Automatizirano testiranje web aplikacije za procjenu rizika obolijevanja od osteoporoze primjenom alata Selenium

Majdandžić, Dominik

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

Sveučilišni preddiplomski studij računarstva

AUTOMATIZIRANO TESTIRANJE WEB APLIKACIJE ZA PROCJENU RIZIKA OBOLIJEVANJA OD OSTEOPOROZE PRIMJENOM ALATA SELENIUM

Završni rad

Dominik Majdandžić

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 19.09.2019.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Dominik Majdandžić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3950, 27.09.2019.
OIB studenta:	79945495999
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Automatizirano testiranje web aplikacije za procjenu rizika obolijevanja od osteoporoze primjenom alata Selenium
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog 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:	19.09.2019.
Datum potvrde ocjene Odbora:	25.09.2019.

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, 15.09.2020.

Ime i prezime studenta:	Dominik Majdandžić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3950, 27.09.2019.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Automatizirano testiranje web aplikacije za procjenu rizika obolijevanja od osteoporoze primjenom alata Selenium**

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

Osteoporoza je najčešći uzrok slomljenih kostiju kod starijih osoba. Ona obično nema nikakve simptome skroz dok ne dođe do loma kosti. Iz tog razloga potrebno je izraditi način procjene rizika obolijevanja na temelju čimbenika rizika (kao što su dob, rasa, težina itd.), a ne na temelju simptoma. Tim načinom je moguće procijeniti rizik da će osoba u budućnosti oboljeti od osteoporoze te preporučiti na koji način osoba može smanjiti taj rizik.

Cilj ovoga završnog rada je izraditi web aplikaciju za procjenu rizika obolijevanja od osteoporoze te ju ispitivati. Za ispitivanje web aplikacije potreban je ogroman broj sati kako bi se osiguralo da je ona potpuno funkcionalna. Prije se to izvodilo ručnim ispitivanjem gdje je bilo potrebno ručno ponavljati stotine različitih testnih slučajeva za svaku promjenu aplikacije te zabilježiti dijelove koji ne rade i pokušati odrediti izvor kvara. Kako bi se olakšalo i poboljšalo ispitivanje, razvilo se automatizirano ispitivanje pomoću kojeg se mogu pisati razni testni slučajevi te se jednostavno pokrenuti nakon svake promjene aplikacije. Ovaj način ispitivanja je brži, ima manju vjerojatnost da će doći do greške i koristi manje ljudskih resursa. Iz tog razloga u ovom radu će se web aplikacija ispitivati pomoću automatiziranog ispitivanja. Osim što će se na web aplikaciji za procjenu rizika obolijevanja od osteoporoze objasniti razvoj web aplikacije, također će se na njoj prikazati postupak ispitivanja web aplikacije pomoću Selenium-a.

U drugom poglavlju rada objašnjen je općeniti postupak razvoja web aplikacije zajedno s objašnjenjem različitih vrsta ispitivanja. Dodatno je objašnjeno automatizirano ispitivanje web aplikacija zajedno s alatima koji se mogu koristiti za automatizirano ispitivanje. Na kraju drugog poglavlja je objašnjen utjecaj ispitivanja web aplikacije na njen razvoj. U trećem poglavlju dana je definicija osteoporoze zajedno s njenim uzrocima, simptomima i čimbenicima rizika. Nakon toga su prikazana postojeća programska rješenja za procjenu rizika obolijevanja od osteoporoze te je prikazano programsko rješenje koje će se koristiti u ovom radu. Zatim su navedeni i objašnjeni jezici i alati koji su korišteni za izradu web aplikacije zajedno s primjerima koda i korištenja aplikacije. U četvrtom poglavlju je objašnjen alat Selenium što podrazumijeva: povijest razvoja Selenium-a, Selenium RC, Selenium IDE, Selenium Grid i Selenium WebDriver. Nakon toga je objašnjen plan ispitivanja zajedno s izradom testnih scenarija pomoću Selenium WebDriver-a i Selenium IDE-a. Na kraju se analiziraju dobiveni rezultati ispitivanja te predlažu načini kako poboljšati web aplikaciju.

1.1. Zadatak završnog rada

U teorijskom dijelu rada treba objasniti alat Selenium i njemu pripadajuće programske tehnologije i alate, te mogućnosti automatiziranog ispitivanja web aplikacije. Nadalje, treba predložiti model i predložak arhitekture, te programski ostvariti web aplikaciju (s bazom podataka, mogućnošću registriranja i prijave korisnika, analizom rizika i prikazom rezultata) za procjenu rizika obolijevanja od osteoporoze. Također, treba napraviti plan ispitivanja, definirati testne scenarije, te napisati i provesti testove izrađene web aplikacije. Rezultate testova potrebno je prikladno analizirati i interpretirati, te predložiti i provesti moguća poboljšanja web aplikacije.

