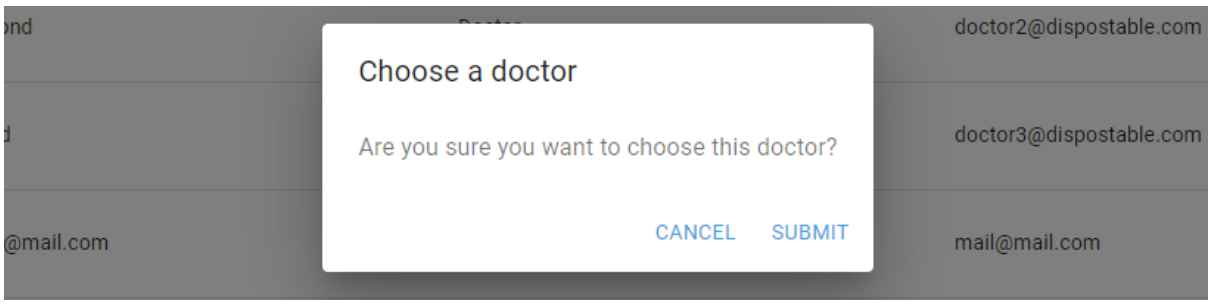
6.5. Sučelje za odabir liječnika

Sučelju za odabir liječnika imaju pristup samo korisnici s ulogom pacijenta koji prethodno nisu odabrali liječnika. Do sučelja za odabir liječnika dolazi se klikom na poveznicu "*Choose doctor*" u navigaciji aplikacije. Dolaskom na ovo sučelje, korisniku se prikaže tablica s popisom svih liječnika u aplikaciji te podataka o tim liječnicima – ime, prezime, e-mail adresa te identifikacijski broj liječnika. Sučelje za odabir liječnika prikazano je na slici 6.6.



Slika 6.6 Sučelje za odabir liječnika

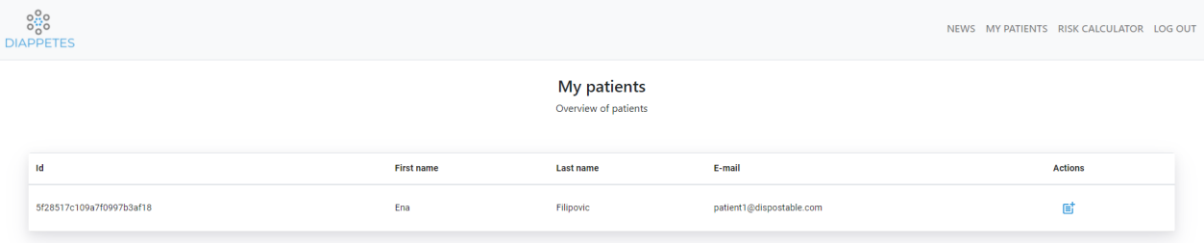
Liječnik se odabire klikom na ikonu pored željenog liječnika. Nakon toga, korisniku se otvara modalni prozor na kojemu je potrebno potvrditi da je siguran da želi odabrati baš tog liječnika. Modalni prozor za potvrdu odabira liječnika prikazan je na slici 6.7.



Slika 6.7 Modalni prozor za potvrdu odabira liječnika

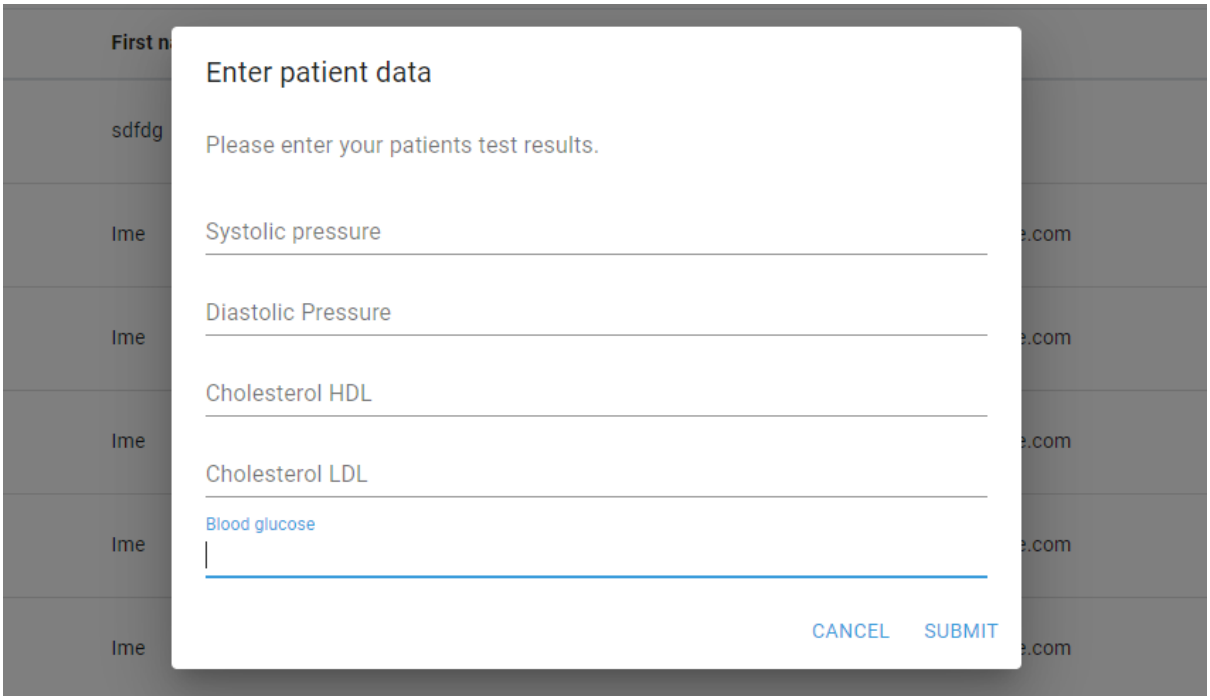
6.6. Sučelje za pregled pacijenata i unos nalaza

Sučelju za pregled pacijenata pristup imaju samo korisnici s ulogom liječnika. Do ovog sučelja dolazi se odabirom poveznice "*My patients*" u navigaciji aplikacije. Na ovom sučelju korisnik ima pregled vlastitih pacijenata u tabličnom obliku. U tablici su prikazani sljedeći podaci o pacijentima: ime, prezime, identifikacijski broj i e-mail adresa. Sučelje za pregled pacijenata prikazano je na slici 6.8.



Slika 6.8 Sučelje za pregled pacijenata

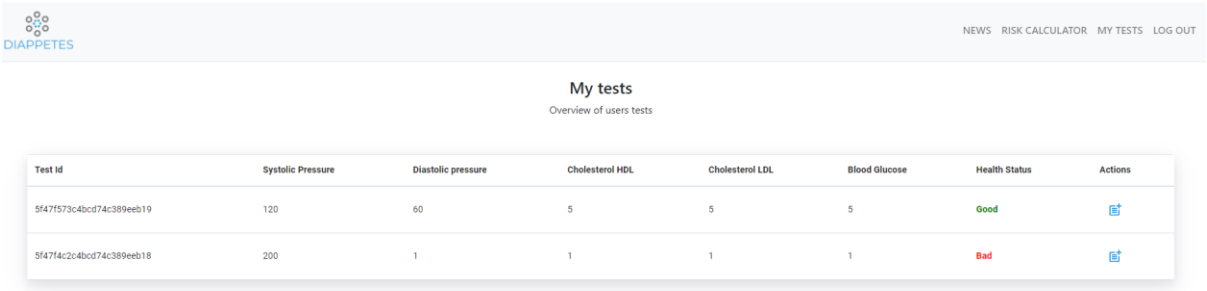
Liječnik svakom pacijentu u tablici može upisati novi nalaz klikom na gumb u stupcu "*Actions*". Klikom na taj gumb, otvara se modalni prozor u kojemu je moguće upisati vrijednosti sistoličkog i dijastoličkog tlaka, razine glukoze u krvi te HDL i LDL kolesterola. Nakon unosa podataka, nalazi će biti vidljivi pacijentu na sučelju za pregled nalaza. Izgled modalnog prozora za unos nalaza prikazan je na slici 6.9.



