Mobilna Android aplikacija za utvrđivanje nedostatka testosterona i praćenje nadomjesnog liječenja

Butković, Branimir

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

Sveučilišni preddiplomski studij računarstva

ANDROID MOBILNA APLIKACIJA ZA PRAĆENJE NEDOSTATKA I PROVEDBU TERAPIJE NADOKNADE TESTOSTERONA

Završni rad

Branimir Butković

Osijek, 2019.

SADRŽAJ

1. UVOD

Testosteron je ponajviše muški spolni hormon kojeg u muškaraca luče testisi. U manjoj mjeri luči se i kod žena u jajnicima i nadbubrežnim žlijezdama. Poremećaji u njegovom radu mogu uzrokovati različite probleme u ljudskom tijelu stoga je važno na vrijeme otkriti ukoliko postoji problem, suprotstaviti se problemu određenim terapijama te na posljetku i riješiti problem. U ovom završnog radu cilj je izraditi mobilnu aplikaciju za praćenje nedostatka i provedbu terapije nadoknade testosterona. Korisnik će kroz aplikaciju imati mogućnost unosa ulaznih parametara prema preporukama medicinske struke prema kojima će dobiti određeni bodovni rezultat i preporuku o fazi liječenja.

U ovom završnom radu korisnik aplikacije ima mogućnost provedbe kratkog upitnika (ankete). Upitnik je osmišljen kroz nekoliko pitanja. Korisnik prolazi kroz svako pitanje i ostavlja određeni utisak tj. ocjenu te nakon toga dobiva rezultate. Na osnovu rezultata upitnika otkrivaju mu se detalji o postojanju ili nepostojanju nedostatka hormona testosterona. Također, na osnovu tih rezultata korisnik dobiva informaciju o potrebitosti obavljanja terapije. Ukoliko se ispostavi da je terapija potrebna, korisnik će o tome biti obaviješten. Upravljanje korisnicima aplikacije bit će ostvareno putem oblaka, točnije baze podataka *Firebase* gdje su spremljeni svi podaci o trenutno postojećim korisnicima uključujući i rezultate upitnika prije i poslije terapije. Korisniku će također biti omogućen uvid u sve prethodne rezultate upitnika koje je izvršavao u prošlosti. Na taj način korisnici mogu pratiti razinu testosterona putem rezultata upitnika. Uz određeni rezultat upitnika, bit će prikazan i datum radi lakše orijentacije na vrijeme provedenih anketiranja.

U poglavlju 2 objašnjen je hormon testosteron, otkrivanje njegovog nedostatka i liječenje te pregled postojećih rješenja. U poglavlju 3 opisan je način prikupljanja podataka unutar aplikacije, prikazan je dijagram toka korištenja aplikacije te pregled postojećih rješenja. Poglavlje 4 daje prikaz platformi i usluga korištenih u aplikaciji, programskih jezika u kojima će biti izvedena mobilna aplikacija te detaljno opisuje komponente aplikacije kao npr. *RecyclerView* i *GraphView*. Prije zaključka, u poglavlju 5 prikazano je korištenje aplikacije kroz nekoliko slučajeva uz priložene slike i objašnjenja.

2. PROBLEMI KOD NEDOSTATKA TESTOSTERONA

2.1. Općenito o testosteronu

Testosteron [1] je spolni hormon koji se kod muškaraca luči u testisima, dok se kod žena, u manjoj mjeri luči u jajnicima. Nešto manji udio hormona također se luči i u nadbubrežnim žlijezdama. 95% hormona proizvodi se u testisima, a ostatak u nadbubrežnim žlijezdama. Glavni je muški hormon te kod žena i muškaraca igra veliku ulogu u održavanju zdravlja. Njegovi učinci povezani su s dobi. U prosjeku, pojačano se javlja od puberteta pa do 18. godine života. Hormon je dakle zastupljen i kod muškaraca i kod žena. Međutim kod žena je zastupljen u puno manjim koncentracijama nego kod muškaraca.

Osim utjecaja na spolnost, glavna funkcija navedenog hormona je u sljedećim dijelovima ljudskog tijela, a to su: glas, kosti, krv, um, koža, dlake, itd. Testosteron produbljuje glas, posebice u pubertetu, povećava koštanu masu, potiče rast dlaka na licu i tijelu, kontrolira seksualni nagon i funkciju, razvija spolne organe, itd. Referentne vrijednosti koncentracije hormona kod muškaraca kreću se između 9.9 – 27.8 nmol/L, dok su kod žena između 0.22-2.9 nmol/L.

Stvaranje testosterona kod muškaraca smanjuje se nakon 55. ili 60. godine života, no ne dolazi do velikog pada njegove razine. Kod muškaraca se u navedenim godinama javlja razdoblje andropauze. Andropauza se poistovjećuje s menopauzom kod žena. Neki od simptoma andropauze su umor, slabiji tek, smanjenje libida, smanjenje ili gubitak potencije, razdražljivost, smanjena razina koncentracije, itd.

Prema [2], stvaranje testosterona kod žena smanjuje se nakon 30. godine života, dok se između 48. i 52. godine javlja razdoblje menopauze, gdje se navedeni spolni hormon smanjuje oko 15%. Žene s hormonalnim poremećajem tj. viškom testosterona mogu imati nepravilan menstrualni ciklus, akne po licu i tijelu, pojačanu dlakavost i sl. Menopauza predstavlja kraj fizioloških menstrualnih krvarenja, točnije kraj reproduktivne dobi kod žene. U užem smislu, menopauza označava zadnju menstruaciju u životu žene te je dio prirodnog procesa starenja.

2.2. Nedostatak testosterona i njegovo liječenje

S obzirom da je testosteron iznimno važan hormon za ljudsko zdravlje, važno je na vrijeme uočiti njegov nedostatak da bi se na vrijeme mogla provesti određena terapija [20]. Za spolnu diferencijaciju, tj. stvaranje razlike između muškaraca i žena, važna je prisutnost testosterona između trećeg i četvrtog mjeseca razvoja zametka. Tijekom spolne diferencijacije dolazi do

poremećaja u spolnom razvoju. Poremećaji nastaju zbog smanjenog ili prekomjernog lučenja hormona odgovornih za spolni razvoj. Prema [5], neki od poremećaja u spolnom razvoju su: kongenitalna adrenalna hiperplazija (KAH), sindrom neosjetljivosti na androgene (PAIS), sindrom pretjerane aromataze (AEXS), itd. KAH je genetički nedostatak kod kojeg je smanjena proizvodnja hormona kore nadbubrežne žlijezde što dovodi do prekomjernog lučenja testosterona. Terapija KAH poremećaja podrazumijeva nadomjestak hormona kore nadbubrežne žlijezde i kirurško rekonstrukciju genitalija. PAIS je poremećaj kod kojeg muškarac ima fizičke karakteristike žene zbog određenih abnormalnosti na genetskom XY kromosomu. PAIS terapija podrazumijeva psihološku podršku i terapiju hormonima. AEXS je rijetka genetska bolest čiji su simptomi povećanje muških grudi u fazi prije i nakon puberteta, smanjena visina u odrasloj dobi, piskutav glas i dr. Uspješnom terapijom za poremećaj AEXS pokazala se psihološka podrška te kirurško odstranjivanje dojke (mastektomija).

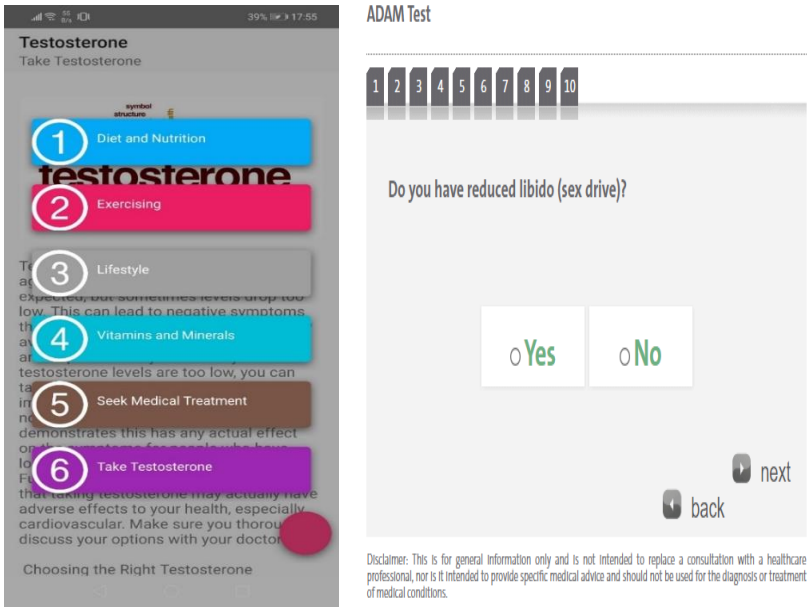
Moguće je da manjak testosterona rezultira depresivnim raspoloženjem. Niske razine ovog oblika spolnog hormona također su vidljive kod nekih kroničnih stanja i bolesti poput bolesti jetre, testisa, dijabetesa tipa 2 i sl. Prema [3], životni stil, traume, nezadovoljstvo partnerom te razina stresa također mogu biti neki od uzročnika manjka testosterona. Postoje različite mogućnosti liječenja muškaraca sa smanjenom razinom testosterona. Mogućnost određivanja koncentracije testosterona može se napraviti jednostavno i to iz uzorka krvi koji je uzet ujutro. Primjerice, simptomi smanjene razine testosterona mogu se poboljšati nadomjesnim liječenjem testosteronom. Nadomjesnim liječenjem testosteronom povisit će se razina testosterona do više konstantne razine i to unutar samo dva dana. U normalnim razinama testosteron ostaje sve dok traje liječenje. Za očekivati je da svi bolesnici neće imati iste rezultate nakon nadomjesnog liječenja, ali pozitivna stavka je da većina muškaraca primjećuje poboljšanje nekih simptoma nakon mjesec dana od početka uzimanja terapije. Osim putem uzorka krvi, moguće je i upitnikom odrediti koncentraciju testosterona. Upitnik (anketa) je namijenjen da bi osobi dao do znanja da poduzme daljnje korake u preventivi zaštite zdravlja, konkretno hormona testosterona. Naravno, postoji više upitnika putem kojih je moguće anketirati osobe poput npr. ADAM testa [6], no unutar ovog rada i same aplikacije korišten je upitnik tvrtke *Solvay Pharma* [4]. Više o upitniku i detaljima vezanih za svaku točku upitnika u potpoglavlju 3.1.

2.3. Pregled postojećih rješenja

Postoje različite aplikacije čija su tema hormoni. Za određene hormone postoje vrlo dobro razvijene aplikacije koje nude mogućnost provedbe upitnika kao i mogućnosti saznanja više informacija o hormonu za kojeg je aplikacija i razvijena. Primjer takve aplikacije je *Blood Test Results*. Aplikacija nudi mogućnost informiranja o različitim čimbenicima koji utječu na krvne nalaze da bi korisnik lakše došao do određenih zaključaka. Aplikacija je podijeljena u više dijelova, kao što su mikrobiološki test, hormonski test, molekularni test, hematološki test, itd. U svakom od tih testova mogu se saznati najvažnije informacije za svaki od njih.

Za hormon testosteron nisu u potpunosti razvijene aplikacije sa svim segmentima uključujući i provedbu upitnika. Uglavnom, sve aplikacije u kojima je hormon testosteron glavna tematika imaju opisan navedeni hormon kroz njegovu ulogu u zdravlju, njegovu ulogu kod muškaraca prvenstveno, načine prevencije kod manjka testosterona, načine povratka koncentracije hormona na standardnu vrijednost, itd. *About Testosterone* je primjer takve aplikacije (slika 2.1 lijevo). Naznačene su najbitnije kategorije i svaka od njih je detaljnije objašnjena, međutim nekakvog oblika ankete nema. Pretražujući Google Play i Apple Store također nije pronađen niti jedan oblik aplikacije koja u sebi sadrži neki oblik ankete kroz koju bi se ostvarila komunikacija između korisnika i aplikacije. Trebalo bi se omogućiti i anketiranje korisnika putem jednostavne ankete koja bi izračunala broj po kojem bi se okarakterizirala rizičnost nedostatka datog hormona te po kojem bi se korisnik signalizirao da nešto nije u redu.

Osim mobilnih rješenja, istraživano je i područje weba. Nadovezujući se na hormon testosteron, najučinkovitije rješenje za testiranje navedenog hormona pronađeno je na jednoj stranici gdje se nudi rješavanje tzv. ADAM testa [6]. ADAM (eng. *Androgen Deficiency of Aging Man*) test vidljiv na slici 2.1 (desno), jednostavan je test koji se koristi u informativne svrhe. Uloga mu je informiranje osobe o razini hormona testosterona te naznaka na posjetu liječniku ukoliko je potrebno. Na temelju prikazanih rješenja i zahtjeva medicinske struke za određivanje razina raznih hormona kao i testosterona te na temelju postojećih rješenja, u poglavlju 3 predložen je model prikupljanja i analize podataka s ciljem određivanja nedostatka testosterona.



Slika 2.1 Aplikacija za hormon testosterona (lijevo) i web stranica koja nudi mogućnost rješavanja ADAM testa (desno)

3. MODEL PRIKUPLJANJA I ANALIZE PODATAKA O NEDOSTATKU TESTOSTERONA

3.1. Prikupljanje podataka putem ankete

Kao što je navedeno u potpoglavlju 2.2, postoje različiti upitnici s kojima je moguće provesti anketiranje, a u ovom završnom radu koristi se upitnik napravljen po uzoru na upitnik tvrtke *Solvay Pharma* [4]. Parametri ankete po kojima je provedeno anketiranje su: bol u zglobovima, znojenje, problemi sa spavanjem, osjećaj pogoršanja zdravstvenog stanja, osjećaj umora, osjećaj razdražljivosti, nervoza, tjeskoba, nedostatak vitalnosti, osjećaj slabosti, depresivno raspoloženje, osjećaj potrošenosti, slabiji rast dlaka i brade, smanjena sposobnost spolnih odnosa, smanjen broj jutarnjih erekcija, slabiji libido (vidljivo u tablici 3.1). Uz svaki od navedenih simptoma, odnosno obilježja po kojima je lako odrediti nedostatak testosterona, korisnik može na svako od danih simptoma, ostaviti utisak, točnije ocjenu od 1-5 (1 – ne, 2 – rijetko, 3 – ponekad, 4 – jako, 5 – vrlo jako), kao što je vidljivo u tablici 3.1.

Na temelju objašnjenih parametara koji su sastavni dio upitnika, može se pristupiti prikupljanju korisničkih podataka što je u aplikaciji i najznačajniji dio. Svakom korisniku omogućeno je registriranje u sustav i otvaranje vlastitog korisničkog računa koji omogućuje ispunjavanje upitnika, ali i pregled njegovih prethodnih anketiranja nakon više uzastopnih korištenja aplikacije i provođenja upitnika. Korisnik će nakon registriranja biti u sustavu, točnije u bazi podataka te će imati mogućnost svakodnevne prijave na svoj profil unutar aplikacije na bilo kojem mobilnom uređaju koji ima instaliranu inačicu aplikacije te pristup internetu.