2. OPĆENITO O RAZVOJU I ISPITIVANJU WEB APLIKACIJA

2.1. Razvoj web aplikacija

Razvoj web aplikacija je izrada aplikacija koje se nalaze na udaljenim poslužiteljima i isporučuju se korisnikovom uređaju putem interneta. Krajnji korisnik može pristupiti web aplikaciji putem web preglednika kao što su Google Chrome, Mozilla Firefox, Opera itd. Razvoj web aplikacija obično ima kratak životni ciklus razvoja koji vodi mali razvojni tim. *Front end* razvoj aplikacije se provodi programiranjem na strani klijenta gdje je klijent računalna aplikacija poput web preglednika. Programi na strani klijenta obično koriste HTML, CSS i JavaScript. HTML programiranje će uputiti preglednik kako prikazati sadržaj aplikacije na zaslonu, dok CSS prikazuje taj sadržaj u ispravnom formatu. JavaScript će pokretati kod na web stranici, čineći time dio sadržaja interaktivnim. Osim programiranjem na strani klijenta postoji programiranje na strani poslužitelja koje se koristi za stvaranje skripti koje web aplikacije koriste. Skripte se mogu pisati na više jezika kao što su Ruby, Java i Python. Skripta na strani poslužitelja stvorit će prilagođeno sučelje za krajnjeg korisnika te sakriti izvorni kod koji čini sučelje. Tijekom pisanja koda koriste se programske biblioteke koje sadrže već napisane funkcije i klase čime se ubrzava i olakšava razvoj aplikacije. Također se koristi *framework* za razvoj web aplikacija koji se sastoji od već navedenih programskih biblioteci, komponenti i raznih alata organiziranih u arhitektonskom sustavu koji omogućuje programerima da izrade i održavaju složene projekte aplikacija koristeći brz i učinkovit pristup. Kako bi se dodatno olakšalo održavanje koda programeri se pridržavaju pravila i standarda za strukturu mapa, nazive datoteka, organizaciju datoteka, oblikovanje i uvlačenje, logiku kodiranja itd. Zadnji bitni dio razvoja web aplikacije je konzistentno ispitivanje te aplikacije kako bi se osiguralo da pravilno funkcionira.

2.2. Osnove ispitivanja programske podrške

Općenito, ispitivanje otkriva koliko dobro nešto funkcionira. U razvoju računalnog sklopovlja i programske podrške ispitivanje se koristi na ključnim kontrolnim točkama u cjelokupnom procesu kako bi se utvrdilo ispunjavaju li se ciljevi. Ispitivanje programske podrške također pomaže u prepoznavanju pogrešaka, praznina ili nedostajućih zahtjeva suprotno stvarnim zahtjevima. Ono se može obavljati ručno ili pomoću automatiziranih alata. U ovome radu web aplikacija će se ispitivati koristeći automatizirano ispitivanje.

2.2.1. Ručno ispitivanje

Ručno ispitivanje [1] je postupak ručnog ispitivanja programske podrške kako bi se pronašli nedostatci. Tester treba imati perspektivu krajnjih korisnika i osigurati da sve značajke pravilno funkcioniraju. Ovim načinom ispitivanja, testeri izvršavaju testne slučajeve i generiraju izvještaje ručno bez korištenja alata za automatizaciju. Osim što je ovaj način skup i spor, također je sklon ljudskim pogreškama koje bi mogle dodatno usporiti ispitivanje.

2.2.2. Automatizirano ispitivanje

Automatizirano ispitivanje [2] je postupak ispitivanja programske podrške pomoću alata za automatizaciju kako bi se pronašle greške. U ovom načinu ispitivanja testeri izvršavaju testne skripte i automatski generiraju rezultate ispitivanja pomoću korištenog alata za automatizaciju. Testne skripte se mogu jako razlikovati u složenosti od provjere pojedinačne metode unutar klase do provjere da izvođenje niza složenih radnji u korisničkom sučelju dovodi do očekivanih rezultata. Za razliku od ručnog ispitivanja, automatizirano ispitivanje je brže, pouzdanije i jeftinije.

2.2.3. Ispitivanje jedinica koda

Ispitivanje jedinica koda [3] predstavlja razinu ispitivanja programske podrške na kojoj se ispituju pojedine komponente programske podrške. Svrha je provjere da svaka jedinica (engl. *unit*) programske podrške radi kako je zamišljeno gdje je jedinica najmanji ispitivani dio bilo koje programske podrške. Jedinica obično ima jedan ili nekoliko ulaza i obično jedan izlaz. U proceduralnom programiranju jedinica može biti pojedinačni program, funkcija, postupak itd. U objektno orijentiranom programiranju najmanja jedinica je metoda koja može pripadati osnovnoj klasi, apstraktnoj klasi ili izvedenoj klasi.

2.2.4. Funkcionalno ispitivanje

Funkcionalni testovi [4] su usredotočeni na poslovne zahtjeve aplikacije. Sve funkcionalnosti se ispituju na način da se preda odgovarajući ulaz kako bi se provjerilo odgovara li stvarni izlaz očekivanom izlazu ili ne. Ovaj način ispitivanja ne bavi se načinom na koji se proces odvija, već rezultatima procesa. On simulira stvarnu upotrebu sustava, ali ne daje nikakve pretpostavke o strukturi sustava.

2.2.5. Integracijsko ispitivanje

Integracijsko ispitivanje [5] je proces ispitivanja povezanosti ili prijenosa podataka između nekoliko ispitivanih modula. U ovome načinu ispitivanja pojedine jedinice programske podrške se kombiniraju i ispituju kao grupa. Na primjer, može se ispitivati interakcija s bazom podataka. Ovi tipovi testova su skuplji za izvoditi jer zahtijevaju da više dijelova aplikacije bude u pogonu.