Slika 6.9. Modalni prozor za unos nalaza

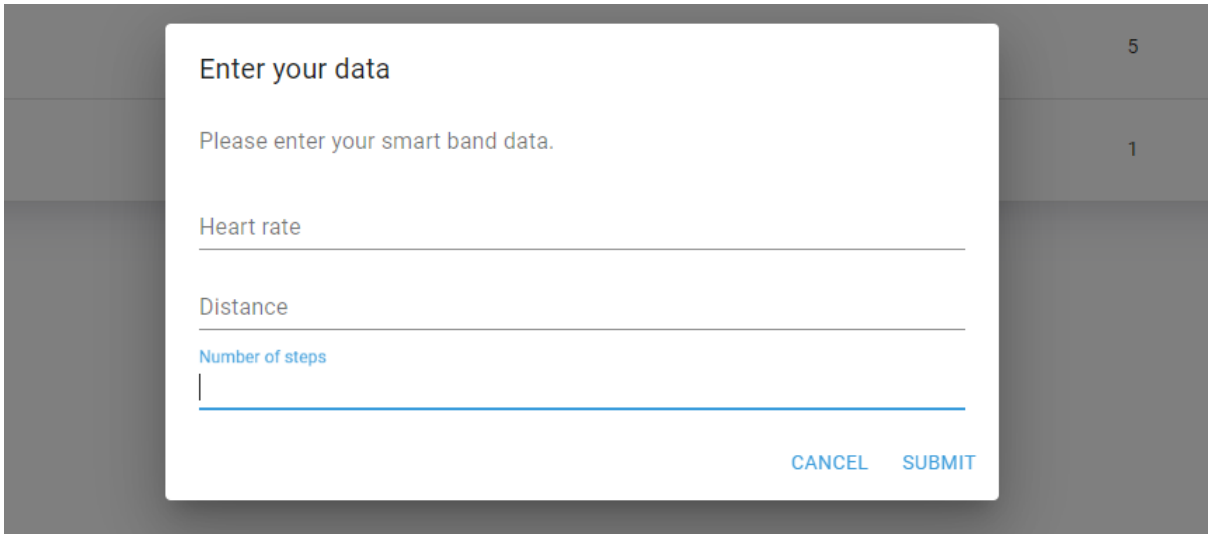
6.7. Sučelje za pregled nalaza

Ovom sučelju pristup imaju samo korisnici s ulogom pacijenta. Do ovog sučelja dolazi se odabirom poveznice *"My tests"* u navigaciji aplikacije. Dolaskom na ovo sučelje, pacijenti imaju pregled svih svojih nalaza poredanih od najnovijih do najstarijih. Za svaki nalaz pacijenti odmah mogu vidjeti kakav im je zdravstveni status u stupcu *"Health status*". Izgled sučelja za pregled nalaza nalazi se na slici 6.10.



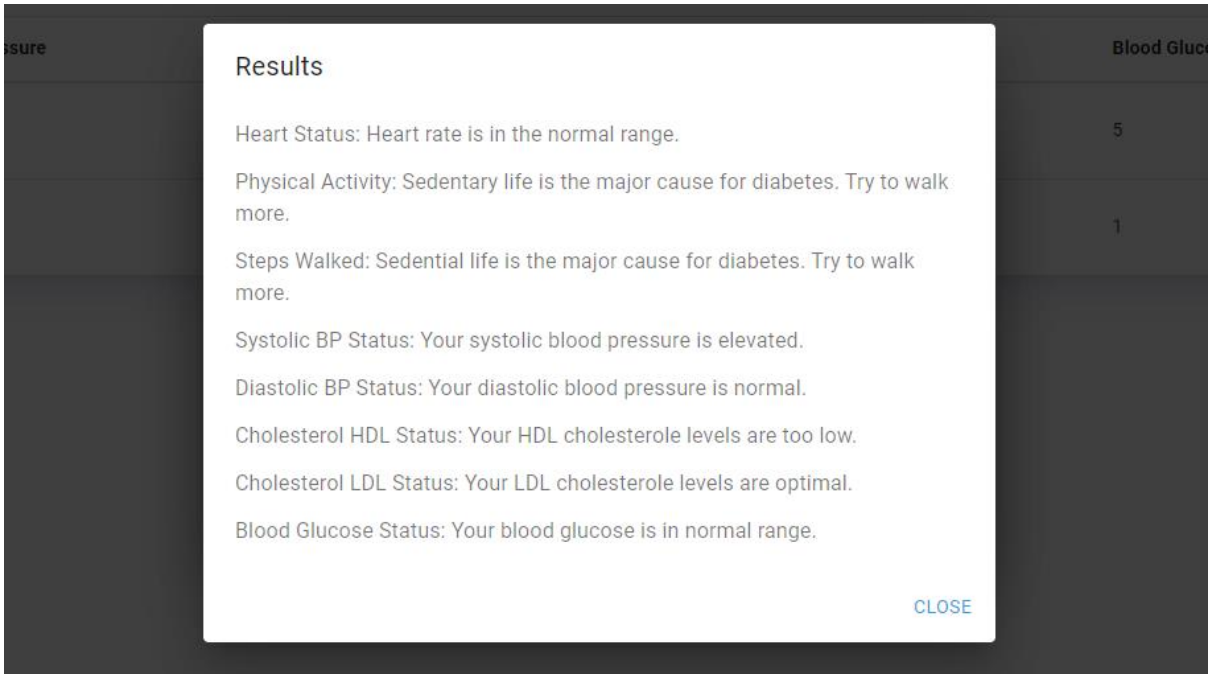
Slika 6.10 Sučelje za pregled nalaza

Pacijent za svaki nalaz u tablici ima mogućnost unosa podataka s pametne narukvice za taj dan klikom na ikonu u stupcu "*Actions*". Klikom na ikonu otvara se modalni prozor za unos podataka s pametne narukvice – prosječnog broja otkucaja srca, prehodane udaljenosti i broja napravljenih koraka u danu. Izgled modalnog prozora prikazan je na slici 6.11.



Slika 6.11 Modalni prozor za unos podataka s pametne narukvice

Nakon unosa podataka i klika na gumb *"Submit",* korisniku se prikazuje modalni prozor s porukama o statusu svakog unesenog parametra. Izgled modalnog prozora sa zdravstvenim statusom prikazan je na slici 6.12.



Slika 6.12 Modalni prozor s prikazom zdravstvenog stanja

U sljedećem poglavlju prikazat će se provedeni postupci osiguravanja kvalitete web aplikacije. Osim toga, prikazat će se i pregled uspješnosti implementacije u formatu izvještaja s testiranja.