Tablica 3.1 Prikaz simptoma korištenih za anketiranje uz moguće utjecaje na ukupnu ocjenu

Simptomi	**Utjecaj na ukupnu ocjenu**
Osjećaj da se Vaše opće stanje pogoršava	1-5
Bol u zglobovima i mišićima	1-5
Pretjerano znojenje	1-5
Potreba za spavanjem, osjećaj umora	1-5
Razdražljivost, osjećaj agresivnosti	1-5
Nervoza, nemir, tjeskoba	1-5
Smanjenje mišićne snage	1-5
Depresivno raspoloženje	1-5

Osjećaj da ste prošli svoj vrhunac	1-5
Osjećaj potrošenosti, osjećaj da ste došli do dna	1-5
Sporiji rast brade	1-5
Smanjena učestalost spolnih odnosa	1-5
Smanjen broj jutarnjih erekcija	1-5
Slabiji spolni nagon/libido	1-5

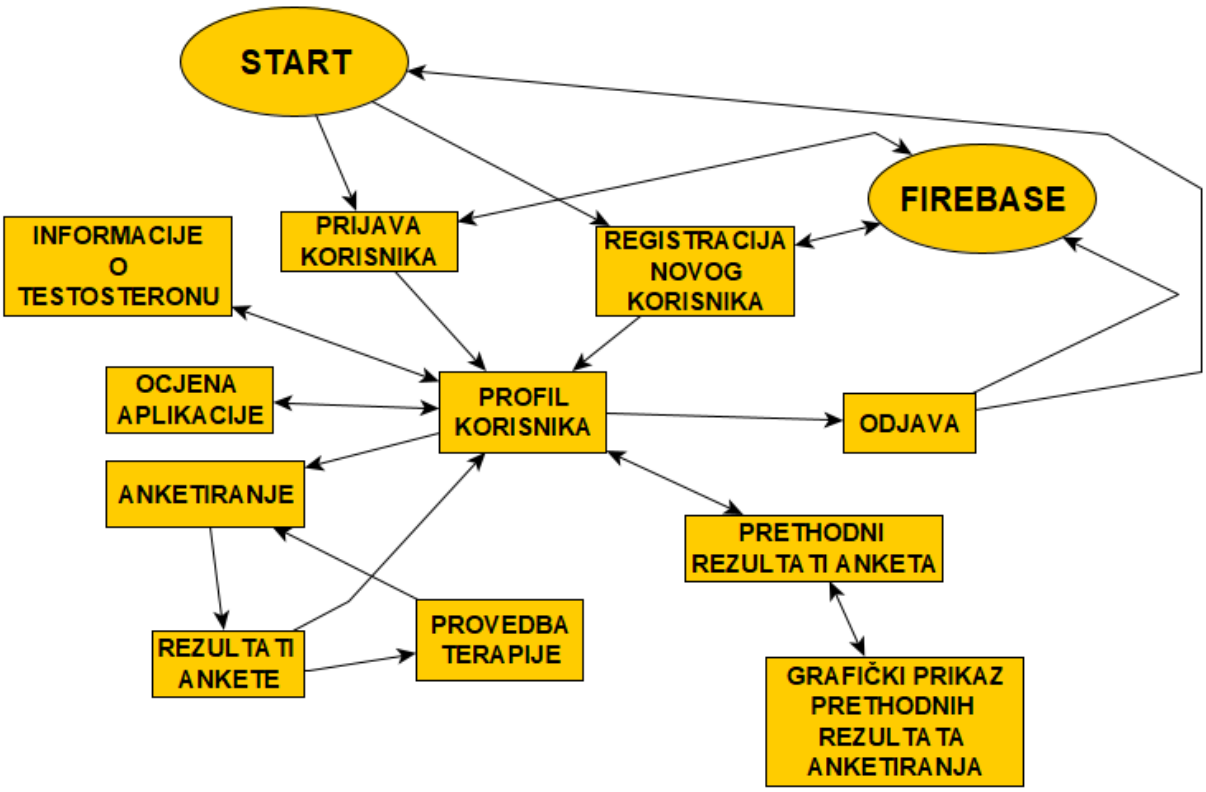
Nakon provedbe upitnika, dobiju se rezultati u obliku tablice 3.2. Tablica 3.2 prikazuje razinu ozbiljnosti simptoma rezultata provedbe ankete i pripadajuće raspone dobivenih bodova.

Tablica 3.2 Klasifikacija rezultata ankete

Simptomi nedostatka testosterona	**Iznos rezultata ankete**
blagi ili nema	<26 bodova
blagi	27-36 bodova
zabrinjavajući	37-49 bodova
ozbiljni	>50 bodova

3.2. Prijedlog ideje vlastitog rješenja

Mobilna aplikacija treba omogućiti i omogućuje informaciju korisniku o nedostatku hormona testosterona ukoliko se to iz ankete i utvrdi. Korisnik treba imati mogućnost registriranja, odnosno kreiranja vlastitog računa uz pomoć kojeg može provesti upitnik, pratiti prethodne rezultate kao i dobiti uvid u prethodne rezultate na temelju grafičkog prikaza prethodnih rezultata kao i prikaz grafa u ovisnosti o rezultatima riješenog upitnika. Ulazni parametar je svaki korisnički odgovor na pitanje unutar upitnika, a izlazni parametar će biti konačni rezultat upitnika. Konačni rezultat signalizira korisniku mogući nedostatak testosterona te upućuje na terapijsko liječenje ukoliko se procjeni da je potrebno. Unutar terapijskog liječenja postoje određeni savjeti. Najbitniji od njih je posjeta liječniku ukoliko rezultat testiranja zaista bude kritičan. Prikaz općenitog modela aplikacije prikazan je na slici 3.1.



Slika 3.1 Prikaz općenitog modela aplikacije

Strelicama je prikazan slijed kojim će se korisnik kretati kroz zaslone aplikacije. Obostrane strelice označavaju da korisnik može bez nekakvih dodatnih akcija sustava izvršiti kretanje između tih zaslona. Npr. korisnik može pregledati detalje vezane za hormon testosteron i vratiti se na zaslon pritiskom tipke nazad (obostrana strelica). Međutim korisnik ne može posjetiti zaslon za prijavu dok na profilnom zaslonu ne odabere gumb za odjavu i prilikom toga pokrene postupak odjavljivanja iz sustava i iz baze podataka (jednosmjerne strelice).

Na temelju predloženih zahtjeva i modela mobilne aplikacije za praćenje nedostatka i provedbu terapije nadoknade testosterona, programsko rješenje koje će biti opisano u poglavlju 4 morat će sadržavati korisničko sučelje, bazu podataka, graf s prikazom rezultata, izbornike i ostale dijelove koji će biti opisani.

4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE

4.1. Operacijski sustav Android

Android [7] je operacijski sustav za pametne telefone i tablet računala, ujedno i platforma koja se koristi prilikom izrade ove aplikacije. Android su osnovali Andy Rubin, Rich Miner, Nick Sears i Chris White 2003. godine, a prvi uređaji koji su ostvareni na operacijskom sustavu Android pojavili su se u javnosti tek 2008. godine. Samom pojavom u javnosti, Android se bacio u utrku s tada trenutno vodećim platformama na tržištu, kao što su to bili Apple-ov iOS, Microsoft-ov Windows Phone, Symbian tvrtke Nokia i Sonny Ericsson.

Zasnovan je na Linux 2.6 jezgri napisanoj u programskim jezicima C i C++. Arhitekturu Android platforme može se promatrati kroz programski stog koji ima više razina [17]. Na dnu stoga nalazi se Linux 2.6 jezgra, dok iznad jezgre postoji niz knjižnica, a neke od najbitnijih su Surface Manager, FreeType, SSL, SQLite, Webkit i ostali. Surface Manager je knjižnica koja nadzire iscrtavanje grafičkog sučelja. FreeType je knjižnica namijenjena iscrtavanju fontova. SSL je knjižnica za sigurnosnu komunikaciju putem interneta. SQLite je knjižnica za upravljanje bazama podataka, dok je WebKit sustav za web preglednike. Nakon knjižnica slijedi sloj Android Runtime koji služi za pokretanje aplikacija. Zatim slijedi aplikacijski okvir dok se na vrhu hijerarhije nalaze same aplikacije, tj. jedini sloj koji je ujedno i vidljiv krajnjem korisniku. Trenutno postoji 10 inačica operacijskog sustava Android, od kojih su najpoznatije Android 4.4 KitKat, 5.0 Lollipop, 6.0 Marshmallow, 7.0 Nougat, 8.0 Oreo, najnovija stabilna verzija 9.0 Pie te najnovija nestabilna verzija 10.0 Q.

Za komunikaciju s drugim uređajima Android koristi Wi-Fi, Bluetooth, LTE, NFC, itd. Od dodatnih alata donosi zaslon osjetljiv na dodir, žiroskop, GPS, akcelerometar kao i mnoga druga dodatna hardverska i softverska rješenja. Podržava mnogo slikovnih formata poput jpg, gif i png formata, ujedno i video formata poput 3gp, mp4, itd.

Prema [17], Neke od ostalih značajki Androida su: podrška za testiranje, predlošci koda, platforma na kojoj se ostvaruju aplikacije za različite uređaje, mogućnost bržeg pokretanja aplikacija nakon minimalnih promjena u kodu, itd. Osim za pametne telefone, operacijski sustav Android koristi se i kod Android TV-a za pametne televizore, Wear OS-a za pametne satove te Android Auto za automobile. Brojne prednosti kao što su raširenost operacijskog sustava, broj korisnika, ali i aplikacija razvijenih upravo na ovom operacijskom sustavu razlog su odabira operacijskog sustava Android pri dizajniranju i programiranju ove aplikacije. Inačica

operacijskog sustava Android na kojoj je aplikacija razvijena je 5.1.1 na sustavu Windows 10 s 4 GB radne memorije.

4.2. Programski jezik Java

Java [16] je viši programski jezik koji u potpunosti podržava objektno orijentiranu paradigmu. Po toj paradigmi, program se sastoji od objekata koji imaju određena stanja i ponašanja te od grupa objekata sa sličnim obilježjima i klasama. Najvažnije karakteristike ovog programskog jezika su: višenitnost, mogućnost rada preko interneta, komunikacija s korisnikom kroz više prozora, rad s bazama podataka, itd.

Java je jedan od najkorištenijih programskih jezika. Na današnjem tržištu, Java se koristi široko i dosljedno tamo gdje je pretežno naglasak na brzinu razvoja programskog sustava nad brzinom zahtjeva rada programa. Iako inspirirana programskim jezikom C, Java pruža veći stupanj sigurnosti i pouzdanosti zahvaljujući VM-u (eng. *Java Virtual Machine*) na kojem se prevodi. Kao programski jezik opće namjene, Java se koristi za izradu samostalnih programa, odnosno aplikacija.

4.3. Baza podataka Firebase

Firebase je NoSQL baza podataka temeljena na oblaku. Prema [8], NoSQL (eng. *Not only SQL*) je relacijska baza podataka koja nije imala podršku SQL-a. U sklopu ove aplikacije korištena je baza podataka *Realtime Firebase* [9] koja omogućuje sinkronizaciju podataka svih klijenata u stvarnom vremenu i omogućuje izvanmrežnu funkcionalnost. Podaci se pohranjuju u bazu podataka u stvarnom vremenu kao JSON [10], a svi povezani klijenti dijele jednu instancu te automatski primaju ažuriranja s najnovijim podacima. JSON je format parova ključ-vrijednost.

Baza podataka *Firebase* dolazi u sklopu operacijskog sustava Android kao ugrađena Google platforma koja se koristi za mnoge zadatke u oblaku. Neke od značajki su: analitika, mrežne baze podataka, mogućnost slanja izvješća o pogrešci u radu aplikacija, strojno učenje u oblaku, spremanje podataka o korisnicima i mnoge druge opcije. U ovom završnom radu glavna funkcija baze podataka Firebase je pohrana podataka o registriranim korisnicima uključujući i rezultate upitnika.

4.4. Prikaz programske implementacije funkcionalnosti aplikacije

4.4.1. Postavljanje baze podataka Firebase

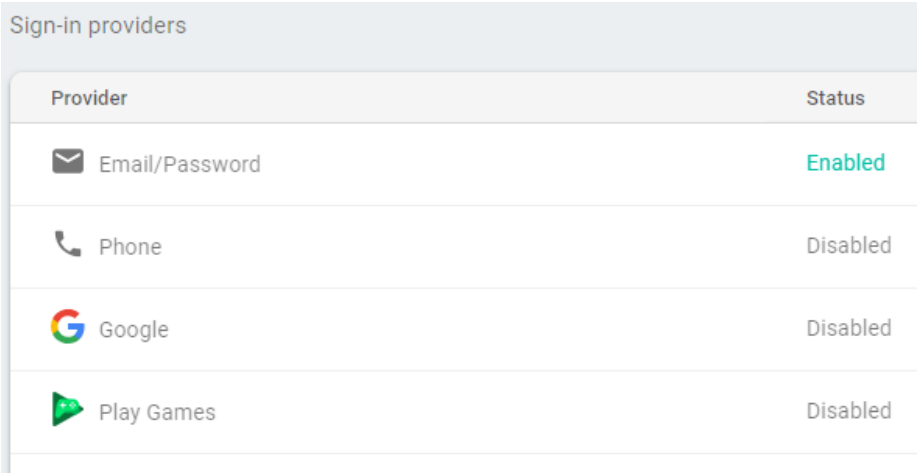
Da bi se mogla koristiti baza podataka *Firebase* koja je detaljnije objašnjena u potpoglavlju 4.3, potrebno je implementirati određene ovisnosti (eng. *dependecies*). Ovisnosti [18] se definiraju u skripti *Gradle* unutar alata Android Studio. Navedene ovisnosti dane su slikom 4.1, a preuzete su iz izvora [11]. Ovisnosti olakšavaju uključivanje vanjskih binarnih zapisa ili drugih modula knjižnice. S obzirom da je unutar aplikacije korištena samo mogućnost autentifikacije korisnika te spremanja podataka o korisniku, nije implementirana npr. ovisnost za pohranu (eng. *storage*) koja bi bila korištena za pohranu fotografija.

```
implementation 'com.google.firebase:firebase-core:16.0.7'
implementation 'com.google.firebase:firebase-auth:16.1.0'
implementation 'com.google.firebase:firebase-database:16.0.6'
```

Slika 4.1 Prikaz potrebnih ovisnosti za bazu podataka Firebase

4.4.2. Autentifikacija pomoću baze podataka Firebase

Implementacijom svih potrebnih ovisnosti za korištenje baze podataka *Firebase*, potrebno je odrediti način na koji će se vršiti autentifikacija korisnika. Autentifikacija se može provesti putem e-pošte i lozinke, što je slučaj u ovoj aplikaciji, ali i putem Facebooka, Twittera, postojećeg Google računa, putem broja telefona, GitHuba, itd. S obzirom da je u radu omogućena autentifikacija putem e-pošte i lozinke, kao što je prikazano slikom 4.2, korisnik treba unijeti navedena dva podatka kako bi se mogao registrirati, a kasnije i prijaviti u kreirani korisnički račun.



Slika 4.2 Dopušteni Sign-in provideri

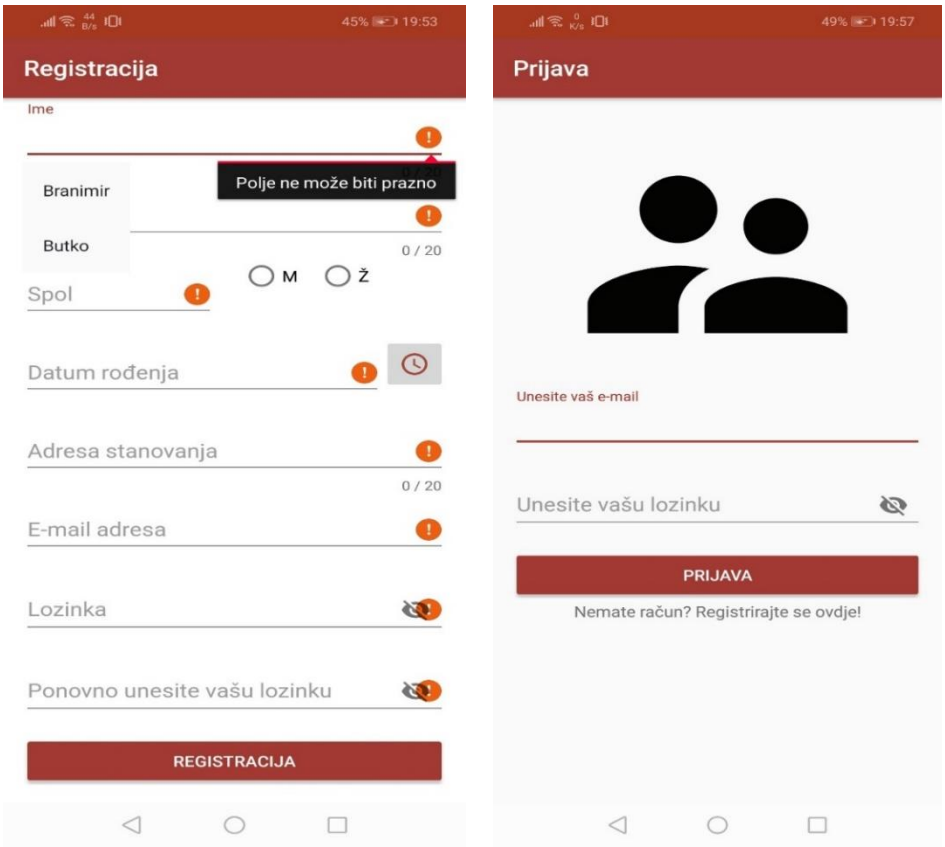
Osim e-pošte i lozinke korisnik prilikom registracije mora unijeti sve podatke, uključujući ime, prezime, spol, godinu rođenja i adresu stanovanja. Ukoliko bilo koji od navedenih parametara nije unesen, korisnik se obavještava porukom, kao što je prikazano slikom 4.4 (lijevo). Nakon što se svi podaci unesu, moguće je podnijeti zahtjev za registracijom korisnika, a ukoliko je registracija uspješna, korisnik se preusmjerava na zaslon s profilom. Programski kod ostvarenja postupka registriranja korisnika prikazan je na slici 4.3.