2.2.6. Ispitivanje performansi

Ispitivanje performansi [6] je proces određivanje brzine, odziva i stabilnosti računala, mreže, programa ili uređaja pod radnim opterećenjem. Ovi testovi su nefunkcionalni i mogu imati različite oblike da bi se razumjela pouzdanost, stabilnost i dostupnost platforme. Na primjer, mogu se promatrati vremena odziva velikog broja zahtjeva ili vidjeti kako se sustav ponaša sa značajnom količinom podataka. Po svojoj prirodi ovakvi testovi su skupi za implementaciju i pokretanje.

2.2.7. Ispitivanje prihvatljivosti

Ispitivanje prihvatljivosti [7] je procjena usklađenosti sustava s poslovnim zahtjevima i procjena je li sustav prihvatljiv za isporuku. Testovi prihvatljivosti zahtijevaju da bude pokrenuta čitava aplikacija i usredotočeni su na repliciranje ponašanja korisnika, ali također mogu i mjeriti rad sustava te odbiti promjene ako ne ispune određene kriterije.

2.2.8. Ispitivanje sustava

Ispitivanje sustava [8], poznato kao i krajnje ispitivanje scenarija (engl. *end to end scenario testing*), predstavlja razinu ispitivanja na kojoj se ispituje cjelovita i integrirana programska podrška. Ovaj način ispitivanja replicira ponašanje korisnika s programskom podrškom u kompletnom aplikacijskom okruženju. Svrha ovog testa je procijeniti usklađenost sustava s navedenim zahtjevima. Testovi sustava su vrlo korisni, ali skupi za izvođenje i teški su za održavanje kada su automatizirani.

2.3. Specifičnosti ispitivanja web aplikacija

Ispitivanje web aplikacija se odvija kroz šest glavnih koraka [9]: ispitivanje funkcionalnosti, ispitivanje upotrebljivosti, ispitivanje sučelja, ispitivanje kompatibilnosti, ispitivanje performansi i ispitivanje sigurnosti.

Funkcionalnim ispitivanjem se simulira stvarna upotreba sustava. Ideja je približiti se stvarnom korištenju sustava i stvoriti testne scenarije koji se odnose na zahtjeve korisnika. Na primjer,

provjera svih poveznica, ispitivanje svih obrazaca, ispitivanje kolačića (engl. *cookies*), ispitivanje baze podataka itd. Ispitivanje upotrebljivosti unapređuje ispitivanje funkcionalnosti tako da ga kombinira s ispitivanjem cjelokupnog korisničkog iskustva. Ovim načinom osim što se provjerava funkcionalnost aplikacije također se osigurava da je aplikacija jednostavna za korištenje. Ispitivanjem sučelja se osigurava da sve interakcije između sučelja web poslužitelja i aplikacijskog poslužitelja teku bez problema. To uključuje provjeru komunikacijskih procesa kao i provjeru da su poruke o pogreškama ispravno prikazane. Nadalje, ispituje pravilno postupanje s prekidima od strane korisnika i od strane poslužitelja. Idući ključni korak u ispitivanju web aplikacije je osiguravanje kompatibilnosti aplikacije sa svim preglednicima i uređajima. Osim što se provjerava kompatibilnost s različitim preglednicima potrebno je i provjeriti kompatibilnost s različitim inačicama tih preglednika. Također potrebno je ispitivati kompatibilnost s različitim operativnim sustavima. Nakon što se osigura da aplikacija ispravno funkcionira na svim preglednicima i uređajima, potrebno je provjeriti kako ona djeluje pod velikim opterećenjem. Ovo uključuje ispitivanje aplikacije pod različitim brzinama interneta i kako se ona ponaša pod normalnim i najvećim opterećenjima. Posljednji korak u ispitivanju web aplikacija je osiguravanje da je aplikacija zaštićena od neovlaštenog pristupa i štetnih radnji putem virusa ili druge zlonamjerne programske podrške. Praćenjem ovih šest glavnih koraka ispitivanja web aplikacije moguće je lagano pronaći većinu grešaka koje se zatim mogu ispraviti prije nego što to bude prekasno.

2.4. Automatizirano ispitivanje web aplikacija

Automatizirano ispitivanje web aplikacija [10] je postupak u kojem se koriste različite programske podrške za ocjenu performansi web aplikacije. Proces pojednostavljuje i standardizira parametre ispitivanja web aplikacije za izmjene koje se događaju tijekom razvojne faze, štedeći tako resurse i pružajući konzistentne rezultate vlasnicima i administratorima. Iako automatizirano ispitivanje štedi vrijeme i resurse prije prelaska na njega moraju se uzeti u obzir početni troškovi automatizacije, jer obično se zahtijeva postavljanje alata za automatizaciju i pisanje skripti za što su potrebni kompetentni ljudi i vrijeme. Također automatizirano ispitivanje nije rješenje za rane faze razvoja web aplikacije kada se ona može toliko dramatično promijeniti da će se potrošiti previše resursa na održavanje skripte za automatizaciju. Osim toga, automatizirano ispitivanje ima samo pozitivne razloge za prelazak na njega (npr. veća preciznost, brže izvođenje testova, jeftinije itd.).