7. TESTIRANJE APLIKACIJE

Testiranje programske podrške skupni je naziv za sve aktivnosti koje osiguravaju da software odgovara zadanim specifikacijama. Osobe zadužene za testiranje pronalaze greške u sustavu – bilo to greške u generalnom smislu ili greške vezane uz poslovni slučaj koji sustav rješava. Glavne zadaće testiranja su utvrditi da:

- Sustav odgovara definiranim zahtjevima
- Sustav odgovarajuće reagira na svaku vrstu ulaznih podataka
- Sustav funkcionalno obavlja zadaće u razumnom vremenskom roku
- Sustav je stabilan u svim zahtijevanim okolinama

Ovisno o dijelu sustava koji se želi testirati, testiranje se dijeli na dvije vrste:

1. Funkcionalno testiranje je vrsta testiranja koja provjerava odgovara li sustav funkcionalnoj specifikaciji. Svrha ovakve vrste testiranja je potvrditi ispravan rad svake funkcionalnosti koju sustav mora imati. Potrebno je koristeći razne ulazne podatke, validirati izlazne podatke s definiranim kriterijima (engl. *acceptance criteria*). Može se izvršavati ručno ili automatiziranim testovima.
2. Nefunkcionalno testiranje provjerava koliko je sustav optimiziran. Ovakvom vrstom testiranja provjeravaju se performanse sustava, skalabilnost, sigurnost, te ponašanje aplikacije kada je korištena od strane velikog broja korisnika. Većinom se izvršava koristeći razne alate.

Proces testiranja obuhvaća mnoge aktivnosti u razvojnom timu. Počinje planiranjem u kojemu se dogovara kakva vrsta testiranja će se raditi, što je sve u engl. *scopeu*, na koji način će se izvršavati testovi te koje kriterije je potrebno ispuniti kako bi se projekt mogao prozvati testiranim. Nakon planiranja dolazi faza dizajna testova u kojoj se kreira testna specifikacija u kojoj se raspisuju testni scenariji s koracima koje testeri izvršavaju kako bi se potvrdilo da neka funkcionalnost sustava ispravno radi. Po završetku planiranja počinje faza izvršavanja u kojoj testeri po testnoj specifikaciji provode planiranje testne slučajeve i testiraju samu aplikaciju. Probleme prijavljuju razvojnim inženjerima koji ih zatim popravljaju i vraćaju na provjeru. Na kraju testiranja pišu se testni izvještaji te testna dokumentacija kako bi bio lakše popratiti koji dijelovi sustava ispravno funkcioniraju.

Osim podjele na funkcionalno i nefunkcionalno, testiranje možemo podijeliti po načinu izvršavanja testova na automatizirano i ručno.

7.1. Vrste testiranja

Kao što je već spomenuto, testiranje se dijeli na dva tipa ovisno o načinu izvršavanja testnih slučajeva – automatizirano i ručno testiranje.

7.1.1. Ručno testiranje

Ručno testiranje podrazumijeva izvršavanje testnih scenarija ručno, bez korištenja ikakvih alata. Test inženjer ručno oponaša akcije koje bi krajnji korisnik stranice izvršavao. Najprimitivnija je vrsta od svih vrsta testiranja. Ručno testiranje osigurava da se sustav ponaša sukladno zadanoj testnoj specifikaciji bez grešaka te da doživljaj korisnika (engl. *user experience)* bude pozitivan. Ovakva vrsta testiranja izvršava se na svakom novom sustavu prije automatizacije. Glavna zadaća mu je utvrditi da ulazni podaci odgovaraju očekivanim izlaznim podacima. Kako bi se ispravno provelo, potrebno je utrošiti puno više vremena no što bi trebalo kod izvršavanja automatiziranih testova.

7.1.2. Automatizirano testiranje

Automatizirano testiranje je proces testiranja u kojemu se koriste alati kako bi se utvrdilo da radi na ispravan način. Skuplje je i brže od ručnog testiranja. Kako bi se test automatizirao potrebno je napisati programski kod koji je modeliran po raspisanim testnim scenarijima kod planiranja testiranja. Osim provjeravanja funkcionalnosti, pomoću automatizacije testira se i sljedeće:

- Sigurnost sustava – engl. *Security testing*
- Ponašanje sustava pod velikim opterećenjem korisnika – engl. *Load testing*
- Ponašanje sustava u stresnim uvjetima – engl. *Stress testing*
- Izgled sustava – engl. *UI testing*

Nakon pisanja testova, test inženjer iste održava u slučaju promjena u aplikaciji, no umjesto njega, testove izvršava računalo. Cilj automatiziranog testiranja je smanjiti broj testnih slučajeva koji se izvršavaju ručno, no ne i eliminirati ih. U svrhu testiranja ove aplikacije koristit će se kombinacija ručnog i automatiziranog testiranja.

7.2. Korištene tehnologije za testiranje programske podrške

Postoje mnogi alati za automatizaciju testiranja no neki od najpoznatijih su:

- Ranorex Studio

Ranorex Studio je razvojni okvir za testiranje korisničkog sučelja. Koristi se za testiranje desktop, web i mobilnih aplikacija.

- Selenium WebDriver

Selenium WebDriver je razvojni okvir koji omogućuje automatizirano testiranje. Podržava testiranje u različitim preglednicima te može izvršavati istovremeno testove u nekoliko preglednika i na nekoliko operacijskih sustava. Selenum WebDriver kod moguće je pisati u mnogo programskih jezika, primjerice C# i Java.

- Katalon Studio

Katalon Studio je alat koji služi olakšavanju automatizacije testiranja. Sadrži snimač koji po poduzetim akcijama test inženjera kreira programski kod koji testira aplikaciju. Osim snimača, moguće je pisati i programski kod.

- Cypress

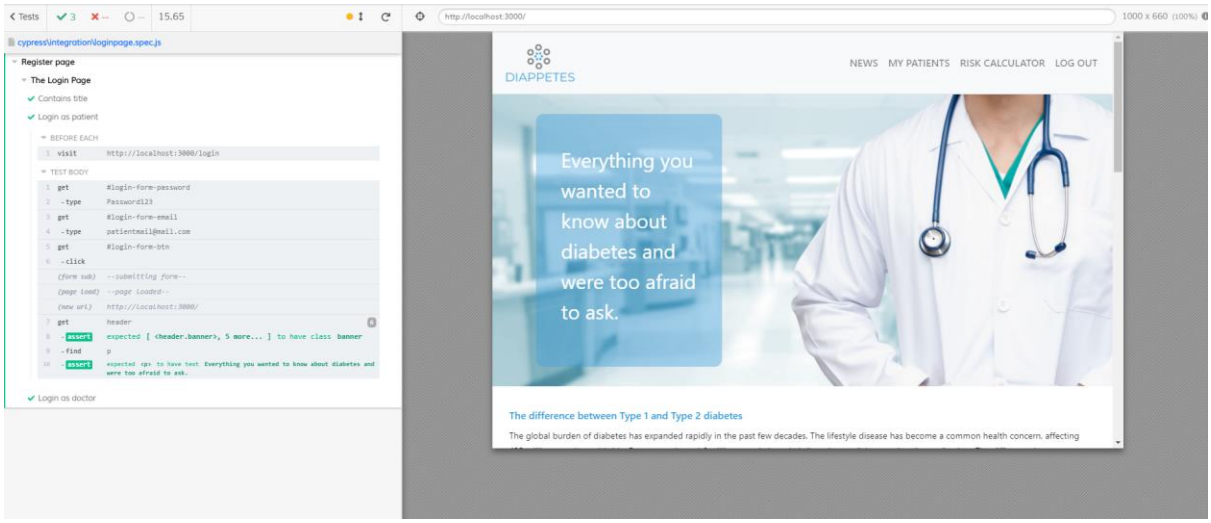
U svrhu automatskog funkcionalnog testiranja ove aplikacije kao optimalan se odabrao Cypress, a u svrhu testiranja API-ja Postman.

7.2.1. Cypress

Cypress je programski okvir jezika JavaScript koji služi za testiranje web rješenja. Brz je, pouzdan i jednostavan za korištenje te se izvršava u browseru. Pomoću Cypress-a moguće je pisati testove pokretati ih te generirati izvještaje s testiranja. Podržava pisanje više vrsta testova:

- Funkcionalno testiranje
- Unit testovi
- Testovi integracije
- API testiranje

Prikaz pokretanja testa u Cypress-u prikazan je na slici 7.1.