Glavna funkcija za izvršavanje registriranja je funkcija *registerUser*. Unutar te funkcije, pozivaju su i druge funkcije poput *validateFirstName*. Funkcija *validateFirstName* provjerava ispravnost unosa imena korisnika. Ukoliko ime ima manje od 20 znakova ili ukoliko nije uneseno, pojavljuje se poruka o pogrešci prilikom registriranja. Funkcije *validateLastName, validateDateOfBirth, validateAdress, validateEmail, validatePassword, validatePassword2,* validatePersonSex funkcioniraju na isti način kao i *validateFirstName*, samo što se kod funkcija *validatePersonSex* i *validateDateOfBirth* ne provjerava broj unesenih znakova nego samo postoji li bilo kakav unos. Sve navedene funkcije su *boolean* povratnog tipa, te ukoliko vraćaju laž (eng. *false*) vraćamo se na početak uvjeta *if* sve dok se ne zadovolje zadane ulazne vrijednosti (eng. *input*).

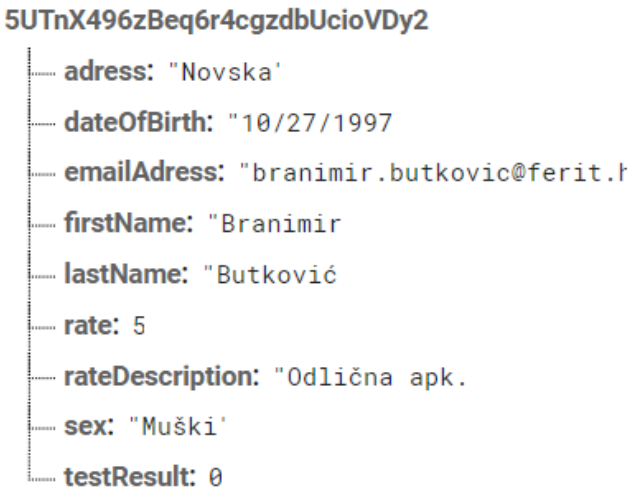
Pritiskom na tipku *Registriranje korisnika* unutar aplikacije, pokreće se mali prozor s porukom o registraciji te izvršavanje ostalih akcija u pozadini. U pozadini se sprema e-pošta i zaporka od korisnika koji se želi registrirati, generira se slučajni korisnički ID i kreira korisnički račun. Osim e-pošte i zaporke, ostali podaci također se spremaju u bazu podataka *Firebase* putem funkcije *addPersonToDatabase*. Izgled spremljenih podataka u bazi podataka *Firebase* prikazan je slikom 4.5. Ukoliko ne dođe do pogreške prilikom registriranja, uključujući moguće greške zbog nedostupnosti mreže, ili pak greške u unosu ulaznih parametara, aplikacija korisnika preusmjerava na zaslon s novokreiranim korisničkim profilom. Ukoliko ipak dođe do pogreške, ispisuje se pripadajuća poruka i prikazuje se isti zaslon dok se ne isprave potencijalne pogreške.

```
private void registerUser(){
    //funkcija za ostvarivanje registracije user-a, te pohranu podataka u Firebase-u

    if(!validateFirstName()||!validateLastName()||!validateDateOfBirth()||!validateAdress()|
            !validateEmail()||!validatePassword()||!validatePassword2()||!validatePersonSex())
        return;
    //ukoliko nešto od ovoga nije zadovoljeno, vraćamo se na početak funkcije dok ne zadovoljimo svaki uvjet

    progressDialog.setMessage("Registriranje...");  //ispis poruke u progressDialogu
    progressDialog.show();      //prikaz te poruke

    String email= etEmail.getText().toString().trim();
    String password=etPassword.getText().toString().trim();

    firebaseAuth.createUserWithEmailAndPassword(email,password)
            .addOnCompleteListener( activity: this, (task) → {
                    progressDialog.dismiss();
                    //gašenje progressDialoga prije provjere uspješnosti spajanja sa bazom
                    if (task.isSuccessful()) {
                        addPersonToDatabase();
                        Toast.makeText( context: RegisterActivity.this, text: "Uspješno ste se registrirali",Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(getApplicationContext(),ProfileActivity.class));
                    }else{
                        Toast.makeText( context: RegisterActivity.this, text: "Pogreška prilikom registracije",Toast.LENGTH_SHORT).show();
                    }
            });
```

Slika 4.3. Programski kod za registraciju korisnika



Slika 4.4 Zaslon za registraciju korisnika (lijevo) i prijavu korisnika (desno)

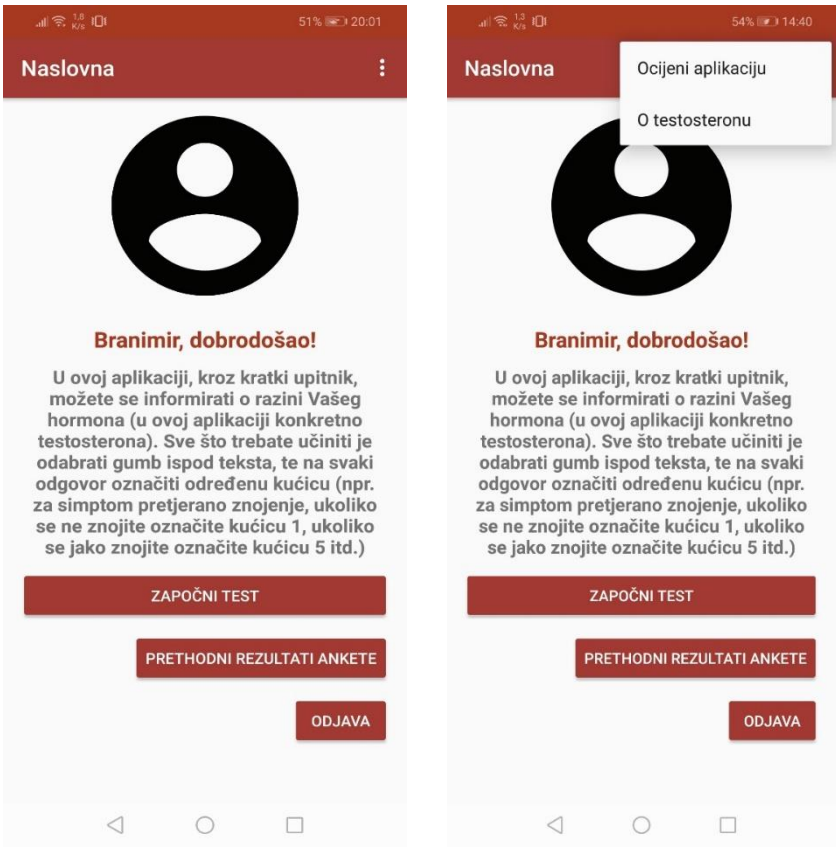


Slika 4.5 Prikaz ulaznih parametara spremljenih u bazi podataka Firebase

Nakon preusmjeravanja na korisnički profil, postoji mogućnost odjave korisnika. Programski kod koji omogućuje odjavu korisnika vidljiv je na slici 4.6. Prvenstveno se korisnik odjavljuje iz baze podataka naredbom *signOut*. Prekidaju se sve povezanosti s bazom podataka *Firebase* putem naredbe *finish* i samim time sve aktivnosti povezane s bazom podataka. Pritiskom na gumb *Odjava*, odvija se preusmjeravanje na zaslon za prijavu korisnika prikazan na slici 4.6. Na zaslonu prijave potrebno je unijeti e-poštu i zaporku korištenu prilikom registriranja korisnika. Nakon pritiska na zahtjev za prijavom, provjerava se postoji li korisnik s takvom e-poštom i zaporkom u sustavu baze podataka, te ukoliko postoji, ostvaruje se prijava i korisnik se preusmjerava na zaslon s profilom. Važno je napomenuti da veza s internetom mora biti stalno omogućena da bi postojala komunikacija između aplikacije i baze podataka. Zasloni s prikazom postupka prijave vidljivi su na slici 4.7.

```
public void Logout(View view){
    //onClick metoda za ostvarivanje logout-a iz baze podataka i pokretanje LoginActivity-a
    firebaseAuth.signOut();     //logout iz baze podataka
    finish();                   //kraj rada sa bazom, prekid konekcije baza-aplikacija
    Toast.makeText( context: this, text: "Uspješno ste se odjavili",Toast.LENGTH_SHORT).show();
    startActivity(new Intent( packageContext: this, LoginActivity.class));
    //prilikom pritiska na logout startamo Login activity
}
```

Slika 4.6 Programski kod za odjavu korisnika



Slika 4.7 Zaslon profila korisnika bez otvorenog izbornika(lijevo) i s otvorenim izbornikom(desno)

Prijava korisnika ostvarena je funkcijom *loginUser*. Prije prijave, potrebno je unijeti e-poštu i zaporku. Ispravnost e-pošte i zaporke provjerava se funkcijama *validateEmail* i *validatePassword*. Ukoliko jedna od njih nije zadovoljena, odnosno ukoliko je rezultat funkcije laž, s obzirom da se radi o *boolean* povratnom tipu podatka, prijava neće biti ostvarena. Također, ukoliko e-pošta ili zaporka ne postoje u bazi podataka *Firebase*, što se provjerava funkcijom *signWithEmailAndPassword* čiji su parametri upravo e-pošta i zaporka, nastupit će poruka o pogrešci prilikom prijave što je i vidljivo u programskom kodu na slici 4.8. Ukoliko korisnik unese ispravnu e-poštu i zaporku, aplikacija preusmjerava korisnika na zaslon s profilom.

```
private void loginUser() {
    /*funkcija za ostvarenje login-a, pokretanje progressDialog-a;
      ukoliko postoji user sa unesenim podacima, spajanje sa bazom podataka i pokretanje ProfileActivity-a,
      ukoliko ne postoji dolazi do poruke o grešci*/
    if (!validateEmail() | !validatePassword())
        return;
    else {
        progressDialog.setMessage("Prijavljivanje...");
        progressDialog.show();

        String email= textInputEmail.getText().toString().trim();
        String password=textInputPassword.getText().toString().trim();

        firebaseAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener( activity: this, (task) → {
                    progressDialog.dismiss();
                    if (task.isSuccessful()) {
                        finish();
                        Toast.makeText(getApplicationContext(), text: "Uspješno ste se prijavili",Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(getApplicationContext(), ProfileActivity.class));
                    }else{
                        Toast.makeText(getApplicationContext(), text: "Unijeli ste krivu lozinku ili e-mail",Toast.LENGTH_SHORT).show();
                    }
            });
    }
}
```

Slika 4.8 Programski kod za prijavu korisnika s postojećim podacima

Također, ukoliko se korisnik ne odjavi s određenog uređaja iz aplikacije, prilikom povratka u aplikaciju (pritiskom na njenu ikonicu), automatski se vraća na zaslon s profilom bez potrebe za ponovnom prijavom. Programski kod na slici 4.9 upravo prikazuje programsko rješenje tog postupka. U prvom redu, provjerava se postoji li trenutno prijavljeni korisnik, a ukoliko ne postoji, prekidaju se sve akcije s bazom podataka *Firebase* i pokreće se zaslon za prijavu korisnika.

```
if(firebaseAuth.getCurrentUser()==null){
    //ukoliko ne postoji trenutni user ulogiran u sustav/bazu, startamo login activity
    finish();
    startActivity(new Intent( packageContext: this, LoginActivity.class));
}
```

Slika 4.9 Programski kod koji provjerava postoji li trenutno prijavljen korisnik u aplikaciji

4.4.3. Profil korisnika i provođenje upitnika

Nakon registriranja novog korisnika ili prijave postojećeg korisnika, korisnik se preusmjerava na zaslon profila. Na zaslonu profila korisnik prvenstveno može pročitati poruku dobrodošlice u kojoj se nalazi njegovo ime. Prvo se pronalazi trenutno prijavljeni korisnik i provjerava se je li on muškog ili ženskog roda. Nakon toga iz baze podataka *Firebase* povlači se podatak o imenu i pripadajućoj poruci dobrodošlice (slika 4.7). Osim poruke dobrodošlice, na istom zaslonu postoji mogućnost provedbe ankete, pregleda prethodno provedenih testiranja, odjave korisnika, korištenja izbornika s dodatnim mogućnostima ocjenjivanja aplikacije te informiranja o

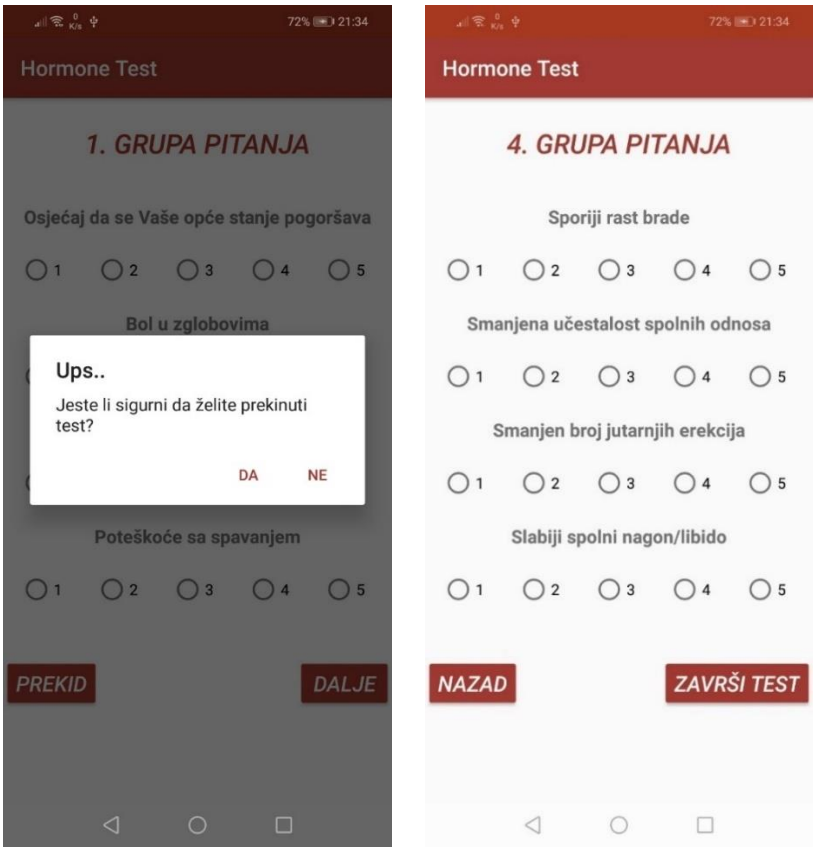
testosteronu (više u potpoglavlju 4.4.9). Anketa je jedan od glavnih dijelova aplikacije, jer je svrha aplikacije provesti anketiranje korisnika.

Anketiranje se provodi kroz četiri zaslona s po četiri pitanja, jer upitnik sadrži ukupno 16 pitanja. Na prvom zaslonu upitnika, postoji mogućnost prekida upitnika uz pripadajuću poruku unutar malog prozora pri čemu se korisnik preusmjerava na zaslon profila. Na ostalim zaslonima (drugom i trećem) postoji mogućnost kretanja između zaslona, dok na zadnjem zaslonu postoji mogućnost završetka anketiranja ili povratka na neki od prethodnih zaslona.