2.5. Alati za automatizirano ispitivanje web aplikacija

2.5.1. Alat Selenium

Selenium [11] je alat otvorenog koda koji se smatra industrijskim standardom za automatizirano ispitivanje web aplikacija. On nudi fleksibilnost koju većina drugih alata i *framework*-a ne nude. Korisnici mogu pisati testne skripte pomoću raznih programskih jezika (npr. Java, Python, C#, PHP, Ruby, itd.) te ih pokretati na različitim operacijskim sustavima (Windows, Linux, Mac) i internetskim preglednicima (Chrome, Firefox, IE, ...). Negativna strana Seleniuma je ta što korisnici trebaju imati napredno znanje iz programiranja te moraju provesti značajnu količinu vremena za razvoj *framework*-a i biblioteka potrebnih za automatizaciju.

2.5.2. Alat TestComplete

TestComplete [12] se koristi za ispitivanje web, mobilnih, *desktop* aplikacija. Omogućuje nam pisanje testnih skripti pomoću različitih programskih jezika kao što su JavaScript, Python i WBScript. TestComplete je osobito koristan kod ispitivanja aplikacija koje imaju dinamička korisnička sučelja koja se često izmjenjuju zbog ugrađenog mehanizma za prepoznavanje objekata koji može precizno odrediti elemente dinamičkog korisničkog sučelja. Također nam nudi mogućnost snimanja i pokretanja testova. Kao proizvod SmartBear-a, TestComplete se može lagano integrirati s ostalim proizvodima koje nudi SmartBear.

2.5.3. Alat IBM Rational Functional Tester (RFT)

IBM RFT [13] je alat za automatizirano ispitivanje dizajniran za ispitivanje aplikacija razvijene pomoću različitih jezika kao što su .Net, Java, Visual Basic, Ajax, Sap itd. On je također testna platforma na temelju podataka (engl. *data-driven testing platform*) korištena za regresijsko i funkcionalno ispitivanje. Jedan od jedinstvenih svojstva IBM RFT-a je „*storyboard testing*“ koji pomaže pri vizualizaciji i uređivanju testova koristeći prirodni jezik i snimke zaslona aplikacija. RFT se također može integrirati s ostalim IBM-ovim alatima za upravljanje životnim ciklusom aplikacija kao što su IBM Rational Team Concert i Rational Quality Manager.

2.5.4. Alat eggPlant

TestPlant je razvio eggPlant [14] za izvođenje različitih tipova testova. On se smatra jednim od najboljih alata za ispitivanje aplikacija i GUI-a. Dok većina alata za automatizaciju koriste objektno orijentirani pristup, eggPlant koristi pristup koji se temelji na slikama (engl. *image-based approach*). Alat omogućuje testerima da komuniciraju s aplikacijom na isti način kao i krajnji korisnici.

Pomoću eggPlant-a možemo koristiti jednu skriptu za provođenje ispitivanja na različitim platformama kao što su Windows, Linux, Mac itd. On nam nudi paket alata za automatizirano ispitivanje za obavljanje različitih tipova ispitivanja.

2.5.5. Alat Ranorex

Ranorex [15] pruža sveobuhvatan i profesionalni skup značajki za ispitivanje weba, mobilnih uređaja, stolnih uređaja i API-a. On ima napredne mogućnosti za identifikaciju, uređivanje i upravljanje elementima korisničkog sučelja. Ranorex olakšava automatizirano ispitivanje za testere pomoću svojeg jednostavnog i intuitivnog GUI-a, mogućnošću snimanja i pokretanja testova te generiranjem skripta. Testeri mogu integrirati Ranorex sa Selenium Grid-om kako bi omogućili raspodijeljeno ispitivanje zajedno s paralelnim ispitivanjem.

2.5.6. Alat Apache JMeter

Apache JMeter [16] je alat otvorenog koda dizajniran za ispitivanje opterećenja i mjerenje performansi (po čemu je poznat). Međutim, alat se sada koristi i za ispitivanje API-a (osobito za performanse API-a) i usluga. Neke od pozitivnih značajki alata su: jednostavno korisničko sučelje, rezultati ispitivanja se mogu ponoviti, podržava integraciju s CI alatima kao što su Jenkins.

2.6. Utjecaj ispitivanja na razvoj web aplikacija

U početku se može činiti poprilično skupo trošiti resurse i vrijeme na ispitivanje koje se moglo upotrijebiti za razvoj web aplikacije. No, lako je preračunati troškove i zaključiti da ispitivanje nije potrebno, ali je teško predvidjeti količinu posljedica te odluke, poput povećavanje troškova za provedbu izmjena i, u ekstremnim slučajevima, stvaranje gubitaka u proračunu tvrtke zbog pogoršanja imidža marke. Posjedovanjem QA (engl. *quality assurrance* – osiguranje kvalitete) odjela u timu može pomoći pri uklanjanju grešaka i nedostataka što se može očitovati u kasnijim uštedama. Zbog toga se u proračunu projekta uvijek traži mjesto za testere. No, uloga testera nije samo pronaći greške, već i kontrolirati kvalitetu. QA tim je prvi kupac i korisnik web aplikacije. Oni su prvi koji pregledavaju aplikaciju, prijavljuju se na nju, kupuju s nje itd. Osim toga oni su i prvi koji otkrivaju je li stranica prilagođena potrošačima i trebaju li se programeri više usredotočiti na to. Vrijeme posvećeno ispitivanju naizgled odgađa objavljivanje gotovog proizvoda, ali u stvari posao dodijeljen QA timu uključen je u raspored od samog početka. Bolje je utrošiti nešto više vremena na ispitivanje aplikacije i tek ju onda objaviti, nego ju objaviti punu grešaka zbog kojih će se ona morati privremeno onemogućiti. Vrijedno je zapamtiti da zbog izmjena i dopuna u ranim fazama