Slika 7.1 Izgled pokretača testova Cypress alata

7.2.2. Postman

Postman je najpoznatiji alat za testiranje API-ja. Pomoću njega je moguće stvarati zbirke testova API-ja, testirati i dokumentirati API. Testiranje API-ja provodi slanjem vlastitih HTTP/S zahtjeva prema poslužitelju te primanja odgovora od poslužitelja. Za slanje HTTP zahtjeva potrebno je unijeti URL zahtjeva, metodu koja se koristi te podatke koji se šalju putem API-ja. Ukoliko je potrebno, na API zahtjeve je moguće dodati kolačiće, tokene i drugu raznu autentikaciju.

7.3. Testni slučajevi

Testni slučajevi su skupina akcija, ulaznih podataka i procedura koje provjeravaju rad bilo koje funkcionalnosti. Osnova pisanja testnih slučajeva je definiranje koraka koje je potrebno poduzeti u sustavu i očekivanih rezultata za te korake. Testni slučajevi za ovu aplikaciju definirani su po uzora na zahtjeve na aplikaciju opisane u poglavlju 4.1. Svaki od testova provjerava jedan dio funkcionalnosti te pokazuje u kojoj mjeri je ta funkcionalnost pravilno implementirana. Kako bi se provjerile funkcionalnosti aplikacije raspisani su testni scenariji navedeni u tablici 7.1.

Tablica 7.1 Popis testnih slučajeva za funkcionalno testiranje web aplikacije

Rbr.	Naziv testnog scenarija	Dio sustava	Koraci	Očekivani rezultat
1	Korisnik se može prijaviti u sustav s ispravnim podacima	Sučelje za prijavu	1. Odabrati poveznicu “Login/Register” 2. Upisati ispravnu e-mail adresu i lozinku	Korisnik je prijavljen u sustav i preusmjeren na sučelje za pregled vijesti.

			3. Odabrati gumb “Login”	
2	Korisnik se može registrirati u sustav kao pacijent	Sučelje za registraciju	1. Odabrati poveznicu “Login/Register” 2. Odabrati poveznicu za registraciju 3. Upisati podatke 3. Odabrati gumb “Register”	Korisnik je uspješno registriran u sustav.
3	Korisnik se može registrirati u sustav kao liječnik	Sučelje za registraciju	1. Odabrati poveznicu “Login/Register” 2. Odabrati poveznicu za registraciju 3. Upisati podatke 3. Odabrati gumb “Register”	Korisnik je uspješno registriran u sustav.
4.	Korisnik ima pregled vijesti o dijabetesu	Sučelje za pregled vijesti o dijabetesu	1. Odabrati poveznicu “News”	Korisnik ima pregled vijesti o dijabetesu.
5.	Korisnik u ulozi liječnika ima pregled vlastitih pacijenata	Sučelje za pregled pacijenata	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom liječnika 3. Odabrati poveznicu “My patients”	Korisnik ima pregled vlastitih pacijenata.
6.	Korisnik u ulozi liječnika može unijeti nalaz za pacijenta	Sučelje za pregled pacijenata	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom liječnika 3. Odabrati poveznicu “My patients” 4. Odabrati gumb za otvaranje modalnog pored pacijenta kojemu se želi unijeti nalaz	Korisnik može unijeti podatke na modalni prozor za unos nalaza.
7.	Korisnik u ulozi pacijenta može odabrati liječnika	Sučelje za odabir liječnika pacijenata	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom pacijenta koji nema liječnika 3. Odabrati poveznicu “Choose doctor” 4. Odabrati gumb za odabir liječnika	Korisnik odabire liječnika.

			pored željenog liječnika	
8.	Korisnik u ulozi pacijenta koji već ima liječnika nema pristup sučelju za odabir liječnika	Sučelje za odabir liječnika	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom pacijenta koji ima liječnika	Korisnik ne vidi poveznicu “Choose doctor” u navigaciji aplikacije.
9.	Korisnik u ulozi pacijenta ima pregled svojih nalaza	Sučelje za pregled nalaza	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom pacijenta 3. Odabrati poveznicu “My tests”	Korisniku ima pregled vlastitih nalaza u tabličnom formatu.
10.	Korisnik u ulozi pacijenta ne vidi nalaze drugih pacijenata	Sučelje za pregled nalaza	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom pacijenta 3. Odabrati poveznicu “My Tests”	Korisnik ne vidi nalaze drugih pacijenata.
11.	Korisnik u ulozi pacijenta može unijeti podatke s pametne narukvice	Sučelje za pregled nalaza	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom pacijenta 3. Odabrati poveznicu “My tests” 4. Odabrati gumb za unos podataka s pametne narukvice pored željenog nalaza	Korisniku je prikazan modalni prozor za unos podataka.
12.	Korisnik u ulozi pacijenta može vidjeti svoj zdravstveni status	Sučelje za pregled nalaza	1. Odabrati poveznicu “Login/Register” 2. Prijaviti se s korisničkim računom pacijenta 3. Odabrati poveznicu “My tests” 4. Odabrati gumb za unos podataka s pametne narukvice pored željenog nalaza 5. Unijeti podatke 6. Odabrati gumb “Submit”	Korisnik u ulozi pacijenta nakon unosa podataka s pametne narukvice ima pregled svog zdravstvenog stanja.

13.	Korisnik može izračunati rizik od obolijevanja dijabetesom tipa 2	Sučelje za izračun rizika	1. Odabrati poveznicu “Risk calculator” 2. Popuniti upitnik za izračun rizika 3. Kliknuti gumb “Calculate risk”	Korisniku je prikazan modalni prozor s rezultatima upitnika.

7.4. Provođenje automatiziranog testiranja

7.4.1 Instalacija alata Cypress

Kako bi se mogao koristiti alat Cypress za testiranje web aplikacije, unutar programskog rješenja izrađen je poseban direktorij koji će sadržavati testove. Nakon kreiranja direktorija, potrebno je navigirati do tog direktorija koristeći komandnu liniju i naredbu *cd*. Unutar direktorija je potrebno instalirati Cypress alat koristeći *npm* upravitelj paketa za JavaScript i naredbu *npm install cypress*. Kako bi se olakšalo samo pokretanje Cypress-a, u datoteku *cypress.json* dodana je skripta za otvaranje Cypress alata prikazana na slici 7.2.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "cypress:open": "cypress open",
  "cypress:report": "generate-mochawesome-report",
  "cy:run": "cypress run"
},
```

Slika 7.2. Skripta za pokretanje Cypress alata

Pomoću dodane skripte Cypress se otvara u terminalu koristeći naredbu *npm run cypress:open* te se otvara Cypress izvršitelj testova (engl. *runner*). Testove je moguće izvršavati i pozadinski koristeći naredbu *npm run cy:run* koja rezultate testova sprema u JSON formatu. Rezultati u JSON formatu se mogu prikazati na korisniku čitljiv način koristeći naredbu *npm run cypress:report* koja generira izvještaj s testiranja u HTML obliku.

7.4.2 Pisanje testova u alatu Cypress

Automatizirano testiranje implementirano je na način da testni alat Cypress izvršava scenarije definirane u poglavlju 7.3. Kako bi se napisao test, potrebno je izraditi datoteku u direktoriju *integration* koja ima naziv *naziv_sučelja.spec.js* kako bi Cypress prepoznao da treba izvršiti tu datoteku. Prikaz strukture direktorija u kojemu se nalaze automatizirani testovi prikazan je na slici 7.3.