Na slici 4.10 prikazan je programski kod s implementacijom prozora za prekid provedbe upitnika. Unutar prozora 4.11 postoji naslov s porukom o prekidu testiranja uz mogućnost pritiska na gumb *DA* ili *NE*. Ukoliko se pritisne gumb *DA*, pokreće se zaslon s profilom uz poruku dobrodošlice. Ukoliko korisnik odabere gumb *NE*, korisnik ostaje na istom zaslonu uz poništavanje prikaza prozora. Za prekidanje testiranja potrebno je kreirati objekt klase *StopTestDialog* (slika 4.12) koji se kreira prilikom pritiska na gumb prekid i izvršava kod sa slike 4.10.

```
public class StopTestDialog extends AppCompatDialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());
        builder.setTitle("Ups..")
                .setMessage("Jeste li sigurni da želite prekinuti test?")
                .setNegativeButton( text: "Da", (dialog, which) → {
                        startActivity(new Intent(getActivity(), ProfileActivity.class));
                        Toast.makeText(getActivity(),  text: "Dobrodošli nazad na svoj profil", Toast.LENGTH_SHORT).show();
                })
                .setPositiveButton( text: "Ne", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {

                    }
                });
        return builder.create();
    }
}
```

Slika 4.10 Programski kod za prekid testiranja



Slika 4.11 Prikaz prvog zaslona uz pripadajući dialog (lijevo) i zadnjeg zaslona testiranja (desno)

```
public void openDialog(){
    StopTestDialog stopTestDialog=new StopTestDialog(); //kreiranje objekta klase StopTestDialog
    stopTestDialog.show(getSupportFragmentManager(), tag: "StopTestDialog"); //poziv funkcije show
}
```

Slika 4.12 Kreiranje objekta klase StopTestDialog

4.4.4. Postavljanje lokalne baze podataka Realm

Realm [12] je lokalna baza podataka koja se koristi za pohranu podataka na uređaj te kasnije korištenje tih podataka. *Realm* podržava *boolean, byte, short, int, float, double, string* i još neke tipove podataka. Također, nudi mogućnost korištenja primarnih ključeva (eng. *primary key*). Kad se koriste primarni ključevi, čitanje upita je nešto brže, ali pisanje tj. stvaranje i ažuriranje objekata je malo sporije. To naravno ovisi i o veličini skupa podataka unutar baze podataka *Realm*. Također, korištenje primarnih ključeva omogućuje upotrebu *insertOrUpdate* i *copyToRealmOrUpdate* metoda. Da bi se mogla koristiti navedena baza podataka, moraju se implementirati određene ovisnosti (slika 4.13) u skripti *Gradle* kao što je učinjeno i pri korištenju baze podataka *Firebase*. Nakon implementirane ovisnosti, potrebno je omogućiti dodatak (eng. *plugin*) za *Realm*, također u skripti *Gradle* (slika 4.14). Ukoliko se inicijalizira *Realm* putem vlastite stvorene klase, potrebno je tu klasu naznačiti u *AndroidManifest.xml* direktoriju (slika 4.15).

```
dependencies {
    classpath 'com.android.tools.build:gradle:3.4.1'
    classpath 'com.google.gms:google-services:4.2.0'
    classpath "io.realm:realm-gradle-plugin:5.7.0"    //plugin za realm
    // NOTE: Do not place your application dependencies here; they belong
    // in the individual module build.gradle files
}
```

Slika 4.13 Ovisnost potrebna za implementaciju Realm-a (označena žutom bojom)

```
apply plugin: 'com.android.application'
apply plugin: 'realm-android'    //plugin za realm
```

Slika 4.14 Plugin za Realm

```
<application
    android:name=".app.MyApplication"
```

Slika 4.15 Naznačivanje kreirane klase MyApplication za inicijalizaciju Realm-a u AndroidManifest.xml direktoriju

Nakon svih implementiranih ovisnosti i dodataka, *Realm* još nije moguće koristiti. Potrebno je u kreiranoj klasi *MyApplication* koja je naznačena u *AndroidManifest.xml* direktoriju inicijalizirati *Realm* (Slika 4.16). Da bi se kontroliralo kreiranje *Realm* objekata koristi se klasa *RealmConfiguration* s pomoću koje je moguće upravljati s više *Realm* instanci unutar jedne aplikacije i sl.

```
public class MyApplication extends Application {  //za inicijalizaciju Realm baze
    @Override
    public void onCreate() {
        super.onCreate();

        Realm.init( context: this);
        RealmConfiguration configuration=new RealmConfiguration.Builder().build();
        Realm.setDefaultConfiguration(configuration);
    }
}
```

Slika 4.16 Inicijalizacija lokalne baze podataka Realm

4.4.5. Analiza ulaznih podataka

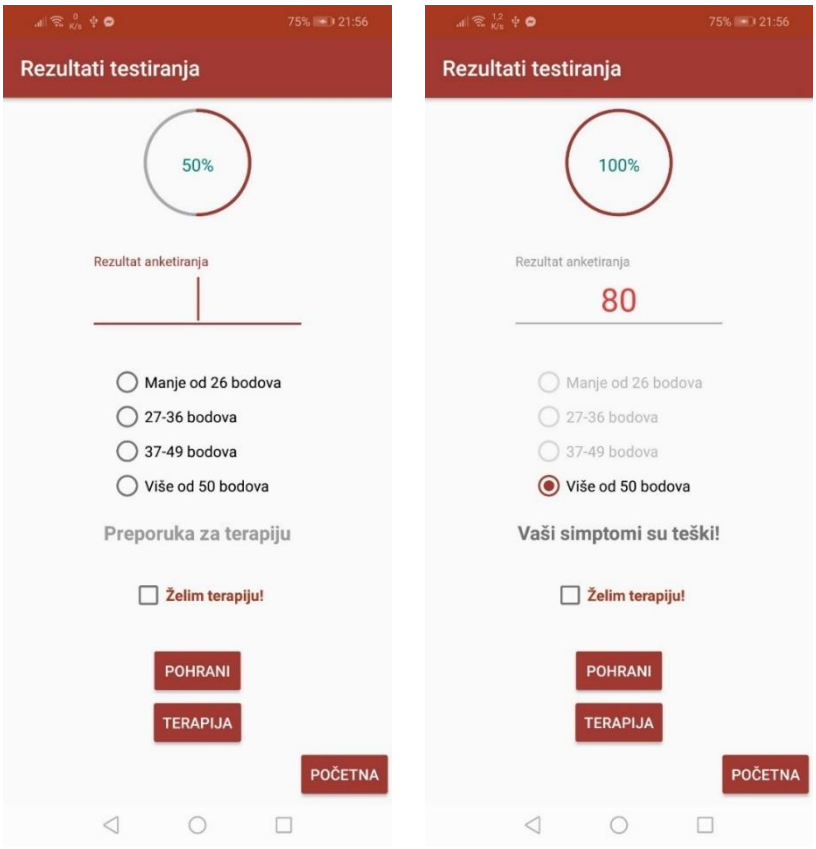
Kroz svaki zaslon upitnika, provode se zbrajanja bodova u ovisnosti o težini (1-5) koju korisnik naznači kao odgovor na pitanje. Prilikom pritiska gumba za završetak provedbe upitnika, svi bodovi s pojedinih zaslona zbrajaju se u jednu varijablu i prikazuju na zaslonu s rezultatima. Maksimalan rezultat upitnika je 80, a minimalan 16. Aritmetika kojom se vrši zbrajanje bodova je poprilično jednostavna. Na svakom zaslonu, od njih ukupno četiri, vrši se zbrajanje, a zatim se zbrajaju bodovi prvog, drugog, trećeg i četvrtog zaslona u zasebne varijable. Na pritisak gumba za završetak upitnika, dohvaćaju se sve varijable iz prethodnih zaslona i spremaju u novu

varijablu s konačnim rezultatom upitnika. Prilikom pokretanja zaslona za prikaz rezultata, odvija se tzv. punjenje *progress bara* u obliku kruga od 0-100%. Prilikom dosega 100%, ispod njega ispisuje se rezultat upitnika, ispod rezultata raspon u koji rezultat spada te preporuka za potencijalno korištenje terapije (slika 4.18). Također, boja rezultata razlikuje se ovisno o rasponu rezultata. Prvi raspon označava se zelenom, drugi narančastom, treći crvenom i posljednji raspon tamno crvenom bojom. U prvom rasponu, terapija je sustavno onemogućena, tj. ne može se odraditi, jer za njom nema potrebe. On označava da je hormon u granicama stabilnosti te da nema potrebe za panikom. U višim rasponima to i nije slučaj, no više o tome u potpoglavlju 4.4.6.

Na slici 4.17 vidljiva je programska implementacija *progress bara*. Postoji sto koraka punjenja *progress bara*, a trajanje svakog je 50 milisekundi. Umnoškom broja koraka i trajanja pojedinog, dobiva se broj od 5000 milisekundi što je jednako 5 sekundi. Dakle, punjenje *progress* bara traje 5 sekundi, te nakon tog vremena ima doseg od 100% uz prikaz navedenih parametara. Prilikom završetka punjenja *progress bara*, metoda *setRingProgressBar* omogućuje ispis poruke i rezultata na zaslonu. Važno je napomenuti i funkciju koja omogućuje označavanje točno jednog raspona od njih ukupno četiri, a to je funkcija *turnOnRadioButtonAndMessageForEachResult*. Funkcija dobiva informaciju o bodovima prije dovršetka punjenja *progressBara*, čeka da se *progressBar* napuni do 100% te označava određeni *radioButton*. Vidljivo je na slici 4.17 (lijevo) kako *progressBar* još nije napunjen te kako ni *radioButton* nije označen, dok je na slici 4.17 (desno) vidljiv označeni *radioButton* čime je ujedno i naznačen raspon bodova.

```
private void setRingProgressBar(){
    ringProgressBar.setOnProgressListener(() → {
            Toast.makeText(getApplicationContext(), text: "Test završen",Toast.LENGTH_SHORT).show();
            String result=Integer.toString(totalSum);
            etResultOfTest.setText(result);
            etResultOfTest.setFocusable(false);
            turnOnRadioButtonAndMessageForEachResult();
    });
    new Thread((Runnable) () → {
            for(int i=0;i<100;i++){
                try {
                    Thread.sleep( millis: 50);
                    myHandler.sendEmptyMessage( what: 0);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
    }).start();
}
```

Slika 4.17 Programski kod s implementacijom progress bara (RingProgressBar)



Slika 4.18 Zaslon s rezultatom upitnika, prije (lijevo) i nakon punjenja progress bara na 100% (desno)

Nakon prikaza rezultata na za to namijenjenim mjestima na zaslonu, potrebno je rezultat pohraniti u baze podataka. Rezultat se pohranjuje i lokalno, ali i u bazi podataka *Firebase*. Razlog iz kojeg se rezultat pohranjuje i lokalno je to što se na taj način pohranjuju podaci koji će biti spremljeni u prethodnim rezultatima upitnika i s pomoću kojih će se iscrtavati graf. To je također moglo biti ostvareno pomoću baze podataka *Firebase*, no ovdje je to ostvareno putem lokalne baze *Realm*. Da bi se podaci mogli pohranjivati lokalno putem baze podataka *Realm* potrebno je prvo inicijalizirati *Realm* što je objašnjeno u potpoglavlju 4.4.4.

Pri samom registriranju korisnika, kreira se i parametar rezultat upitnika (*testResult*) unutar baze podataka *Firebase* koji se nakon registracije postavlja na 0. Kad određeni korisnik ispuni upitnik i odluči pohraniti svoj rezultat, pronalazi se trenutno prijavljeni korisnik te nulu zamjenjuje rezultatom upitnika. Gumb za pohranu također onemogućuje ponovno pohranjivanje ukoliko je pritisnut više od jednom.

Na slici 4.19 vidljiv je programski kod za pohranu rezultata anketiranja. Prilikom spremanja rezultata u lokalnu bazu podataka *Realm*, spremaju se dva parametra, a to su datum i rezultat anketiranja. Funkcija Store omogućuje pohranu rezultata. To je *onClick* metoda, a to znači da će prilikom pritiska na gumb za pohranu, sav kod unutar metode *Store* biti izvršen. Prvo se

pronalazi trenutno prijavljeni korisnik u bazi podataka metodom *getCurrentUser* i u polju namijenjenom za spremanje rezultata upitnika (*testResult*) postavlja vrijednost na ukupnu vrijednost bodova ankete izračunatu pri završetku anketiranja. Također, poziva se funkcija *storeInfoWithRealm* (slika 4.19) koja ima svrhu pohrane trenutnog vremena, tj. datuma i rezultata upitnika.

```
public void Store(View view) {

    String date;
    Date cal = (Date) Calendar.getInstance().getTime();
    date = cal.toLocaleString();

    buttonClick+=1; //testiranje koliko je puta button Store pritisnut
    FirebaseUser firebaseUser=FirebaseAuth.getInstance().getCurrentUser();
    mDatabase=FirebaseDatabase.getInstance().getReference();
    /*tvTherapy.setText(firebaseUser.getUid());*/
    mDatabase.child("Persons").child(firebaseUser.getUid()).child("testResult").setValue(totalSum);
    if(buttonClick==1){
        String message="Vaš rezultat testiranja ("+totalSum+") je pohranjen u bazi podataka.";
        storeInfoWithRealm(date,totalSum);
        //funkcija za pohranu podataka u Realm bazu
        Toast.makeText( context: this, message,Toast.LENGTH_SHORT).show();
    }
    else if (buttonClick>1){
            Toast.makeText( context: this, text: "Rezultat je već spremljen!",Toast.LENGTH_SHORT).show();
        }
    }
```

Slika 4.19 Programski kod za pohranu rezultata anketiranja

Na slici 4.20 prikazan je programski kod za spremanje trenutnog vremena i rezultata ispunjavanja upitnika u lokalnu bazu podataka *Realm*. Parametri funkcije *storeInfoWithRealm* su upravo trenutno vrijeme i rezultat upitnika/ankete. Metodom *getDefaultInstance* dohvaća se trenutna instanca za *Realm* bazu podataka i započinje događaj tj. transakcija na kreiranom *Realm* objektu. Ukoliko još nije stvoren niti jedan podatak u bazi podataka *Realm*, prvi podatak postavlja se na prvo mjesto. Ako već postoji jedan ili više podataka u bazi, sljedeći podatak postavlja se na mjesto iza prethodnog podatka. Podaci se spremaju u klasu kreiranu za spremanje trenutnog vremena i rezultata te se metodom *insertOrUpdate* spremaju u lokalnu bazu podataka *Realm*.

```
private void storeInfoWithRealm(final String date, final int result){
    //pohrana podataka u Realm bazu podataka
    Realm realm=Realm.getDefaultInstance(); //dohvaćanje instance za Realm bazu podataka
    realm.executeTransaction((realm) → {
            Number currentPosition=realm.where(ResultInfo.class).max( fieldName: "position");
            //dobija trenutnu poziciju u bazi podataka
            int nextPosition;
            if(currentPosition==null){
                nextPosition=1;
            }
            else {
                nextPosition=currentPosition.intValue()+1;
            }
            ResultInfo resultInfo=new ResultInfo();
            resultInfo.setDate(date);
            resultInfo.setPosition(nextPosition);
            resultInfo.setResult(result);
            realm.insertOrUpdate(resultInfo);
    });
}
```

Slika 4.20 Spremanje trenutnog vremena i rezultata anketiranja u lokalnu bazu podataka Realm

Tablicom 4.1 prikazana je lista poruka za simptome koji su ispisani ispod raspona, u ovisnosti o kojem rezultatu upitnika se radi, odnosno u koji raspon rezultat upitnika spada.

Tablica 4.1 Lista poruka o simptomima uz priložene raspone bodova

Raspon bodova	**Ispis poruke o simptomima**
<26	Vaši simptomi su blagi ili ih nema!
27-36	Vaši simptomi su blagi!
37-49	Vaši simptomi su zabrinjavajući!
>50	Vaši simptomi su ozbiljni!