razvoja proces rada može biti puno brži, lakši i jeftiniji od prepisivanja cijelog koda u kasnijim fazama. Također je bitno ispitivanje na različitim platformama osim računala, poput mobitela i tableta. Potrebno je zadovoljiti krajnjeg korisnika koji želi imati brzu uslugu kojoj može lako pristupiti preko bilo kojeg uređaja. Osim same funkcionalnosti aplikacije potrebno je i brinuti se o sigurnosti, osobito kod aplikacija koje se temelje na bazama podataka o korisnicima i kupcima. Kako bi se poboljšala sigurnost QA tim provjerava kod i pronalazi greške koje su promakle programerima. QA tim je potreban svakoj profesionalnoj interaktivnoj agenciji i ne isplati se ukinuti njihovo sudjelovanje u izradi web aplikacije. Ovaj tim je prvi primatelj stvorene usluge koji ne samo da može pronaći greške, već i značajno poboljšati funkcionalnost projekta. Broj izmjena i dopuna se može značajno smanjiti zahvaljujući njihovom radu što također može uvelike utjecati na uštedu u proračunu.

3. MODEL I PROGRAMSKO RJEŠENJE WEB APLIKACIJE ZA PROCJENU RIZIKA OBOLIJEVANJA OD OSTEOPOROZE

3.1. Osteoporoza

Osteoporoza je bolest koja uzrokuje da kosti postanu slabe i krhke do razine da pad ili čak blaga naprezanja poput savijanja ili kašljanja mogu uzrokovati lom. Prijelomi vezani uz osteoporozu se najčešće javljaju u kuku, zglobu ili kralježnici. Osteoporoza nastaje kada je stvaranje nove kosti sporije od gubitka stare kosti.

Osteoporoza obično nema simptome u ranoj fazi gubitka kostiju, ali nakon što osteoporoza dovoljno oslabi kosti mogući simptomi su sljedeći: bol u leđima, gubitak visine s vremenom, nagnuto držanje ili ako se kost slomi puno lakše nego očekivano.

Kosti su u stalnom stanju obnavljanja. Izrađuje se nova kost i razgrađuje stara kost. Kada su ljudi mladi njihovo tijelo izgrađuje novu kost brže nego što razgrađuje staru kost te se time povećava koštana masa. Nakon ranih 20-tih taj se proces usporava, a većina ljudi dostiže svoj vrhunac koštane mase do 30. godine života. Kako ljudi stare, koštana mase se počinje gubiti brže nego što se stvara. Vjerojatnost razvijanja osteoporoze dijelom ovisi o tome koliko je koštane mase osoba postigla. Što je veća maksimalna koštana masa, manja je vjerojatnost da će se razviti osteoporozu tijekom starenja.

Brojni čimbenici mogu povećati vjerojatnost razvijanja osteoporoze. Postoje nepromjenjivi rizici i promjenjivi rizici.

Nepromjenjivi rizici su:

- Spol – Žene imaju veću vjerojatnost da će razviti osteoporozu nego muškarci.
- Dob – Što je osoba starija, to je veći rizik od osteoporoze.
- Rasa – Najveći rizik od osteoporoze imaju bijele osobe ili osobe azijskog podrijetla.
- Obiteljska povijest – Imati roditelja, brata ili sestru s osteoporozom povećava rizik od osteoporoze.
- Veličina okvira tijela – Muškarci i žene koje imaju male okvire tijela imaju veći rizik jer mogu imati manju koštanu masu.

Promjenjivi rizici su:

- Steroidi – Korištenje steroida smanjuje količinu apsorbiranog kalcija iz želudca i povećava potrošnju kalcija putem bubrega čime utječe na izgradnju kostiju.
- Estrogen – Nedostatak estrogena ubrzava proces gubljenja koštane mase.
- Vježbanje – Vježbanje potiče razvoj kostiju, a nedostatak vježbanja znači da će osoba biti više izložena gubitku kalcija iz kostiju.
- Dijeta – Ako osoba unutar svoje dijete ne uključuje dovoljnu količinu vitamina D i kalcija, ili ako je osoba pothranjena, imati će veći rizik od osteoporoze.
- Pušenje – Duhan je izravno toksičan za kosti. Kod žena smanjuje razinu estrogena a kod muškaraca smanjuje aktivnost testosterona.
- Alkohol – Pijenje puno alkohola smanjuje sposobnost tijela da stvara kosti.

3.2. Postojeća programska rješenja

Već postoje razna rješenja za procjenu rizika obolijevanja od osteoporoze, ali mora se uzeti u obzir da te procjene nisu 100% točne i ne bi se trebalo oslanjati isključivo na njih kao dijagnozu. Uvijek se preporuča odlazak na liječnički pregled kako bi se pravilno dijagnoziralo osobu.