```
|   cypress.json
|   package.json
+---cypress
|   +---integration
|   |       choosedoctor.spec.js
|   |       loginpage.spec.js
|   |       mypatients.spec.js
|   |       mytests.spec.js
|   |       newspage.spec.js
|   |       registerpage.spec.js|   |
|   +---plugins
|   |       index.js|   |
|   +---reports
|   |   \---html
|   |       |   mochawesome.html|   |       |
|   +---results
|   |   \---json
|   |           mochawesome.json
|   +---support
|   |       commands.js
|   |       index.js|   |
```

Slika 7.3 Prikaz strukture direktorija *tests*

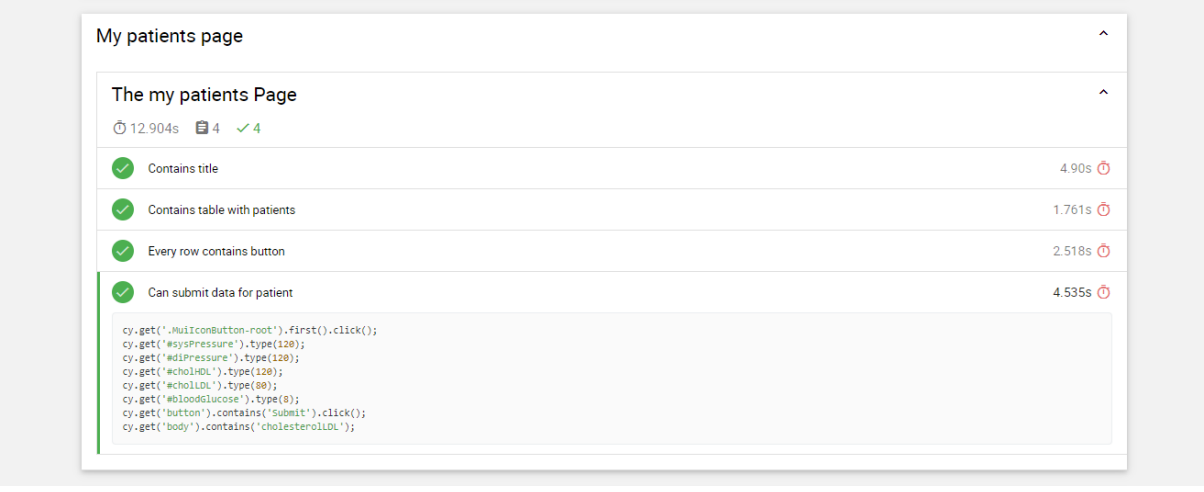
Unutar *spec* datoteke, pisanje testa započinje se dodavanjem *describe* funkcije koja opisuje funkcionalnost koja se testira. Unutar te funkcije pišu se testovi po raspisanim testnim slučajevima. Svaki testni slučaj piše se unutar funkcije *it* u kojoj se dohvaćaju elementi DOM-a i definiraju koraci koje će testni alat poduzeti pri testiranju aplikacije – upisati tekst, kliknuti na gumb i sl. Osim toga, u metodi *beforeEach()* moguće je definirati testne korake koji će se izvršiti prije svakog testa unutar *describe* bloka testova. Izgled testa napisanog u Cypress razvojnom okviru koji provjerava funkcionalnosti za odabir liječnika prikazan je na programskom kodu 7.1. U testu se dohvaćaju elementi pomoću identifikatora – klase ili id parametra elementa te se provjerava postoji li naslov sučelja te sadrži li ispravan tekst, postoji li tablica koja sadrži sve liječnike u aplikaciji te postoje li gumbi za odabir liječnika u svakom retku tablice.

```
context('Choose doctor page', () => {
    beforeEach(() => {
        let randMail = Math.floor(Math.random() * 89999 + 10000);
        cy.register('Ime', 'Prezime', 'patient' + randMail +
'@dispostable.com', 'Password123', false)
        cy.login('patient' + randMail + '@dispostable.com',
'Password123')
        cy.visit('http://localhost:3000/choosedoctor')
    })

    describe('The choose doctor Page', () => {
        it('Contains title', () => {
            cy.get('h1')
                .should('have.text', 'All doctors')
        })

        it('Contains table with tests', () => {
            cy.get('#patienttable')

        })

        it('Every row contains button', () => {
            cy.get('tr').each(() => {
                cy.get('.MuiIconButton-root')
            })
        })

        it('Can choose doctor', () => {
            cy.get('.MuiIconButton-root').first().click()
            cy.get('button').contains('Submit').click()
        })
    })