4.4.6. Provedba terapije

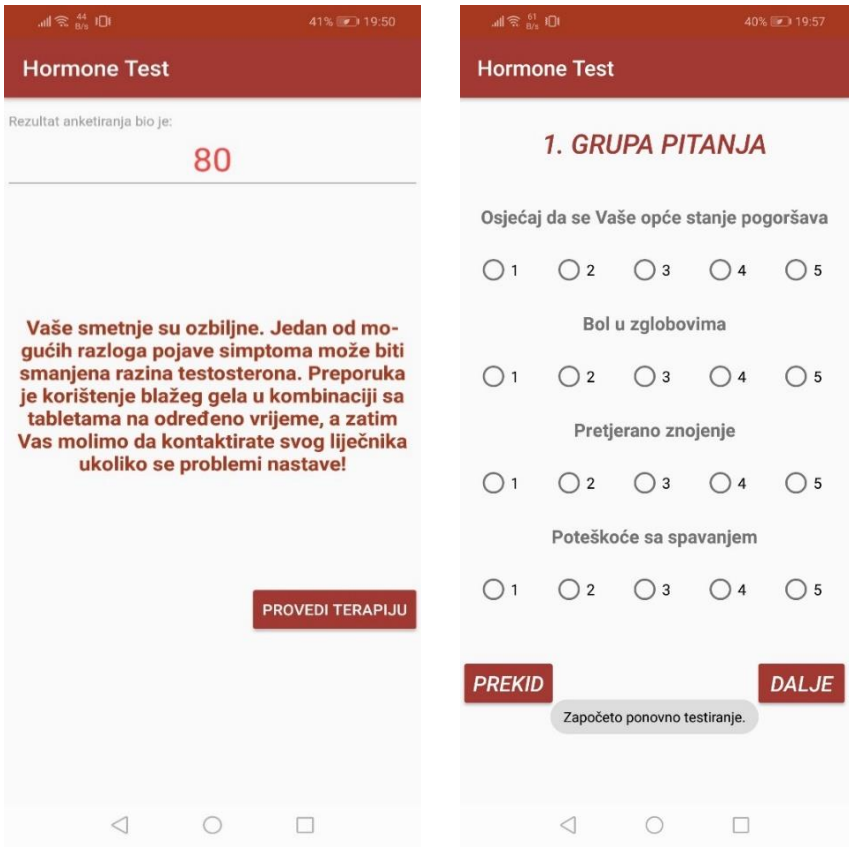
Na osnovu dobivenih rezultata, ukoliko rezultati upućuju na potrebu za provođenjem terapije, moguće je odrediti terapiju. Prije upućivanja na provedbu terapije potrebno je označiti pripadajući *checkBox*. Nakon odabira *checkBoxa*, pritiskom na gumb *Terapija* moguće je izvršiti terapiju. Aplikacija preusmjerava korisnika na zaslon kao na slici 4.20 (lijevo). Na vrhu zaslona nalazi se informacija prenesena iz prethodnog zaslona, a radi se o ukupnom rezultatu upitnika.

Na osnovu rezultata navedenog anketiranja korisnik dobiva određene stručne savjete koje bi trebao uvažiti. Prikaz mogućih poruka u ovisnosti o rezultatu prikazana je tablicom 4.2.

Tablica 4.2 Ispis poruka o terapiji u ovisnosti o rasponu bodova odnosno simptomima

Raspon bodova	**Ispis poruke o potrebnoj terapiji**
<26	Nemate razloga za zabrinutost. Nemate osobitih smetnji. Razviju li se kod Vas dodatni simptomi, molimo Vas da kontaktirate svog liječnika!
27-36	Vaše smetnje su blage. Jedan od mogućih razloga pojave simptoma može biti smanjena razina testosterona. Molimo vas da kontaktirate svog liječnika!
37-49	Vaše smetnje su blage do ozbiljne. Jedan od mogućih razloga pojave simptoma može biti smanjena razina testosterona. Preporuka je korištenje blažeg gela određeno vrijeme, a zatim Vas molimo da kontaktirate svog liječnika ukoliko se problemi nastave!
>50	Vaše smetnje su ozbiljne. Jedan od mogućih razloga pojave simptoma može biti smanjena razina testosterona. Preporuka je korištenje blažeg gela u kombinaciji sa tabletama na određeno vrijeme, a zatim Vas molimo da kontaktirate svog liječnika ukoliko se problemi nastave!

Ovisno o dobivenom stručnom savjetu o potrebnoj terapiji, mogu se poduzeti određene mjere po pitanju zdravlja. Na slici 4.21 (lijevo), prikazan je zaslon terapije za rezultat anketiranja u iznosu od 80 bodova. Poruka je identična poruci koja je napisana u tablici 4.2. Ostale poruke ispisuju se u skladu s rasponom bodova. Ključna uloga aplikacije je signalizirati korisniku na određene poteškoće s testosteronom. Anketu je moguće ponovno provesti, ali to se preporučuje tek nakon određenog vremena korištenja propisane terapije radi provjere trenutnog stanja. Potrebno je dakle poslušati savjete i nakon određenog vremena ponovno provesti anketiranje. Na zaslonu prikazanom na slici 4.21 (desno) ponovno je pokrenuto anketiranje uz priloženu poruku.



Slika 4.21 Zaslon terapije (lijevo) i zaslon u kojem je pokrenuto novo anketiranje (desno)

4.4.7. Komponenta RecyclerView

Prema [13], *RecyclerView* jedna je od bitnih komponenti Androida. Koristi se pri prikazivanju velike količine podataka u obliku liste koju je moguće pomicati gore ili dolje. Njegova najveća odlika je recikliranje vlastitih elemanata što povećava brzinu rada same komponente. Ima sličnu funkcionalnost kao i lista, no jedna od prednosti komponente *RecylcerView* nad njom je korištenje *LayoutManagera* s kojim je moguće direktno utjecati na raspored i ponašanje elemanata liste. Prije korištenja *RecyclerView*-a potrebno je dodati ovisnosti u skripti *Gradle* (slika 4.21), kao što je napravljeno i za obje baze podataka korištene u aplikaciji.

```
implementation 'com.android.support:recyclerview-v7:28.0.0'
```

Slika 4.21 Ovisnost potrebna za korištenje RecyclerView-a

Nakon dodavanja ovisnosti, kreira se *RecyclerView* komponenta u XML-u [14] te dohvaća na zaslonu gdje se spremaju prethodni rezultati ankete. Nakon što je *RecyclerView* dohvaćen, potrebno je na njega postaviti L*ayoutManager* koji upravlja podacima, odnosno njihovom prezentacijom u listi. Postoje *LinearLayoutManager*, *GridLayoutManager* i *StaggeredLayoutManager* i svaki od njih slaže podatke u listu na sebi specifičan način. U razvoju ove aplikacije korišten je *LinearLayoutManager*.

Svaki podatak potrebno je na neki način prezentirati unutar *RecyclerView*-a, pa je stoga potrebno napraviti XML datoteku koja opisuje jedan element liste (slika 4.22). Što je jednostavniji XML, to će *RecyclerView* bolje raditi. Unutar XML dokumenta jednog elementa liste korišten je *CardView* [15] radi urednijeg izgleda između pojedinih elemenata liste.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp"
    app:cardUseCompatPadding="true"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:gravity="center"
        android:paddingStart="16dp"
        android:paddingEnd="16dp">

    <TextView
        android:id="@+id/tvResultList"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:hint="Rezultat"
        android:textAlignment="center"
        android:textSize="20dp" />

    </LinearLayout>

</android.support.v7.widget.CardView>
```

Slika 4.22 XML dokument jednog elementa liste

Da bi se *RecyclerView* u potpunosti osposobio za rad, potrebne su klase *ViewHolder* i *RecyclerAdapter*. *ViewHolder* (slika 4.23) je klasa koja prezentira jedan element *RecyclerView-a*. Upravo se ona koristi kod recikliranja elemenata. *RecyclerView* će napraviti onoliko instanci ViewHolder-a koliko je potrebno da bi se jedan zaslon popunio podacima. Za sve ostale podatke koji se trenutno ne nalaze na zaslonu reciklirat će već postojeći ViewHolder i popuniti ga novim podacima. Pri kreiranju *ViewHolder* objekta potrebno je kroz konstruktor poslati pogled (eng. *view*) koji predstavlja sastavljeni XML dokument. Kreiranje *ViewHolder*-a vrši se u adapteru [19].

Adapter je klasa koja se brine o kreiranju *ViewHolder* klase kao i držanju podataka koji će se prikazivati u *RecyclerView*-u. Postoje tri sustavske metode koje se moraju upotrijebiti kako bi adapter funkcionirao, a to su *onCreateViewHolder*, *onBindViewHolder* i *getItemCount* metode vidljive na slici 4.24. Metoda *onCreateViewHolder* zadužena je za kreiranje i recikliranje

ViewHolder-a. Metoda *onBindViewHolder* omogućuje povezivanje *ViewHolder-a* i elemenata koji se u njemu nalaze, dok metoda *getItemCount* vraća ukupan broj elemenata *RecyclerView* komponente. Osim tri sustavske metode, postoji i dodatno napisana funkcija *addData*. Funkcija *addData* namijenjena je korištenju unutar klase gdje se *RecyclerView* komponenta popunjava podacima. Prilikom poziva te metode, svi elementi *RecylerView*-a bit će obrisani, a zatim će ponovno biti dodani. Na kraju se *RecyclerView* ponovno postavlja metodom *notifyDataSetChanged*. Ukoliko se dodaje ili uklanja jedan podatak u komponenti *RecyclerView*, u tom slučaju praktičnije je koristiti metode *notifyItemInserted* i *notifyItemRemoved*. Također, potrebno je obratiti pozornost i na poziciju koja se izmjenjuje u listi podataka jer ukoliko se pošalje kriva pozicija dolazi do rušenja aplikacije.

```
public class NameViewHolder extends RecyclerView.ViewHolder {
    //klasa koja prezentira jedan element RecyclerVieva, koristi se kod recikliranja elemenata liste

    private TextView tvResultInRecycler;

    public NameViewHolder(@NonNull View itemView) {
        super(itemView);
        tvResultInRecycler=itemView.findViewById(R.id.tvResultList);
    }

    public void setName(String name) { tvResultInRecycler.setText(name); }
}
```

Slika 4.23 ViewHolder klasa RecyclerView-a (pod nazivom NameViewHolder)

```
public class RecyclerAdapter extends RecyclerView.Adapter<NameViewHolder> {
    //klasa koja se brine o kreiranju NameViewHoldera i držanju podataka koji će se prikazati u Recycleru

    private List<String> dataList=new ArrayList<>();

    @NonNull
    @Override
    public NameViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        //metoda zadužena za kreiranje i recikliranje ViewHoldera
        View cellView= LayoutInflater.from(parent.getContext()).inflate(R.layout.one_result_of_list, parent, attachToRoot: false);
        return new NameViewHolder(cellView);
    }

    @Override
    public void onBindViewHolder(@NonNull NameViewHolder nameViewHolder, int position) {
        //metoda za povezivanje ViewHoldera i podataka koji se nalaze u adapteru
        nameViewHolder.setName(dataList.get(position));
    }

    @Override
    public int getItemCount() {  //vraća ukupan broj elemenata Recyclera
        return dataList.size();
    }

    public void addData(List<String> data){
        this.dataList.clear();
        this.dataList.addAll(data);
        notifyDataSetChanged();
    }
}
```

Slika 4.24 Adapter RecyclerView-a

4.4.8. Rezultati prethodno provedenih anketiranja

Nakon što su sve klase za ostvarivanje *RecyclerView* komponente napisane (potpoglavlje 4.4.7), još je potrebno povezati novostvoreni adapter s *RecyclerView* komponentom pomoću metode *setAdapter* (slika 4.25).

```
private void setupRecycler(){
    recyclerView=findViewById(R.id.rvResults);

    btnBackOnTheProfile=(Button)findViewById(R.id.btnBackOnTheProfile);
    btnShowGraph=(Button)findViewById(R.id.btnShowGraph);

    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    //postavljanje layoutManagera koji upravlja podacima u listi, ovdje se koristi LinearLayoutManager

    recyclerAdapter=new RecyclerAdapter();
    recyclerView.setAdapter(recyclerAdapter);
    //povezivanje novonapravljenog adaptera sa Recyclerom
}
```

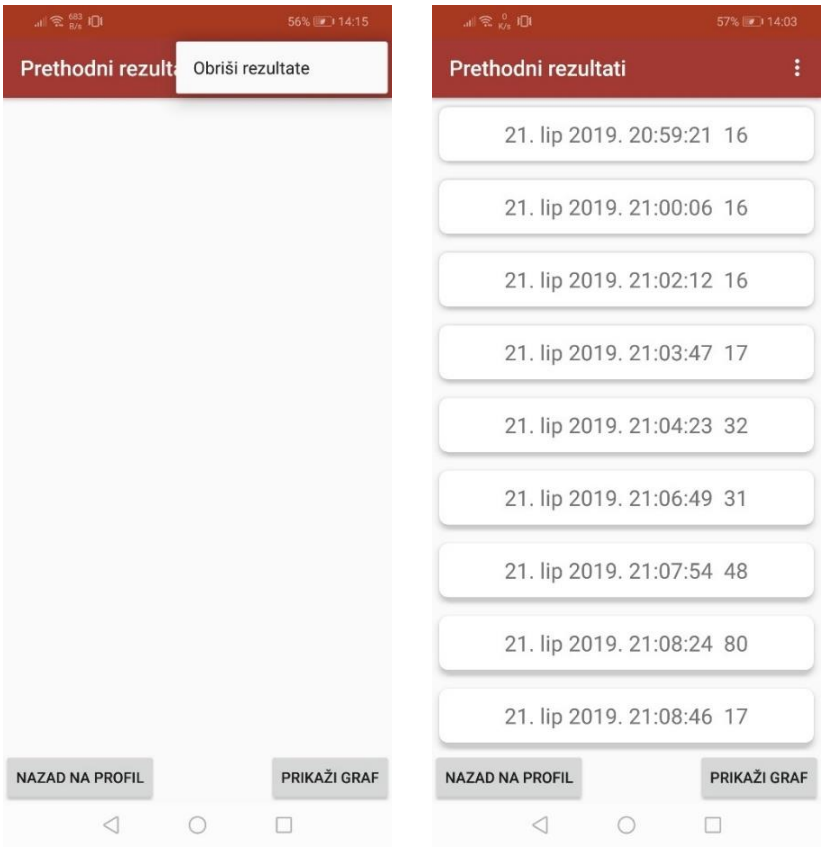
Slika 4.25 Inicijalizacija RecyclerView-a i postavljanje adaptera pomoću setAdapter metode

Nakon što je *RecyclerView inicijaliziran* i postavljen odgovarajući adapter za *RecyclerView*, potrebno je popuniti adapter podacima. Adapter je popunjen s podacima spremljenim u lokalnoj bazi podataka *Realm*. Svi rezultati spremljeni nakon anketiranja pohranjivani su u bazi podataka *Realm*. Svi se podaci (datum i rezultat) dohvaćaju pomoću petlje i popunjavaju adapter [19]. To je ostvareno pomoću koda sa slike 4.26. Nakon dohvaćanja podataka, moguće je pokrenuti potpuno funkcionalni *RecyclerView*.