3.2.1. Medscape osteoporosis risk SCORE

Medscape osteoporosis risk SCORE [18] (engl. *SImple Calculated Osteoporosis Risk Estimation*) je, kao što samo ime govori, web aplikacija za jednostavnu procjenu rizika obolijevanja od osteoporoze. Koristi se samo šest ulaznih parametara koji se unose u izraz izveden iz [19] i [20] te kao izlaz se dobiva brojčana vrijednost koja se uspoređuje sa zadanom tablicom kako bi odredili rizik. Način procjene rizika u ovom završnom radu se zasniva na temelju ove aplikacije.

3.2.2. QFRacture

QFracture [21] se koristi za izračun rizika pojave prijeloma uzrokovanog od osteoporoze. Ova web aplikacija, za razliku od prijašnje, traži unos većeg broja parametara i kao izlaz se dobiva postotak. Taj postotak predstavlja šanse da će se pojaviti prijelom uzrokovan od osteoporoze. Algoritam su razvile Julia Hippisley-Cox i Carol Coupland, a temeljji se na rutinski prikupljenim podacima od više tisuća liječnika opće prakse koji su dobrovoljno doprinijeli podacima u bazi podataka QResearch-a. QFracture je razvijen za populaciju Ujedinjenog Kraljevstva i namijenjen je za upotrebu u Ujedinjenom Kraljevstvu.

3.2.3. FRAX

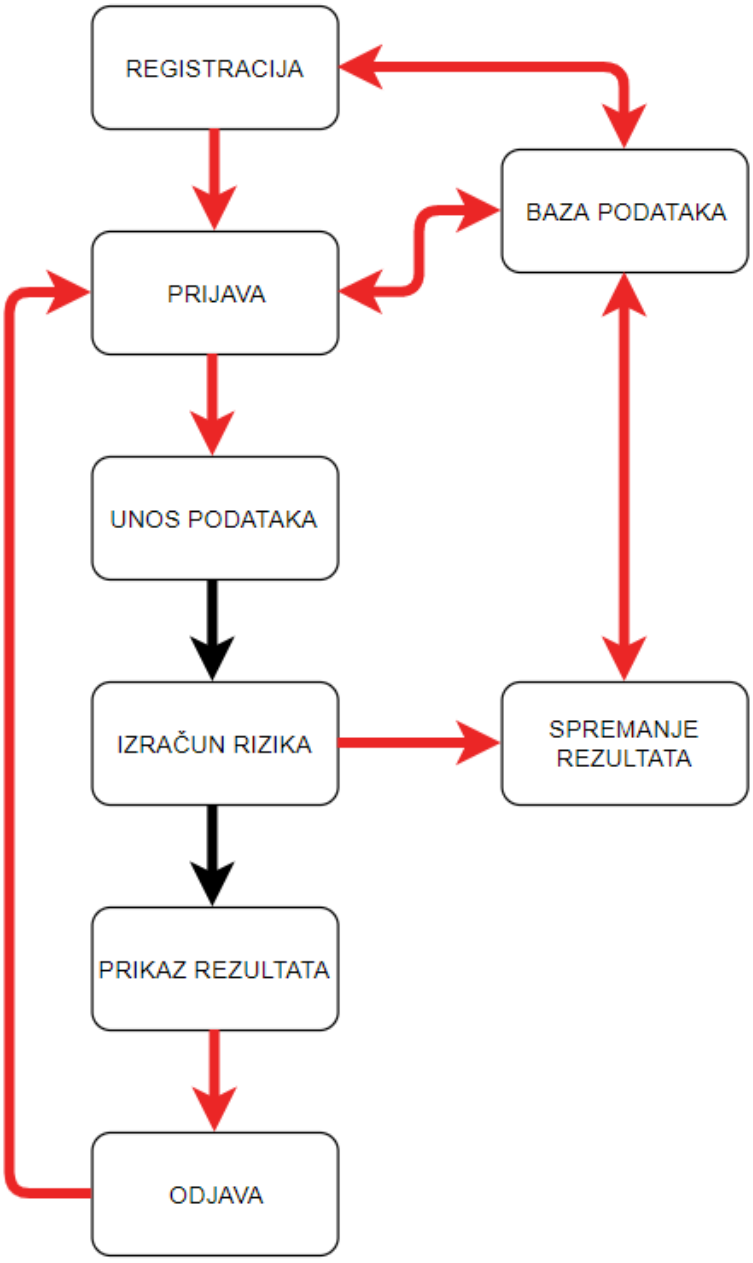
Alat FRAX [22] (engl. *Fracture Risk Assessment Tooll*) je razvijen za procjenu rizika od prijeloma. Temelji se na pojedinačnim modelima pacijenata koji integriraju rizike povezane s kliničkim čimbenicima rizika, kao i koštanom mineralnom gustoćom (BMD – engl. *bone mineral density*) na vratu bedrene kosti. FRAX modeli su razvijeni iz proučavanja populacijskih skupina iz Europe, Sjeverne Amerike, Azije i Australije. U svojem najsofisticiranijem obliku alat FRAX se pokreće na računalu. Dostupno je i nekoliko inačica za papir, koje se temelje na broju čimbenika rizika, koje se mogu preuzeti za upotrebu u uredu.

3.2.4. IOF risk check

Za razliku od prije navedenih alata za procjenu rizika obolijevanja od osteoporoze, IOF risk check [23] kao rezultat ne daje brojčanu vrijednost. Nakon što se odgovori na nekoliko jednostavnih pitanja dobiva se pismeni rezultat gdje se obavještava korisnika koliki mu je rizik. Osim rezultata također se predloži odlazak liječniku i neke preporuke kako si smanjiti rizik u budućnosti.