});
```

Programski kod 7.1. Primjer testa u Cypress alatu

7.4.3 Rezultati automatiziranog testiranja

Nakon izvršavanja automatiziranih testova, generira se izvještaj s testiranja. Izgled izvještaja s testiranja prikazan je na slici 7.4.



Slika 7.4. Izvještaj s testiranja u Cypress alatu

Izvještaj s testiranja prikazuje broj testnih skupina, broj provedenih testova, njihovo vrijeme izvršavanja te uspješnost samog izvršavanja testova. Za svaki od testova posebno moguće je vidjeti sve korake koji su u testu izvršeni te koliko je automatiziranom browseru trebalo da ih izvrši. U slučaju da se test nije uspješno izvršio, na izvještaju je vidljivo u kojem koraku je postojao problem te poruku o grešci koja se dogodila u tom trenutku. Osim poruke, na izvještaju je prikazana i slika zaslona uslikana u trenutku kada je test pao. Tokom razvoja testova pojavio se problem jedinstvenosti identifikatora po kojima se dohvaćaju elementi te se radi toga nekim elementima dodao dodatni atribut po kojemu se može diferencirati. Jedna od glavnih funkcionalnosti Cypress alata je upravo velika brzina izvršavanja, što možemo vidjeti zajedno s provedenim testovima i pronađenim greškama u tablici 7.2.

Tablica 7.2 Izvještaj s testiranja

Sučelje	Naziv testa	Greška (ako postoji)	Vrijeme izvršavanja	Uspješnost izvršavanja
Sučelje za prijavu korisnika	Sučelje sadrži ispravan naslov	-	961ms	Uspješno izvršen
	Korisnik se može prijaviti kao pacijent	-	2.810s	Uspješno izvršen
	Korisnik se može prijaviti kao liječnik	-	1.557s	Uspješno izvršen
Sučelje za registraciju korisnika	Sučelje sadrži ispravan naslov	-	1.159s	Uspješno izvršen
	Korisnik se može registrirati kao pacijent	Greška da korisnik već postoji. Rješenje: Greška u implementaciji testa, dodano nasumično generiranje mailova.	2.499s	Neuspješno izvršen
	Korisnik se može registrirati kao liječnik	-	2.528s	Uspješno izvršen
Sučelje za pregled pacijenata i unos nalaza	Sučelje sadrži ispravan naslov	Zatipak u naslovu sučelja, umjesto „My patients“, naslov je „MyPatients“. Rješenje: Ispravljen tekst na <h1> elementu.	4.90s	Neuspješno izvršen
	Sučelje sadrži tablicu s pacijentima	-	1.761s	Uspješno izvršen
	Svaki redak tablice s pacijentima sadrži gumb za unos nalaza	-	2.518s	Uspješno izvršen

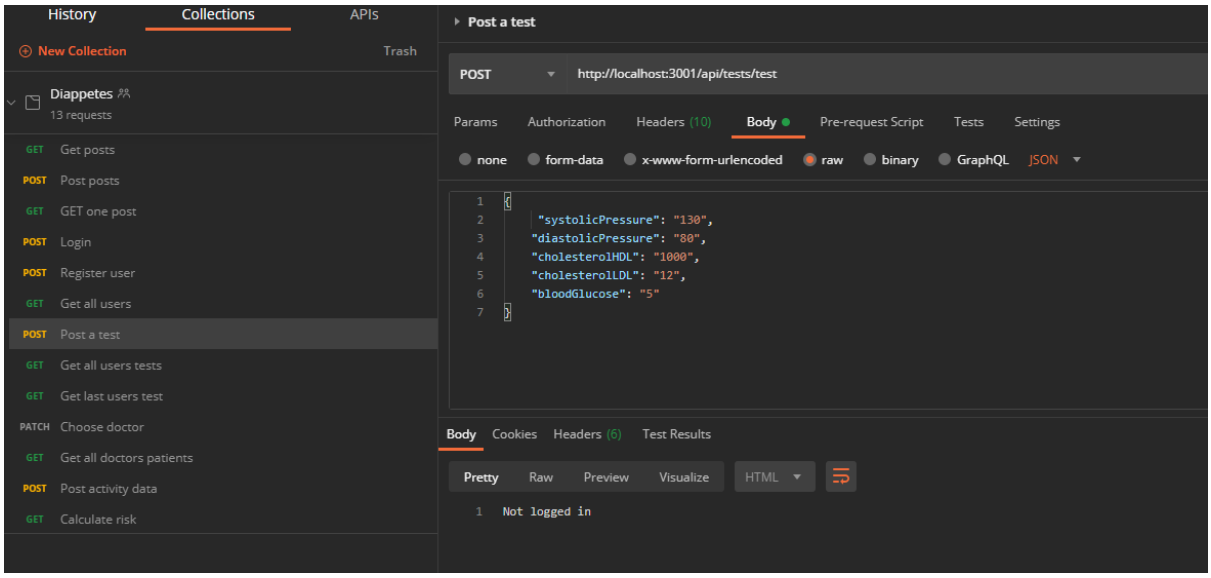
	Liječnik može unijeti nalaz za pacijenta	-	4.535s	Uspješno izvršen
Sučelje za pregled nalaza i unos podataka s pametne narukvice	Sučelje sadrži ispravan naslov	-	4.208s	Uspješno izvršen
	Sučelje sadrži tablicu s nalazima	-	1.772s	Uspješno izvršen
	Svaki redak tablice s nalazima sadrži gumb za unos podataka s pametne narukvice	-	2.601s	Uspješno izvršen
	Pacijent može unijeti podatke s pametne narukvice	-	3.954s	Uspješno izvršen
Sučelje za odabir liječnika	Sučelje sadrži ispravan naslov	-	7.999s	Uspješno izvršen
	Sučelje sadrži tablicu s liječnicima	-	4.277s	Uspješno izvršen
	Svaki redak tablice s liječnicima sadrži gumb za odabir liječnika	-	4.909s	Uspješno izvršen
	Pacijent može odabrati liječnika	-	5.789s	Uspješno izvršen
Sučelje za pregled vijesti	Sučelje sadrži naslovnu sliku	-	2.34s	Uspješno izvršen
	Sučelje sadrži objave o dijabetesu	Neprijavljeni korisnik ne vidi objave o dijabetesu. Rješenje: Za ispravak problema promijenjena je API metoda – uklonjen je uvjet da korisnik treba biti prijavljen za pregled vijesti	563ms	Neuspješno izvršen
Ukupno	20 testova	Sve pronađene greške u sustavu su uklonjene.	1:02min	17 uspješnih, 3 neuspješna, 85 % uspješnosti

Nakon ispravka greški pronađenih tokom izvršavanja automatiziranih testova, testovi su se pokrenuli ponovno te su pokazali 100% uspjeha u izvršavanju te nikakvi dodati problemi nisu bili pronađeni u web aplikaciji.

7.5. Provođenje testiranja API-ja

API testiranje izvršavalo se tokom implementacije poslužiteljskog dijela aplikacije. U alatu Postman kreirana je zbirka API testova koji testiraju API upravljače implementirane na poslužiteljskoj strani web aplikacije. Kako bi se izvršio test, potrebno je dodati HTTP metodu koja će se koristiti, URL API zahtjeva te ukoliko je potrebno dodati kolačić za autorizaciju i podatke

koji će se slati. Primjer API testa koji prikazuje kako nije moguće poslati podatke za nalaz ukoliko korisnik nije prijavljen prikazan je na slici 7.5.

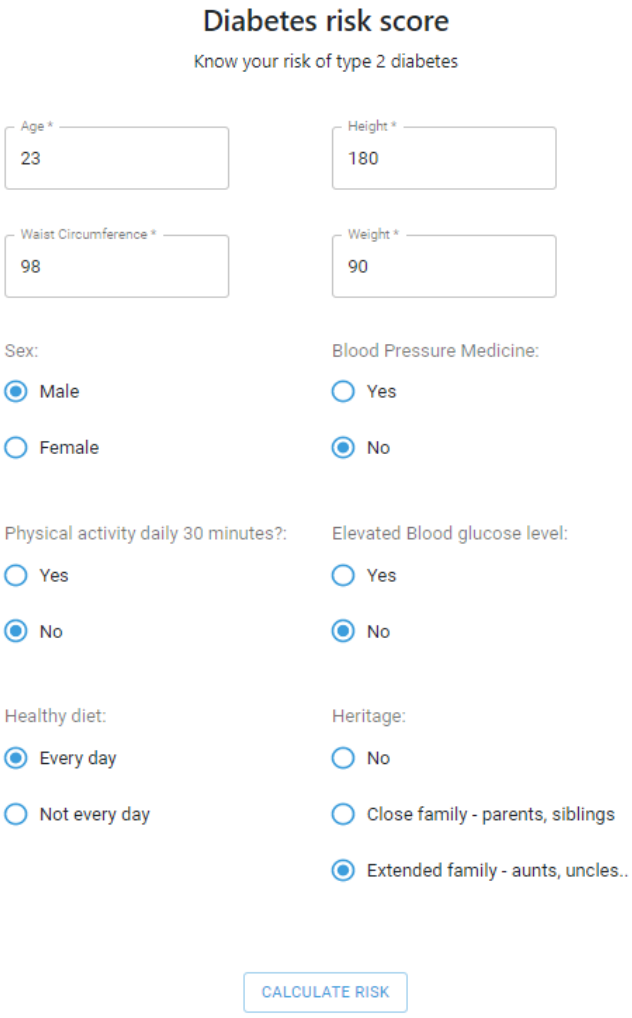


Slika 7.5. Primjer API testa

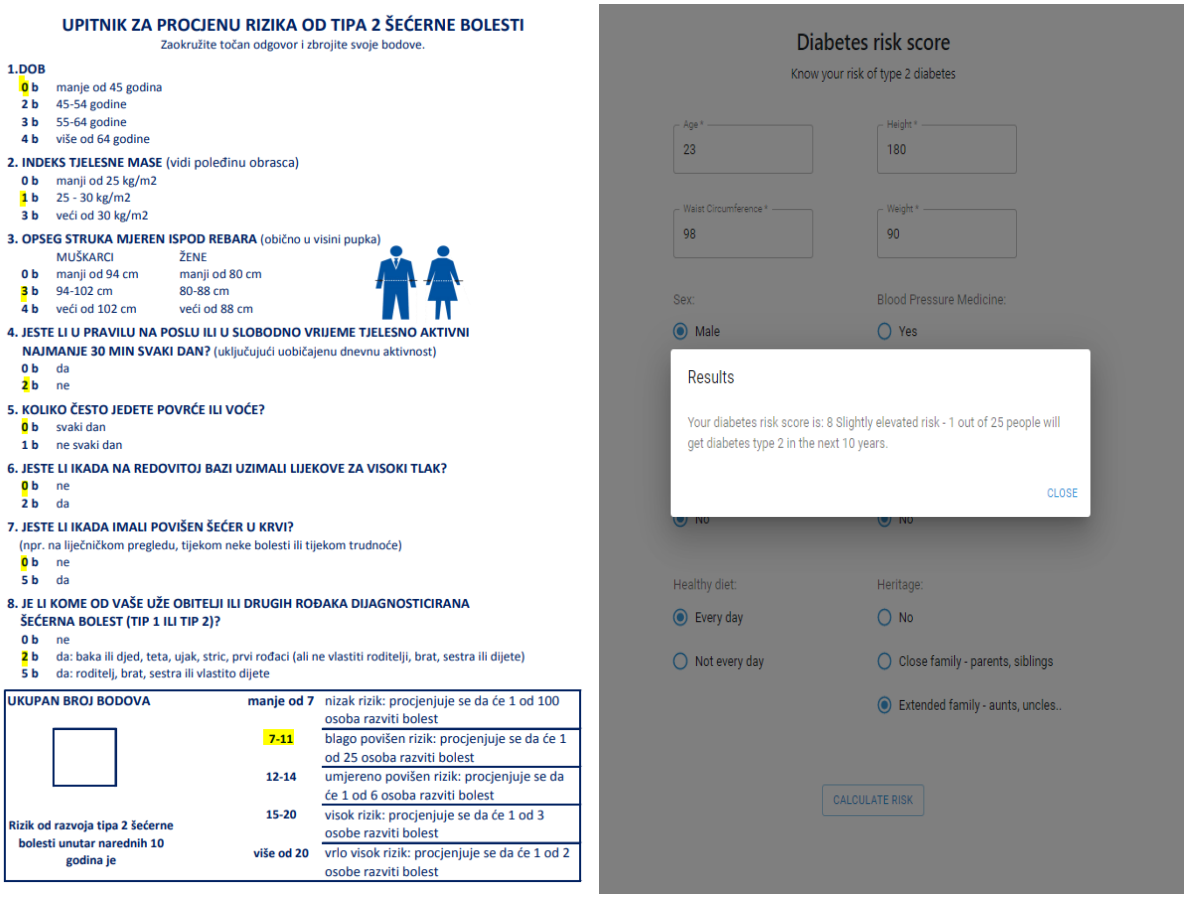
Sve greške otkrivene tokom testiranja API-ja popravljene su tokom implementacije te svaka implementirana API metoda ispravno radi.

7.6. Provođenje ručnog testiranja

Ručno testiranje provodilo se izvršavajući scenarije navedene u tablici 7.1. Ručno su se uspoređivali ulazni podaci i/ili akcije s očekivanim rezultatom. Primjerice izračun rizika testiran je na više setova ulaznih podataka te se rezultat uspoređivao s onim dobivenim kod rješavanja upitnika iz kojeg je izračun rizika preuzet. Primjer ulaznih podataka za testni slučaj za izračun rizika prikazan je na slici 7.6, a rezultat je prikazan na slici 7.7.



Slika 7.6. Primjer ulaznih podataka na korisničkom sučelju



Slika 7.7. Rezultat za uneseni set ulaznih podataka i očekivani rezultat

Svi testovi su uspješno izvedeni te su pronađeni problemi zapisani u tablici 7.3.