```
private void getInfoFromRealm(){  //funkcija za uzimanje podataka spremljenih u Realm lokalnoj bazi podataka
    Realm realm=Realm.getDefaultInstance();  //dohvaćanje instance za Realm bazu podataka
    RealmResults<ResultInfo> realmResults=realm.where(ResultInfo.class).findAllAsync();
    //traži sve "objekte" spremljene u Realm bazi podataka
    if(realmResults!=null){
        for(ResultInfo resultInfo:realmResults){
            String dateAndResult=(resultInfo.getDate()+"  "+Integer.toString(resultInfo.getResult()));
            setupRecyclerData(dateAndResult);
        }
    }

}
private void setupRecyclerData(String dateAndResult){  //popunjavanje adaptera podacima
    data.add(dateAndResult);
    recyclerAdapter.addData(data);
}
```

Slika 4.26 Dohvaćanje podataka iz baze podataka Realm i popunjavanje adaptera tim podacima

Nakon više obavljenih anketiranja i spremljenih podataka dobiven je *RecyclerView* kao na slici 4.27.



Slika 4.27 Prazan RecyclerView (lijevo) i RecyclerView popunjen podacima (desno)

Nakon pregleda prethodnih rezultata anketiranja, moguće je prikazati graf (više u potpoglavlju 4.4.10) ili se vratiti na zaslon profila. Osim navedenog, moguće je i obrisati sve podatke u listi putem izbornika (eng. *menu*) i odabira na *Obriši rezultate* vidljivo na slici 4.27 (lijevo). Izbornik je izveden na način prikazan na slici 4.28 dok se metode korištene za njegovu inicijalizaciju i postavljanje mogu vidjeti na slici 4.29.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/itemDeleteResults"
        app:showAsAction="always"
        android:title="Obriši rezultate">
    </item>

</menu>
```

Slika 4.28 Kreiranje izbornika za brisanje rezultata anketiranja

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater=new MenuInflater( context: this);
    menuInflater.inflate(R.menu.results_menu, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.itemDeleteResults:
            if(data.isEmpty()){
                Toast.makeText( context: this, text: "Lista je prazna.",Toast.LENGTH_SHORT).show();
            }
            deleteDataFromRealm();
            data.clear();
            recyclerAdapter.addData(data);   //dodavanje prazne liste koja će biti prikazana (refreshana)
            recyclerAdapter.notifyDataSetChanged();
            return true;

         default: return super.onOptionsItemSelected(item);
    }
}
```

Slika 4.29 Metode potrebne za ostvarivanje izbornika za brisanje rezultata anketiranja

U metodi *onCreateOptionsMenu* pronalazi se XML dokument s definiranim izbornikom (slika 4.28) i povezuje s izbornikom inicijaliziranim u metodi. Zatim se u metodi *onOptionsMenuSelected* provjeravaju slučajevi za svaki element izbornika. S obzirom da na zaslonu s prethodnim rezultatima postoji samo jedan element, i to onaj za brisanje svih rezultata upitnika, provjerava se samo taj element i izvršavaju se naredbe sa slike 4.30. Da bi se podaci obrisali točno u trenutku odabira brisanja, potrebno je obrisati podatke iz lokalne baze podataka *Realm*, zatim isprazniti listu, popuniti adapter praznom listom i obavijestiti adapter da je došlo do promjene podataka.

```
public void deleteDataFromRealm(){
    Realm realm=Realm.getDefaultInstance();
    realm.executeTransaction(new Realm.Transaction() {
        @Override
        public void execute(Realm realm) {
            realm.deleteAll();
        }
    });
}
```

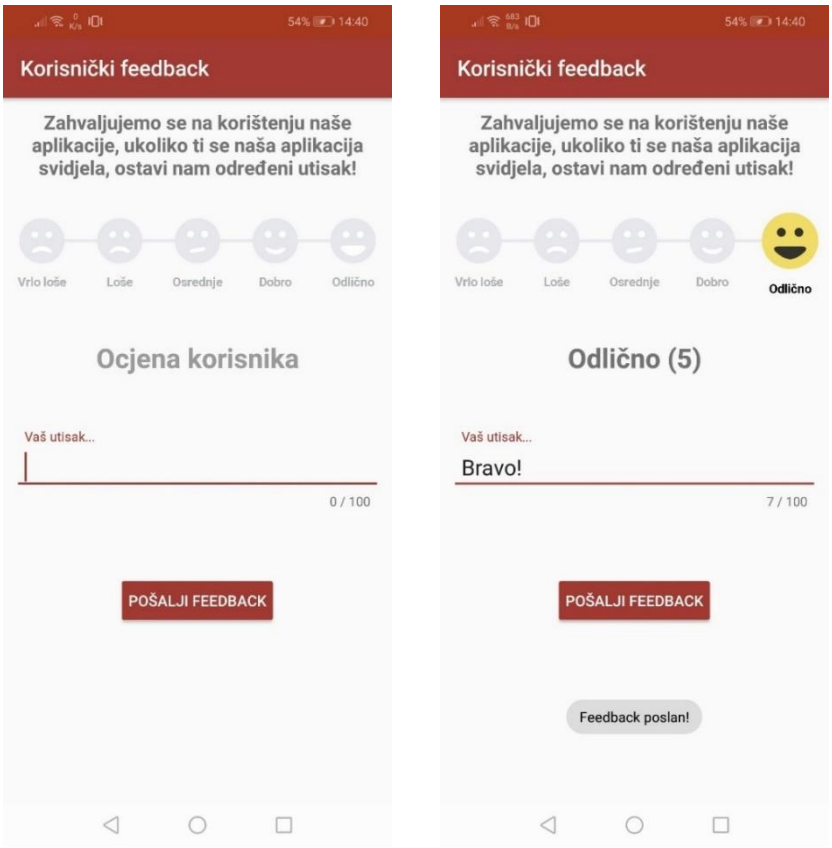
Slika 4.30 Programski kod za brisanje podataka iz lokalne baze podataka Realm

4.4.9. Izbornik za ocjenjivanje aplikacije i detalje o testosteronu

Nakon što je objašnjeno ostvarivanje izbornika u sklopu zaslona za pregled prethodnih rezultata upitnika, važno je spomenuti i izbornik na zaslonu profila koji omogućuje ocjenjivanje

aplikacije kao i saznanje dodatnih informacija o hormonu testosteronu, što je vidljivo na slici 4.6 (desno).

U ovom izborniku postoje dva elementa, od kojih je jedan element za ocjenjivanje aplikacije, a drugi za informiranje o testosteronu. Kod kreiranja izbornika vrijedi ista procedura kao na slici 4.28 samo što su ovdje dva elementa izbornika. Također, prva metoda za pronalazak XML dokumenta s pripadajućim izbornikom ostaje ista kao na slici 4.29, samo što se ovdje dohvaća drugi izbornik. Jedino se razlikuje metoda *onOptionsMenuSelected* gdje je naznačeno što se odvija prilikom kojeg pritiska na element izbornika. Ako se pritisne prvi element za ocjenjivanje, korisnik se preusmjerava na zaslon za ocjenjivanje, ako se odabere drugi element, korisnik se preusmjerava na zaslon s informacijama o testosteronu. Na slici 4.31 vidljivo je kako izgleda zaslon za ocjenjivanje prije i poslije ocjenjivanja aplikacije.



Slika 4.31 Zaslon za ocjenjivanje aplikacije prije (lijevo) i poslije ocjenjivanja (desno)

Za ostvarivanje ocjenjivanja nije korišten klasični *RatingBar*, već *SmileyRatingBar*. Kod *RatingBar*-a koriste se najčešće zvjezdice, dok se ovdje koriste različite ikonice (eng. *smile*) za pojedinu ocjenu od 1-5. Da bi *SmileyRatingBar* bio omogućen, bilo je potrebno implementirati ovisnost u skripti *Gradle* (slika 4.32), a zatim kreirati *SmileyRatingBar* u XML dokumentu i povezati ga s zaslonom na kojem se *RatingBar* i nalazi. *SmileyRatingBar* nudi mogućnost

promjene ikonica, promjenu naziva elementa ispod pojedine ikonice, promjenu boja, postavljanje teksta ispod svake ikonice, itd.

```
implementation 'com.github.sujithkanna:smileyrating:1.6.8'
```

Slika 4.32 Ovisnost za implementiranje SmileyRatingBara

Važno je napomenuti da se prilikom registriranja osim rezultata upitnika, kreira i parametar za ocjenu aplikacije, te parametar za objašnjenje ocjene aplikacije (slika 4.31 desno). Ocjena aplikacije je prilikom registriranja postavljena na nulu, kao i bodovi anketiranja, a objašnjenje ocjene aplikacije bilo je postavljeno na *null*, jer se radi o tekstualnom (eng. *string*) tipu podatka. Nakon ostavljanja određenog utiska, određivanja ocjene i upisa objašnjenja ocjene, potrebno je poslati utisak (eng. *feedback*). Utisak se šalje u bazu podataka *Firebase* te tamo ostaje pohranjen. Vlasnik aplikacije ima mogućnost uvida u podatke te može vidjeti koju je ocjenu korisnik dodijelio i što je napisao u objašnjenju.

Kao i kod pohrane bodova anketiranja, potrebno je pronaći trenutno prijavljenog korisnika, te pod njim spremiti navedene podatke što je vidljivo u programskom kodu na slici 4.33. Metoda *saveRating* je *onClick* metoda koja se aktivira pritiskom na gumb za pohranu utiska aplikacije. Prvenstveno se provjerava ocjena korisnika metodom *getRating*, te ukoliko ona ne postoji, tj. ukoliko nije označena od strane korisnika signalizira korisnika porukom o tome. Također, provjerava se i unos korisničkog obrazloženja aplikacije metodom *validateDesription*. Nakon što je ocjena označena i obrazloženje napisano, pronalazi se trenutno prijavljeni korisnik i podaci se spremaju u bazu podataka *Firebase*.

```
public void saveRating(View view){

    if(view==btnSaveRating){
        validateDescription();
        if(smileRating.getRating()==0){
            Toast.makeText( context: this, text: "Molimo Vas da dodijelite ocjenu!",Toast.LENGTH_SHORT).show();
        }
        else{
        FirebaseUser user= FirebaseAuth.getInstance().getCurrentUser();
        databaseReference= FirebaseDatabase.getInstance().getReference();
        databaseReference.child("Persons").child(user.getUid()).child("rate").setValue(smileRating.getRating());
        databaseReference.child("Persons").child(user.getUid()).child("rateDescription").setValue(etRateDescription.getText().toString());
            //ove četiri linije iznad dohvaćaju trenutno ulogiranog korisnika i pohranjuju vrijednost ocjene i opis koji je korisnik
            // dodijelio aplikaciji

        Toast.makeText( context: this, text: "Feedback poslan!",Toast.LENGTH_SHORT).show();
        }
    }
}
```

Slika 4.33 Pohrana ocjene aplikacije i objašnjenja u bazu podataka Firebase

4.4.10. Grafički prikaz rezultata upitnika

Za mogućnost korištenja grafa unutar Android platforme ponovno je potrebno imati ovisnost koju je potrebno implementirati u skripti *Gradle* (slika 4.34).

```
implementation 'com.jjoe64:graphview:4.2.2'
```

Slika 4.34 Ovisnost potrebna za implementiranje grafa

GraphView je biblioteka u sastavu operacijskog sustava Android koja omogućuje programsko kreiranje fleksibilnih i urednih grafikona. Jednostavno ih je integrirati u sustav, kao i naknadno podešavati. Nakon implementiranja ovisnosti, potrebno je postaviti *GraphView* u XML dokumentu te povezati taj isti *GraphView* s zaslonom gdje će graf biti prikazan. Postoje različite vrste *GraphView*-a, kao što su točkasti grafikon, linijski grafikon, itd. Ovdje je korišten linijski grafikon. Nakon povezivanja *GraphView*-a potrebno je dodati neke podatke u graf. S obzirom da se radi o linijskom grafikonu, kreiran je objekt klase *LineGraphSeries* u kojem su svrstane određene točke. Točke koje su unesene u grafikon, upravo su rezultati prethodnih anketiranja. Na osi Y nalazi se ljestvica za bodove ankete, a na osi X broj anketiranja s početkom u nuli. Moguće je ograničiti maksimalan broj točaka na X osi, tj. maksimalan broj rezultata ankete koji će biti prikazani. Ovdje je to ograničeno na broj 100.

Za popunjavanje točaka grafa potrebno je dohvatiti rezultate iz lokalne baze podataka *Realm*. Napravljena je *for* petlja u kojoj je za svaki broj anketiranja unesena nova točka, odnosno novi rezultat prethodnog anketiranja (slika 4.35).

```
private void getInfoFromRealm(){
    Realm realm=Realm.getDefaultInstance();
    RealmResults<ResultInfo> realmResults=realm.where(ResultInfo.class).findAllAsync();
    if(realmResults!=null){
        int i=0;
        for(ResultInfo resultInfo:realmResults){
                int result = resultInfo.getResult();
                list.add(new DataPoint(i,result));
                i++;
        }
    }
}
```

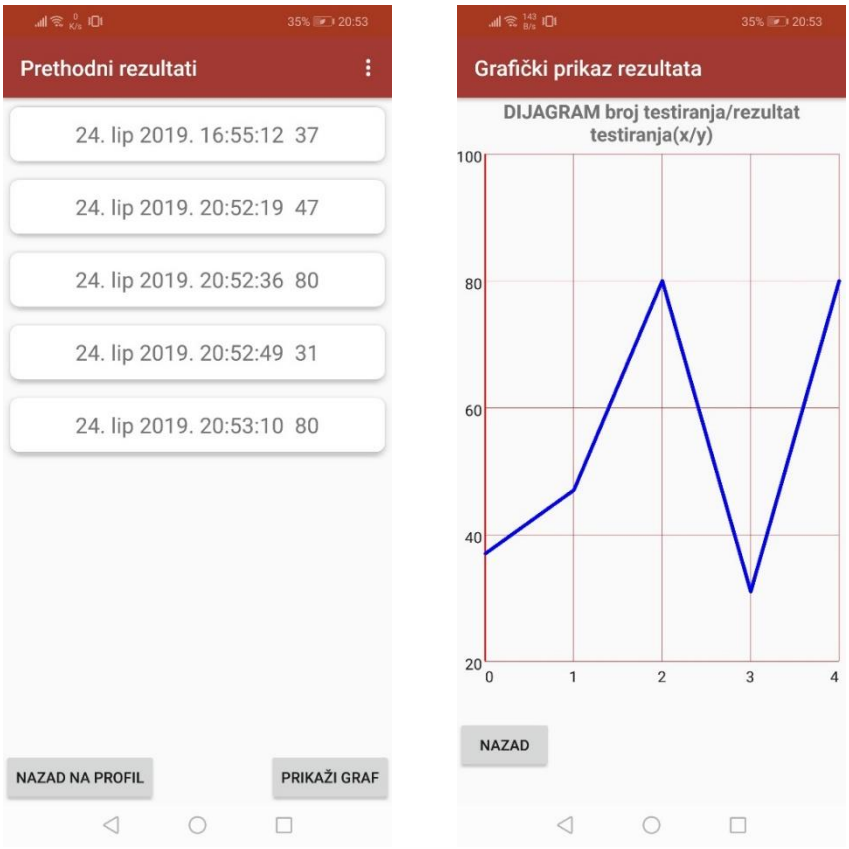
Slika 4.35 Popunjavanje liste točkama koje će biti prikazane na grafu

Nakon što su prethodni rezultati ankete dohvaćeni iz lokalne baze podataka *Realm* i pohranjeni u listu, potrebno je postaviti graf s tim točkama. Postavljanje grafa prikazano je kodom na slici 4.36.