3.3. Model web aplikacije

Jedan od ciljeva ovoga rada je izrada web aplikacije za procjenu rizika obolijevanja od osteoporoze. Kao što je i prije navedeno u radu ova web aplikacija se zasniva na već postojećem rješenju Medscape osteoporosis risk SCORE [7]. Kako bi se procijenio rizik koristi se šest čestih čimbenika koji utječu na rizik obolijevanja od osteoporoze (dob, rasa, težina, broj prijeloma koji nisu uzrokovani traumom, prisutnost reumatoidnog artritisa i prijašnje korištenje estrogena). Svaki od tih čimbenika se zatim boduje na određeni način te se unose u izraz za izračun rizika. Način bodovanja i izraz se može pronaći u [7]. Na slici 3.1 prikazan je model web aplikacije. Na modelu je crnim strelicama označen dio koji je uvijek dostupan korisniku, a crvenim strelicama dodatni dio koji je dostupan samo ako je korisnik registriran i prijavljen. Od korisnika se traži unos svih traženih podataka koji će se zatim koristit za izračun i prikaz rezultata. Korisnik nije prisiljen na registraciju ili prijavu na stranicu kako bi mogao pristupio testu za procjenu rizika, ali ako je korisnik registriran i prijavljen na web stranicu biti će mu ponuđena dodatna opcija da spremi svoje rezultate.



Slika 3.1. *Model web aplikacije za procjenu rizika obolijevanja od osteoporoze*

3.4. Alati i jezici korišteni za izradu web aplikacije

3.4.1. HTML

HTML (engl. *HyperText Markup* Language) [24] je računalni jezik osmišljen kako bi omogućio izradu web stranica. HTML opisuje strukturu web stranice, određuje naslove, odlomke, obrasce za popunjavanje i druge male aplikacije koje se pojavljuju na stranici. On opisuje samo strukturu

elemenata, a ne i njihov izgled. Web preglednici primaju HTML dokumente s web poslužitelja ili iz lokalne pohrane te ih pretvaraju u multimedijske web stranice. Kod rada s HTML-om koriste se jednostavne strukture koda (oznake i atributi) za označavanje web stranice.

3.4.2. CSS

CSS (engl. *Cascading Style Sheets*) [25] je deklarativni opisni jezik koji pomoću selektora određuje element po nazivu, atributu, *id*-u ili na neki drugi način te na njega primjenjuje određene stilove (boja teksta, tip fonta, veličina fonta, pozicija elementa na web stranici itd.). Kao i HTML, CSS se piše jednostavnim tekstom pomoću uređivača teksta ili programa za obradu teksta. Postoje tri glavna načina dodavanja CSS koda na HTML stranice. CSS kod može biti vanjski, unutarnji ili unutar linije. Vanjski stilovi se spremaju kao .css datoteke i mogu se koristiti za određivanje izgleda cijele web stranice kroz jednu datoteku. Za unutarnji stil, CSS upute su napisane direktno u zaglavlje određene .html datoteke. Stilovi unutar linije su isječci CSS koda zapisani izravno u HTML kod nekog elementa.

3.4.3. JavaScript

JavaScript [26] je skriptni jezik koji se primarno koristi na webu. Koristi se za poboljšanje HTML stranica i obično se nalazi u HTML kodu. JavaScript je interpretirani jezik te ga zbog toga nije potrebno prevoditi (engl. *compile*). On prikazuje web stranice na interaktivan i dinamičan način. To omogućava stranicama da reagiranju na događaje, prihvaćaju promjenjiv tekst, provjeravaju podatke, izrađuju kolačiće, prikazuju posebne efekte itd. Postoje dva načina korištenja JavaScripta-a u .html datoteci. Prva uključuje umetanje čitavog JavaScript koda u HTML kod, dok druga metoda koristi zasebnu JavaScript datoteku s proširenjem .js. Iako se JavaScript uglavnom koristi za interakciju s HTML objektima, može se koristiti i za interakciju s drugim objektima koji nisu HTML kao što su proširenja web preglednika, svojstva CSS-a, trenutni datum ili sam preglednik. Za pisanje koda, kao za HTML i CSS, potreban je samo osnovni uređivač teksta.

3.4.4. MySQL

MySQL [27] je potpuno opremljeni sustav za upravljanje relacijskim bazama podataka. MySQL sponzorira švedska tvrtka MySQL AB koja je u vlasništvu Oracle Corp. Međutim, izvorni kod MySQL-a je slobodno dostupan jer je prvobitno razvijen kao besplatna programska podrška (engl. *freeware)*. MySQL je napisan na C i C++ i kompatibilan je sa svim glavnim operativnim sustavima.

MySQL je vrlo popularan za web hosting programe zbog mnoštva web optimiziranih značajki poput HTML podataka i zbog toga što je potpuno besplatan. On je dio Linux, Apache, MySQL, PHP arhitekture (kombinacija platformi koja se često koristi za isporuku i podršku naprednih web aplikacija).