7.6.1. Rezultati ručnog testiranja

Nakon ručnog testiranja pronađeno je nekoliko bugova u samoj implementaciji aplikacije. Popis bugova nalazi se u tablici 7.3.

Tablica 7.3. Popis bugova

Rbr.	Naziv buga	Koraci	Prioritet
1	Prijavljeni korisnik može pristupiti sučelju za prijavu	1. Prijaviti se u aplikaciju 2. Klik na poveznicu “Login/Register”	1
2	Na sučelju za pregled pacijenata nisu vidljivi podaci u stupcu “Last Name”	1. Prijaviti se kao liječnik 2. Klik na poveznicu “My Patients”	2
3	Odabir gumba za unos nalaza na sučelju za pregled pacijenata radi samo za prvi redak u tablici.	1. Prijaviti se kao liječnik 2. Klik na poveznicu “My patients” 3. Klik na bilo koji gumb u tablici osim prvoga	1
4	Zatipak na sučelju za prikaz vijesti	1. Otvoriti sučelje za prikaz vijesti 2. Zatipak na komponenti hero image	3

Svakoj od grešaka dodijeljen je određeni prioritet ispravljanja. Prioritet 1 označava kritičnu grešku u funkcionalnom radu sustava koju se treba ispraviti što je ranije moguće. Prioritet 2 je greška umjerene važnosti u sustavu, potrebno je da se ispravi, no nije hitno. Prioritet broj 3 predstavlja grešku koja ne utječe na funkcionalnost rada aplikacije, većinom su to razne vizualne greške, ukoliko ih se ne ispravi, sustav će i dalje normalno raditi. Svi pronađeni problemi i neslaganja sa zahtjevima su uspješno ispravljeni u programskom kodu redoslijedom koji je u skladu s određenim prioritetom. Nakon ispravka problema, ponovno testiranje je pokazalo da su sve funkcionalnosti aplikacije ispravno implementirane.

8. ZAKLJUČAK

U diplomskom radu izrađena je i testirana web aplikacija za potporu oboljelima od dijabetesa. Ciljana skupina korisnika su joj ne samo oboljeli, već i zdravi ljudi s rizikom od dijabetesa koji mogu pomoću aplikacije izračunati svoj rizik obolijevanja od te bolesti. Osim toga, aplikacija poboljšava i olakšava komunikaciju između pacijenata i liječnika te ubrzava cijeli proces dostavljanja i interpretiranja nalaza pacijentima. Pacijenti su dolaskom nalaza trenutno obaviješteni o vlastitom zdravstvenom stanju što je veoma bitno kod pravovremene reakcije stanja koja ugrožavaju zdravlje i život.

Za izradu programskog rješenja napisani su zahtjevi i funkcionalnosti koje bi aplikacija trebala sadržavati. Sukladno sa zahtjevima, razrađen je model web aplikacije i izabrane odgovarajuće tehnologije. U aplikaciji je implementirano javno sučelje i sučelje dostupno samo prijavljenim korisnicima. Javno sučelje prikazuje korisnicima novosti i članke o dijabetesu te im dopušta izračun rizika od razvijanja dijabetesa tipa 2 u sljedećih 10 godina. Također su implementirana sučelja za registraciju u ulozi liječnika ili pacijenta i prijavu nakon kojih se može pristupiti zaštićenom dijelu portala. U ovom dijelu portala događa se komunikacija pacijenta i liječnika. Pacijent odabire liječnika koji zatim ima mogućnost unosa nalaza pacijentu. Analiza zdravstvenog stanja detaljno je opisana u radu te je korištena u informiranju pacijenta o vlastitom zdravstvenom statusu. Nad aplikacijom je provedeno detaljno funkcionalno testiranje, ručno i automatizirano, koje je ukazalo na implementacijske greške koje su popravljene u programskom rješenju.

Aplikaciju bi bilo moguće unaprijediti dodavanjem integracije s vanjskim sustavom koji bi provjeravao legitimitet liječnika i pacijenata te dodavanjem dodatne korisničke uloge administratora koji bi upravljao korisnicima i sadržajem na web aplikaciji.

LITERATURA

[1] M. M. Kebede, C. R. Pischke, Popular Diabetes Apps and the Impact of Diabetes App Use on Self-Care Behaviour: A Survey Among the Digital Community of Persons With Diabetes on Social Media, ožujak 2019., objavljeno online, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6407478/,

[2] J. Li, J. Huangh, L. Zheng, Application of Artificial Intelligence in Diabetes Education and Management: Present Status and Promising Prospect, svibanj 2020., objavljeno online, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7273319/

[3] P. Hjortdahl, J.J. Tufano, E. Arsand, Designing Mobile Dietary Management Support Technologies for People with Diabetes, J Telemed Telecare, 14(7), str. 329. – 332., 2008., https://pubmed.ncbi.nlm.nih.gov/18852310/

[4] A. Anand, Importance of Software Testing in the Process of Software Development, International Journal for Scientfic Research and Development, 12(6), siječanj 2019., https://www.researchgate.net/publication/331223692_Importance_of_Software_Testing_in_the_Process_of_Software_Development

[5] J.J. Majikes, R. Pandita, T. Xie, Testing Techniques for Medical Device Software, 2013., objavljeno online, http://rahulpandita.me/files/majikesLitReview.pdf

[6] Diabetes.co.uk, Diabetes and Metabolism, https://www.diabetes.co.uk/diabetes-and-metabolism.html, pristupljeno: 5.7.2020.

[7] G. Wilcox, Insuline and Insuline Resistance, Clin Biochem Rev, 26(2), str. 19. – 39., svibanj 2018., https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1204764/

[8] Mediligo, Inzulinska pumpa, https://mediligo.hr/inzulinska-pumpa/, pristupljeno: 5.7.2020.

[9] American Diabetes Association, Diagnose and Classification of Diabetes Melitus, Diabetes Care, 28(1), siječanj 2005., https://care.diabetesjournals.org/content/diacare/28/suppl_1/s37.full.pdf