```
private void setGraph(){
    int i;
    for (i=0;i<list.size();i++) {
        series.appendData(list.get(i), scrollToEnd: true, maxDataPoints: 100);
    }
    graph.addSeries(series);
    series.setColor(Color.BLUE);  //boja linije svake točke grafa
    graph.getGridLabelRenderer().setGridColor(Color.RED);   //boja grid-a
    series.setThickness(10);  //postavljanje debljine linije
}
```

Slika 4.36 Postavljanje grafa u potpunosti

Prolaskom kroz *for* petlju, dohvaćena je svaka točka povučena iz lokalne baze *Realm*, a spremljena u listi. Ovdje se također može uvidjeti broj koji je spomenut, a to je broj 100 koji označava maksimalan broj točaka koje će biti prikazane. Postavljena je boja linije u plavu, kao i boja rešetke (eng. *grid*) u crvenu. Povećana je debljina linije s zadnjom linijom koda sa slike 4.36. Postoje još razne mogućnosti uređivanja grafikona, počevši od postavljanja naslova (eng. *title*) osi Y, postavljanja naslova osi X, postavljanja margina za svaki od naslova, itd. Priložena je također slika s grafom uz sliku prethodno spremljenih rezultata anketiranja (slika 4.37).



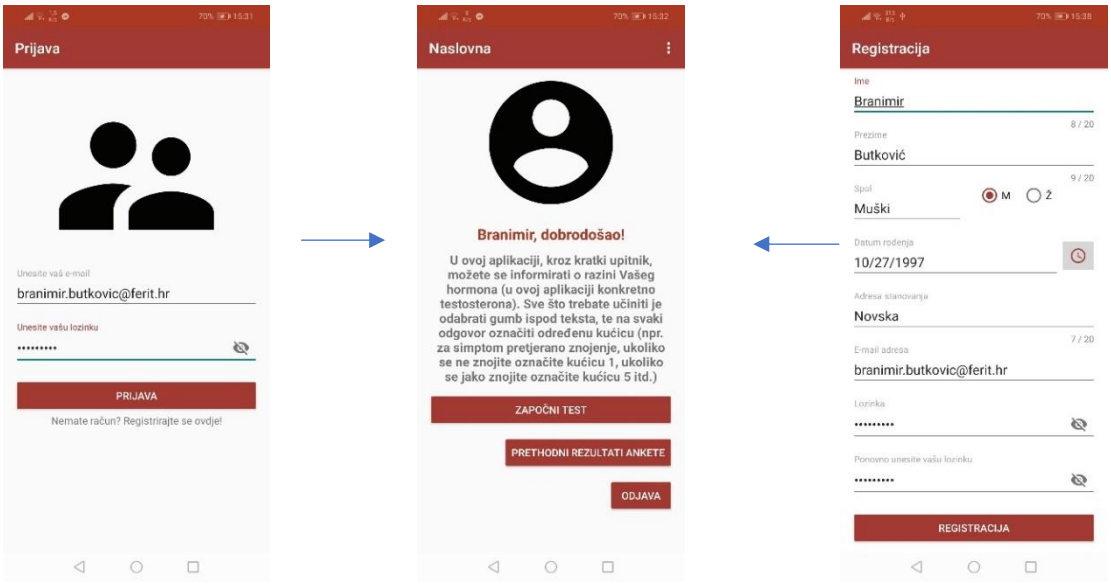
Slika 4.37 Rezultati prethodnih rezultata ankete (lijevo) i graf s priloženim rezultatima (desno)

5. KORIŠTENJE I ISPITIVANJE RADA APLIKACIJE S ANALIZOM REZULTATA

5.1. Način korištenja aplikacije

Iako je način korištenja mobilne aplikacije objašnjen kroz prethodna poglavlja, posebice u potpoglavlju 3.2, u ovom poglavlju sve to je objašnjeno na jednom mjestu. Dakle, korisnik ima mogućnost registriranja u sustav. Prilikom registriranja potrebno je navesti podatke uključujući ime, prezime, datum rođenja, adresu stanovanja, e-poštu i zaporku. Nakon uspješne registracije, podaci se spremaju u bazu podataka *Firebase* te aplikacija preusmjerava korisnika na novokreirani profil. Na profilu postoji mogućnost ocjenjivanja aplikacije te informiranja o testosteronu putem izbornika. Osim izbornika postoji mogućnost odjave iz sustava prilikom koje se ponovno mora izvršiti prijava u sustav putem e-pošte i zaporke korištene prilikom registracije (slika 5.1). Ukoliko se zaporka ili e-pošta zagube, postoji mogućnost resetiranja te korisnik opet ostvaruje mogućnost korištenja aplikacije. Ukoliko se korisnik ne odluči za izbornik ili odjavu iz sustava, postoje još dvije opcije. Jedna od njih je pregled prethodnih rezultata ankete, a druga je pokretanje anketiranja. Anketiranje se pokreće pritiskom na gumb na zaslonu profila. Sastoji se od 16 pitanja podijeljenih kroz četiri zaslona. Nakon ostavljanja utiska na svako od 16 pitanja, aplikacija preusmjerava korisnika na novi zaslon gdje će biti prikazani ukupni bodovi ankete. Na osnovu tih bodova ispisat će se procjena razine testosterona prema rasponu rezultata koji se nalaze u tablici 4.1.

Sljedeće što korisnik može učiniti je pohrana rezultata. Prilikom pohrane, rezultati se spremaju u bazu podataka *Firebase*, točnije u oblak računala i u lokalnu bazu podataka *Realm* gdje će osim rezultata biti pohranjen i datum kad je anketa izvršena. Ukoliko simptomi daju naznaku korisniku da je preporučljiva terapija, odabirom gumba za terapiju omogućeno je upućivanje na terapiju. Iskazane terapije ovisne o rezultatu nalaze se u tablici 4.2. Unutar zaslona gdje su prikazani prethodni rezultati ankete, dohvaćaju se rezultati spremljeni u lokalnoj bazi podataka *Realm* uz pripadajući datum te bivaju prikazani u listi. Ukoliko korisnik želi obrisati navedenu listu, može to učiniti putem odabira brisanja liste iz izbornika. Iz zaslona prethodnih rezultata ankete, korisnik može pogledati i graf koji je temeljen upravo na tim prethodnim rezultatima anketiranja. Graf na osi x prikazuje broj anketiranja, dok na osi y prikazuje rezultat ankete.



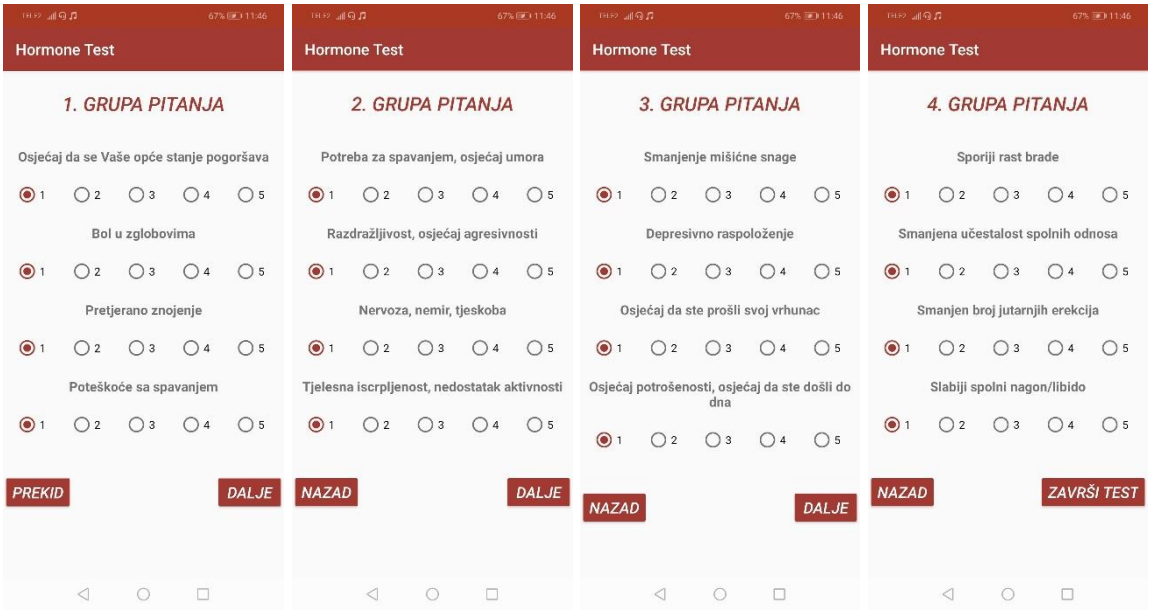
Slika 5.1 Prikaz zaslona za prijavljivanje i registraciju te profil korisnika

5.2. Ispitivanje funkcionalnosti aplikacije

U ovom potpoglavlju prikazano je nekoliko primjera provedbe upitnika s ciljem pokazivanja vjerodostojnosti rada aplikacije u skladu s tablicama 4.1 i 4.2. Prikazana su četiri slučaja kroz četiri naredna potpoglavlja.

5.2.1. Rezultat ankete manji od 26 bodova

Prvi slučaj odnosi se na rezultat ankete manji od 26 bodova. Na slici 5.2 prikazan je korisnički utisak na pitanja, dok je na slici 5.3 vidljiv ishod ankete. Zaključak je da je rezultat zadovoljavajući i mogućnost terapije je isključena.

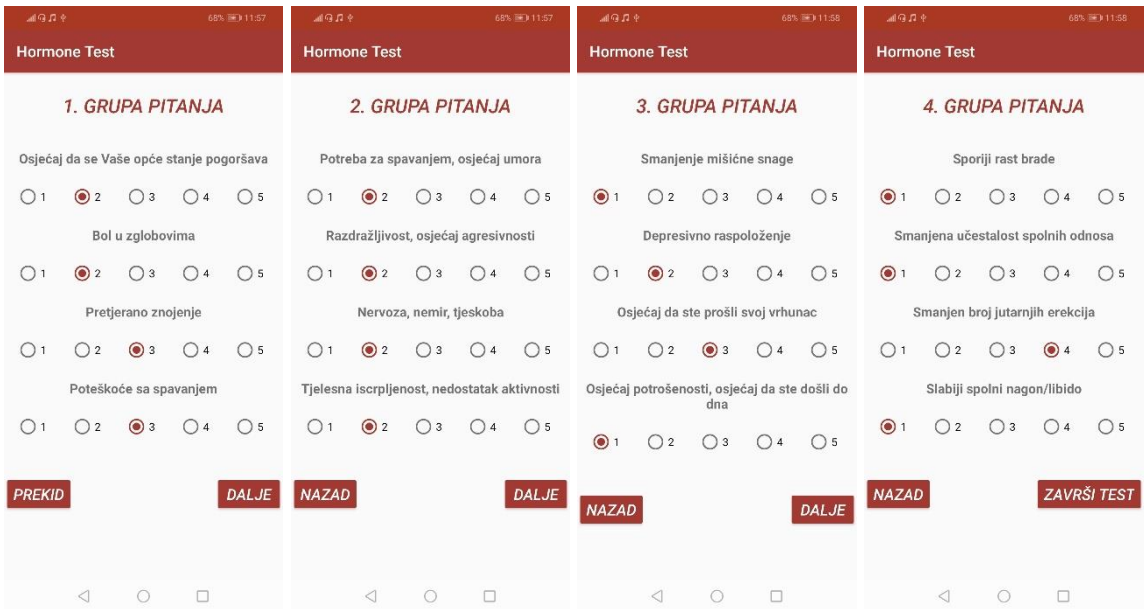


Slika 5.2 Prikaz korisničkog utiska na pitanja za rezultat ankete manji od 26 bodova

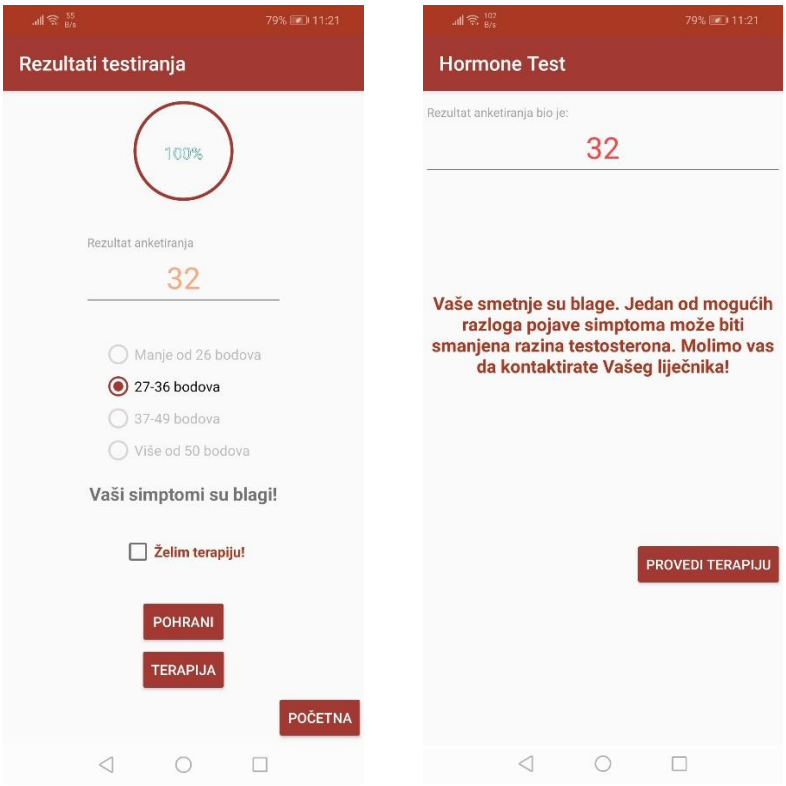
Slika 5.3 Rezultat ankete manji od 26 bodova

5.2.2. Rezultat ankete u rasponu od 27-36 bodova

Drugi slučaj odnosi se na rezultat ankete u rasponu od 27-36 bodova. Na slici 5.4 prikazan je korisnički utisak na pitanja, dok se na slici 5.5 vidi rezultat ankete. Zaključak je da su rezultati u normalnom rasponu, no ipak ih ne treba zanemariti. Postoji mogućnost upućivanja na provedbu terapije putem odabira pripadajućeg *checkBoxa* i gumba za terapiju. Unutar terapije ispisana je poruka ovisno o rezultatu ankete.



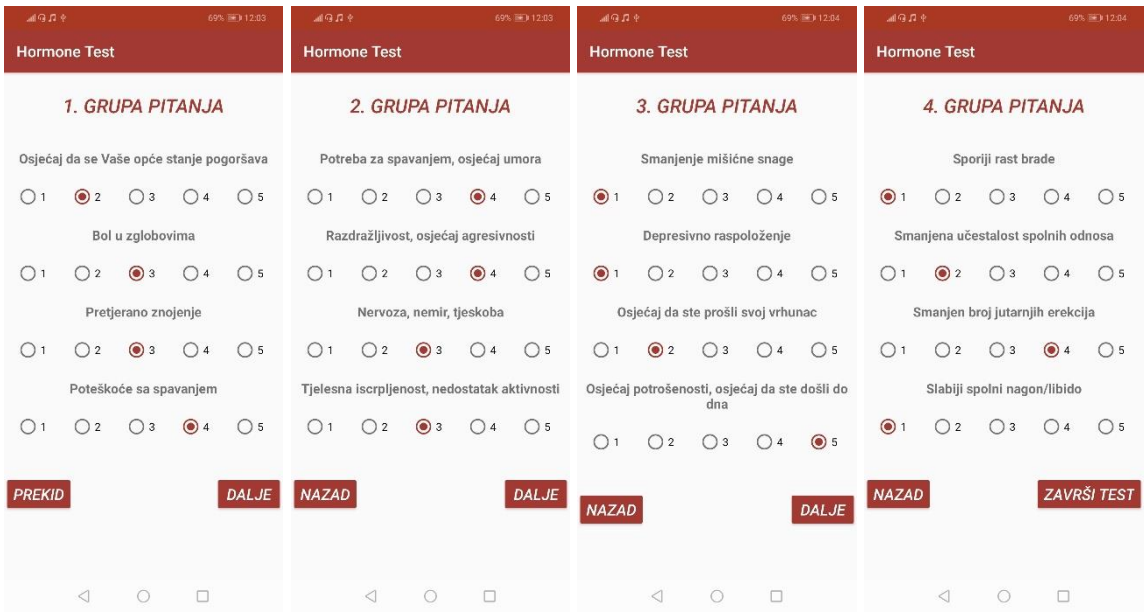
Slika 5.4 Prikaz korisničkog utiska na pitanja za rezultat ankete u rasponu od 27-36 bodova



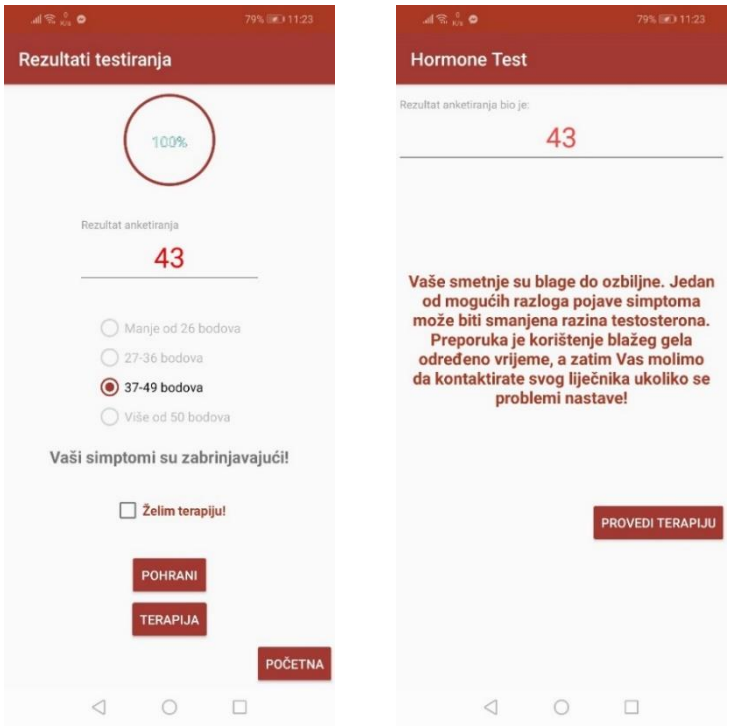
Slika 5.5 Rezultat ankete u rasponu od 27-36 bodova (lijevo) uz pripadajuće savjete za terapiju (desno)

5.2.3. Rezultat ankete u rasponu od 37-49 bodova

Treći slučaj odnosi se na rezultate ankete u rasponu od 37-49 bodova. Na slici 5.6 prikazan je korisnički utisak na pitanja, dok se na slici 5.7 vidi ishod ankete. Zaključak je da su rezultati relativno zabrinjavajući i preporučljivo je obaviti terapiju.



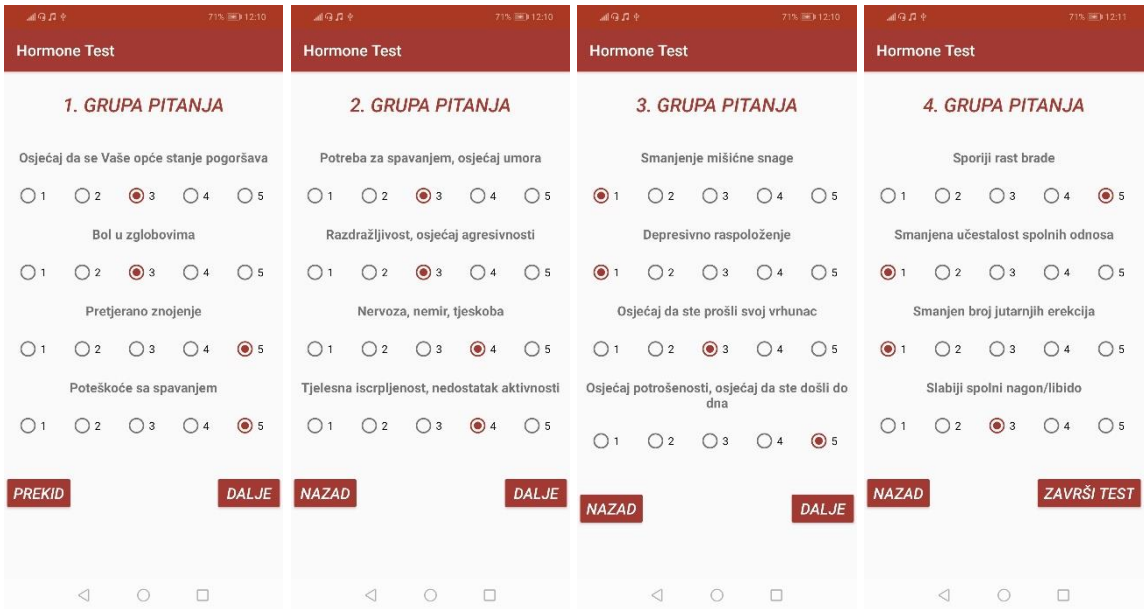
Slika 5.6 Prikaz korisničkog utiska na pitanja za rezultat ankete u rasponu od 37-49 bodova



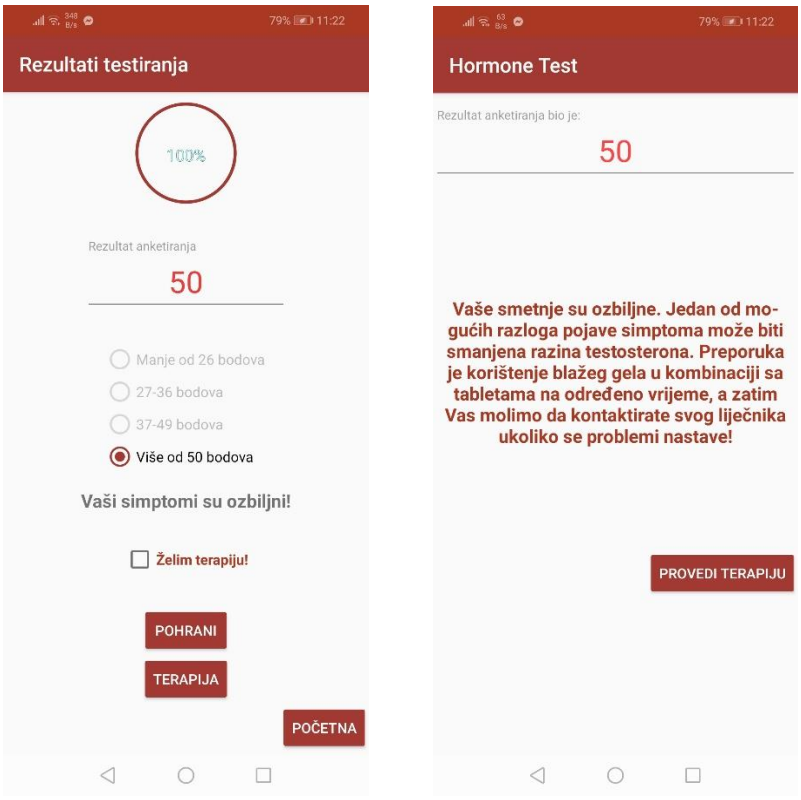
Slika 5.7 Rezultat testiranja u rasponu od 37-49 bodova (lijevo) uz pripadajuće savjete za terapiju (desno)

5.2.4. Rezultat ankete veći od 50 bodova

Četvrti slučaj, ujedno i posljednji, odnosi se na rezultate ankete veće od 50 bodova. Na slici 5.8 prikazan je korisnički utisak na pitanja, dok na slici 5.9 vidimo ishod ankete. Zaključak je da rezultati upitnika ukazuju na ozbiljan nedostatak testosterona i poruku o nužnosti terapije ne treba shvatiti neozbiljno. Preporuka je u sva tri slučaja, osim prvog, poslušati savjete za obavljanje terapije, odnosno posjetiti liječnika.



Slika 5.8 Prikaz korisničkog utiska na pitanja za rezultat ankete veći od 50 bodova



Slika 5.9 Rezultat ankete veći od 50 bodova (lijevo) uz pripadajuće savjete za terapiju (desno)

5.2.5. Usporedba rezultata ankete za sve raspone bodova

Vidljivo je da su simptomi za rezultat ankete manji od 26 bodova blagi ili ih nema. Također, mogućnost provedbe terapije je isključena. Za rezultat ankete u rasponu od 27-36 bodova simptomi su blagi. U ovom slučaju postoji mogućnost provedbe terapije, iako to nije nužno. Za rezultat ankete u rasponu od 37-49 bodova, korisnik ima zabrinjavajuće simptome prilikom čega je u terapiji naznačeno da su blagi do ozbiljni i preporuka je obratiti se liječniku. Za rezultat ankete veći od 50 bodova simptomi su ozbiljni. Provedba terapije je obvezna, kao i kontaktiranje liječnika. Vidljivo je da se promjenom korisničkog utiska na pitanja mijenjaju i ukupni rezultati. Stoga se može konstatirati da aplikacija radi ispravno, odnosno daje očekivane raspone izlaznih vrijednosti za različite ulazne vrijednosti.

6. ZAKLJUČAK