[10] MSD priručnici, Poliurija, http://www.msd-prirucnici.placebo.hr/msd-simptomi/poliurija , pristupljeno: 5.7.2020.

[11] P. C. Cryer, S. N. Davis, H. Shamoon, Hypoglycemia in Diabetes, Diabetes Care, 26(6), str. 1902. – 1912., lipanj 2003., https://care.diabetesjournals.org/content/26/6/1902.short

[12] MSD priručnici, Hipoglikemija, http://www.msd-prirucnici.placebo.hr/msd-prirucnik/endokrinologija/secerna-bolest-i-otkloni-mijene-ugljikohidrata/hipoglikemija, pristupljeno: 5.7.2020.

[13] MSD priručnici, Diabetes Mellitus, http://www.msd-prirucnici.placebo.hr/msd-prirucnik/endokrinologija/secerna-bolest-i-otkloni-mijene-ugljikohidrata/diabetes-mellitus, pristupljeno: 5.7.2020.

[14] WHO, Diabetes, https://www.who.int/news-room/fact-sheets/detail/diabetes, pristupljeno: 5.7.2020.

[15] K. Alberti, P. Zimmet, J. Shaw, International Diabetes Federation: A Consensus in Type 2 Diabetes Prevention, Diabetic Medicine, 24(5), str. 451. – 463., 2007. , https://onlinelibrary.wiley.com/doi/full/10.1111/j.1464-5491.2007.02157.x

[16] HZJZ, Dijabetes, https://www.hzjz.hr/sluzba-epidemiologija-prevencija-nezaraznih-bolesti/odjel-za-koordinaciju-i-provodenje-programa-i-projekata-za-prevenciju-kronicnih-nezaraznih-bolest/dijabetes/, pristupljeno: 5.7.2020.

[17] M. Baretić, Fizička aktivnost i šećerna bolest, Acta Med Croatica, 71 (1), str. 57.-62., 2017., https://hrcak.srce.hr/184907

[18] H. Brlečić, L. Ružić, Učinci tjelesne aktivnosti aerobnog i anaerobnog tipa na smanjenje doze inzulina kod dijabetičara, Hrvatski športsko medicinski vjesnik, 29 (2), str. 60 – 66, 2014., https://hrcak.srce.hr/136932

[19] J.M. Dos Santos, M. L. Moreli, S. Tewari, S.A. Benite-Ribeiro., The Effect of Exercise on Skeletal Muscle Glucose Uptake in Type 2 Diabetes: An Epigenic perspective, prosinac , Metabolism, 64(12), str. 1619. – 1628., rujan 2015., https://pubmed.ncbi.nlm.nih.gov/26481513/

[20] HZJZ, Upitnik za procjenu rizika od dijabetesa tipa 2.https://www.hzjz.hr/wp-content/uploads/2016/08/UPITNIK-ZA-PROCJENU-RIZIKA-OD-TIPA-2-%C5%A0E%C4%86ERNE-BOLESTI.pdf], pristupljeno: 20.8.2020.

[21] Mayo clinic, Normal resting heart rate per minute, https://www.mayoclinic.org/healthy-lifestyle/fitness/expert-answers/heart-rate/faq-20057979#:~:text=A%20normal%20resting%20heart%20rate%20for%20adults%20ranges%20from%2060,to%2040%20beats%20per%20minute., pristupljeno: 20.8.2020.

[22] Heart.org, Understanding Blood Pressure Readings, https://www.heart.org/en/health-topics/high-blood-pressure/understanding-blood-pressure-readings, pristupljeno: 20.8.2020.

[23] Diabetes Self-management, Blood sugar chart, https://www.diabetesselfmanagement.com/managing-diabetes/blood-glucose-management/blood-sugar-chart/, pristupljeno: 20.8.2020.

[24] Mayo clinic, HDL Cholesterol, https://www.mayoclinic.org/diseases-conditions/high-blood-cholesterol/in-depth/hdl-cholesterol/art-20046388, pristupljeno: 20.8.2020.

[25] Medline Plus, LDL – The bad cholesterol, https://medlineplus.gov/ldlthebadcholesterol.html, pristupljeno: 20.8.2020.

SAŽETAK

U diplomskom radu obrađuje se izrada i funkcionalno testiranje aplikacije za potporu oboljelima od dijabetesa. U praktičnom dijelu rada je izrađena aplikacija koja omogućuje pregled vijesti o dijabetesu, računanje rizika od dijabetesa tipa 2 te lakšu komunikaciju između pacijenta i liječnika. U svrhu osiguravanja kvalitete, provedeno je funkcionalno testiranje izvršavajući raspisane testne slučajeve te je nakon pronalaska i ispravka problema zaključeno da su ispravno razvijene sve potrebne funkcionalnosti i zahtjevi na aplikaciju. Osim samog načina implementacije, testiranja i korištenja raznih alata, opisana je i problematika dijabetesa kao globalnog zdravstvenog problema.

Ključne riječi: dijabetes, funkcionalno testiranje, web aplikacija, izračun rizika obolijevanja

ABSTRACT

This paper elaborates on the implementation and testing of the application for support of the patients who suffer from diabetes. In the practical part of the thesis, a web application was implemented which enables its users to keep track of the diabetes news, calculating their diabetes risk and easier doctor-patient communication. Functional testing was conducted by executing defined test cases. When found problems were fixed, application was deemed to have all desired functionalities. Global problem with the diabetes is described along with the implementation, testing and tools used.

Key words: diabetes, functional testing, web application, diabetes risk calculation

ŽIVOTOPIS

Ena Filipović je rođena 25. ožujka 1997. godine u Požegi. Završila je osnovnu školu Vilima Korajca 2011. godine u mjestu Kaptol nakon koje u Požegi upisuje opću gimnaziju u Gimnaziji Požega. Nakon završene srednje škole, upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija na Sveučilištu J. J. Strossmayera u Osijeku. Na trećoj godini studija zapošljava se u informatičkoj tvrtki Span gdje radi do danas.

_____

Ena Filipović

PRILOZI

Prilog 1: Diplomski rad u .pdf formatu.

Prilog 2: Diplomski rad u .docx formatu

Prilog 3: Programski kod aplikacije i automatiziranih testova u .zip datoteci.

Prilog 4: Izvješće o obavljenom testiranju naziva „TestReport.html“