Cilj završnog rada bio je predložiti model vlastitog rješenja mobilne Android aplikacije za praćenje hormonalnih poremećaja za hormon testosteron. Aplikacija pomaže pri otkrivanju nedostatka testosterona i upućuje korisnike na potrebu za provedbom terapije ovisno o tome kolika je razina kritičnosti rezultata upitnika. Putem jednostavnog upitnika, odnosno ankete prikupljaju se procjene simptoma na temelju kojih je moguće procijeniti razinu nedostatka testosterona. Aplikacija je informativnog karaktera i u svakom od prijedloga terapije predložen je posjet liječniku što je uobičajeni medicinski pristup. Analiziranje rada aplikacije obavljeno je za četiri slučaja s različitim rezultatima upitnika, te je pokazano da aplikacija radi ispravno i da ovisno o rasponu bodova, odnosno rezultata upitnika predlaže različite pristupe provedbe ili neprovedbe terapije. Rezultati trenutnog upitnika prikazani su brojčano, a rezultati prethodne provedbe upitnika prikazuju se i grafički. Važno je također spomenuti i moguća proširenja. Naime, aplikacija koristi dvije baze podataka, lokalnu bazu podataka *Realm* i bazu podataka u oblaku, odnosno bazu podataka *Firebase*. Lokalna baza podataka mogla bi se koristiti u druge svrhe, primjerice za spremanje e-pošte i zaporke trenutnog korisnika radi lakše prijave u sustav. Prilikom odjave ili ponovnog ulaska na zaslon prijave može biti omogućeno pamćenje korisničkih podataka i lakša prijava u sustav. Što se tiče ostalih proširenja, ukoliko bi se aplikacija pokazala korisnom bilo bi korisno dodati grafove s ostalim parametrima i time unaprijediti dizajn i preglednost aplikacije. Također, na temelju ove aplikacije moguće je programski omogućiti i analiziranje razine nekih drugih hormona.

LITERATURA

[1] M. Bošnjak, Testosteron, Sveučilište u Splitu, Prirodoslovno-matematički fakultet – Odjel za kemiju, May 2016, https://zir.nsk.hr/islandora/object/pmfst%3A143 (posjećeno 29.6.2019.)

[2] N. Šimić, Sex hormones and cognitive functioning of women, Archives of Industrial Hygiene and Toxicology, Vol. 60, Issue 3, May 2009, pp. 363-374.

[3] Building Body, 7 kritičnih znakova koji ukazuju na nizak testosteron, December 2016. https://www.building-body.com/7-kriticnih-znakova-ukazuju-nizak-testosteron/ (posjećeno 30.6.2019.)

[4] AndroGel Smpc, Solvay Pharma, November 2016

[5] K. Toljan, E. Bokulić, T. Katić, S. Mikulec, B. Tomić, Hormoni i ponašanje, Sveučilište u Zagrebu, Medicinski fakultet, godina.

[6] ADAM test, Bayer AG, August 2016, https://www.nebido.com/en/patients/check-yourself/adam-test/ (posjećeno 08.09.2019.)

[7] Android, Android Studio https://developer.android.com/studio/intro/ (posjećeno: 23.06.2019.)

[8] S. Mijalkovic, NoSQL Baze podataka http://poincare.matf.bg.ac.rs/~vladaf/Courses/Matf%20MNSR/Prezentacije%20Individualne/Mijalkovic_NoSQL_baze_podataka.pdf (posjećeno: 24.06.2019.)

[9] L. Moroney, The Firebase Realtime Database, https://link.springer.com/chapter/10.1007/978-1-4842-2943-9_3 (posjećeno: 24.06.2019.)

[10] JSON, Journal of Computational Information Systems, https://www.researchgate.net/profile/Dunlu_Peng/publication/265874991_Using_JSON_for_Data_Exchanging_in_Web_Service_Applications/links/5523cd1f0cf2c815e07325ea.pdf (posjećeno: 24.06.2019.)

[11] Firebase, Google Developers, https://firebase.google.com/docs/database/ (posjećeno: 23.06.2019.)

[12] Realm Database, Realm, https://realm.io/docs/java/latest/ (posjećeno: 23.06.2019.)

[13] RecyclerView, Android Developers, https://developer.android.com/reference/android/support/v7/widget/RecyclerView (posjećeno: 24.06.2019.)

[14] XML (Extensible Markup Language), Android Developers, https://developer.android.com/guide/topics/ui/declaring-layout (posjećeno: 24.06.2019.)

[15] CardView, Android Developers, https://developer.android.com/reference/android/support/v7/widget/CardView (posjećeno: 24.06.2019.)

[16] L. Kraus, Programski jezik Java sa rešenim zadacima, September 2013.

[17] N. Gandhewar, R. Sheikh, Google Android: An Emerging Software Platform For Mobile Devices, 2010.

[18] Add build dependecies, Android Developers, https://developer.android.com/studio/build/dependencies

[19] B. Zorić, Osnove razvoja web i mobilnih aplikacija, RecyclerView, 2018.

[20] G. Rastrelli, F. Guaraldi, Y. Reismann, A. Sforza, A. Isidori, M. Maggi, G. Corona, Testosterone Replacement Therapy for Sexual Symptoms, July 2019.

SAŽETAK

Cilj ovog završnog rada je stvaranje Android mobilne aplikacije za praćenje nedostatka i predlaganje provedbe terapije nadoknade testosterona. Mobilna aplikacija omogućuje korisniku provjeru trenutnog stanja hormona testosterona putem kratkog upitnika. Korisniku je omogućeno registriranje i prijava u sustav, te korištenje baza podataka za rukovanje korisnicima i pohranu rezultata. Prikaz rezultata upitnika o nedostatku testosterona na temelju ulaznih parametara omogućen je brojčano i grafički, a posebno je zanimljiva mogućnost pregleda rezultata prethodnih upitnika, te rezultata upitnika nakon moguće provedbe terapije. Ulazni parametri su korisnikove procjene intenziteta simptoma koji kroz zbrojeve ukupnih bodova upitnika ukazuju na moguću razinu nedostatka testosterona, a na temelju njih aplikacija prikazuje procjenu razine ozbiljnosti simptoma, te sugerira potrebu za nadomjesnom terapijom putem posjeta liječniku. Ispravnost rada aplikacije potvrđena je kroz četiri različita ispitna slučaja procjene razine simptoma koji ukazuju na nedostatak testosterona koji su rezultirali s ispravnim ukupnim bodovima odgovarajućih razina nedostatka testosterona.

Ključne riječi: baza podataka, mobilna aplikacija, testosteron, upitnik o nedostatku testosterona.

ABSTRACT

The aim of this paper is creating an Android app for tracking the deficiency in testosterone and recommending suitable therapy. The app offers to check the momentary testosterone levels via short questionare. The user is also allowed to registrate and sign in, and to use the data base and results storage. The results of the questionare on testosterone based on the valuation input are potrayed both numerally and graphically. The ability to check the results of the previously taken questionares and the results after the therapy has been conducted is a very interesting and useful feature. The valuation input are the users estimation of the intensity of symptoms that, when answered, carry a definite point value which can suggest the testosterone deficiency. Based on these, the app decides on severity of the problem and suggest the suited recovery treatment and, of course, a visit to doctor's. The correct functioning of the app is approved via four different examinations of the assessment of the symptoms suggesting the testosterone deficiency resulting in correct results which correspond to the compatible levels on testosterone deficiency.

Key words: database, mobile application, testosterone, questionare about testosterone deficiency

ŽIVOTOPIS

Branimir Butković rođen je 27. listopada 1997. godine u Zagrebu, Hrvatska. Stanuje u okolici Novske, u mjestu Stara Subocka. 2004. godine započinje osnovnoškolsko obrazovanje u OŠ Josipa Kozarca, Stara Subocka. Nakon odličnog uspjeha u osnovnoj školi ostvaruje direktan upis u Tehničkoj Školi Kutina. 2016. godine završava srednjoškolsko obrazovanje odličnim uspjehom i ostvaruje izravan upis na preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Trenutno završava treću godinu i planira nastaviti studiranje na nekom od diplomskih studija. Posjeduje vozačku dozvolu B kategorije. Poznaje strukture programskih jezika poput C, C++, uključujući Javu i Android. Najveći interes pokazuje u dizajniranju mobilnih aplikacija.

PRILOZI

Prilog 1. Dokument rada

Prilog 2. Pdf rada

Prilog 3. Programski kod Android mobilne aplikacije