3.4.5. PHP

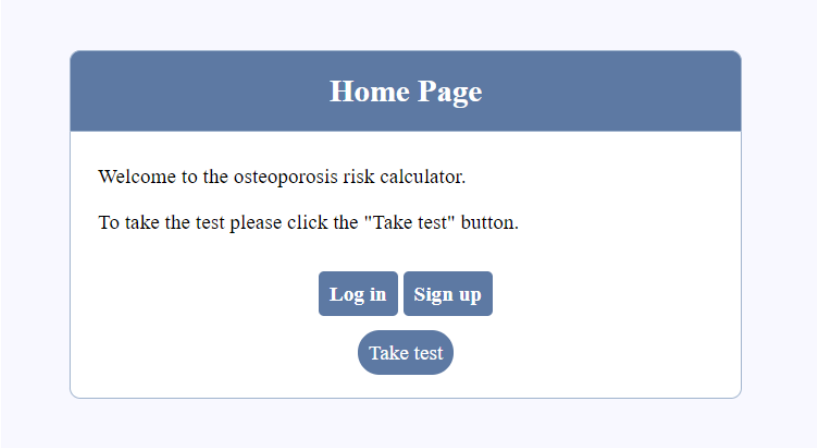
PHP [28] je rekurzivna kratica za PHP: Hypertext Preprocessor, skriptni jezik koji se koristi za stvaranje dinamičkih i interaktivnih HTML web stranica. Poslužitelj obrađuje PHP naredbe kada posjetitelj web stranice otvori stranicu, a zatim rezultate pošalje pregledniku posjetitelja. PHP je vrlo jednostavan za početnike, a nudi i mnoge napredne funkcije za profesionalne programere. On radi najučinkovitije na Apache poslužitelju, ali može se izvoditi i na IIS-u. PHP je jezik otvorenog koda i može se graditi kao Apache modul ili CGI skripta.

3.5. Programska implementacija web aplikacije

Pri izradi web aplikacije za procjenu rizika obolijevanja od osteoporoze potrebno je ostvariti određene funkcionalnosti, koje su: izračun rizika obolijevanja od osteoporoze, registracija korisnika, prijava korisnika te prikaz i spremanje rezultata.

3.5.1. Izgled web aplikacije

Za dizajniranje izgleda web stranice korišteni su HTML i CSS za statičke elemente, a JavaScript i PHP za dinamičke. Pomoću njih se izradila i stilizirala struktura web stranice. Na slici 3.2 je prikazan izgled početne stranice.



Slika 3.2. *Početna stranica web aplikacije*

Na početnoj stranici može se uočiti zaglavlje gdje piše naslov stranice i ispod njega glavni sadržaj u kojemu se nalazi običan tekst i tri gumba. Klikom na gumb korisnik će biti odveden na drugu stranicu. Dio HTML koda korišten za izradu početne stranice prikazan je kodom 3.1. Unutar *head* oznaka se nalazi dio koda kojim se povezuje CSS datoteka s HTML-om. Dio koda CSS datoteke za početnu stranicu prikazan je kodom 3.2. U njemu se vidi kako se poziva element klase *content* te se na njega primjenjuju razni stilovi (širina, rubovi, boja pozadine itd.). Također je prikazan način kako stilizirati sve odjeljke unutar *content*-a te sve poveznice unutar tih odjeljaka. Nastavljajući dalje kroz HTML kod nailazi se na PHP dio koda. Taj kod provjerava je li korisnik prijavljen u sustav ili ne. Ako nije onda će se prikazati HTML kod koji se nalazi između PHP koda (linkovi za registraciju i prijavu). U ovome dijelu koda nedostaje još jedan dio PHP koda koji se koristi za prikazivanje elemenata ako korisnik je prijavljen u sustav (korisničko ime, gumb za odjavu, rizik korisnika, poruka o uspješnoj prijavi). Potpuni kod za početnu stranicu i sve ostale stranice web aplikacije nalazi se u prilogu 2.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home</title>
    <link rel="stylesheet" type="text/css"
href="styles/home_style.css?v=<?php echo time(); ?>">
  </head>
  <body>
    <div class="header">
        <h2>Home Page</h2>
    </div>
    <div class="content">

        <p>Welcome to the osteoporosis risk calculator.</p>
        <p>To take the test please click the "Take test" button.</p>
        <br>

        <?php if (!isset($_SESSION['username'])) : ?>
          <table>
            <tr>
              <th>
                <a href="user_authentication/login.php">Log in</a>
                <a href="user_authentication/register.php">Sign up</a>
              </th>
            </tr>
          </table>

        <?php endif ?>

        <p><a href="test_risk/risk_calculator.php">Take test</a></p>
    </div>
  </body>
</html>
```

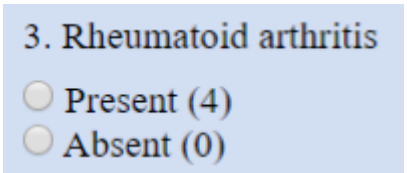
Programski kod 3.1. *Dio HTML koda početne stranice web aplikacije*

```
.content {
  width: 30%;
  min-width: 250px;
  margin: 0px auto;
  padding: 20px;
  border: 1px solid #a3b6cf;
  background: #ffffff;
  border-radius: 0px 0px 10px 10px;
}
.content > p {
  padding: 10px 5px 10px 5px;
  text-align: justify;
}
.content > p > a {
  color: #ffffff;
  padding: 10px;
  background: #5d79a3;
  border-radius: 50px;
  transition: background 0.2s;
}
```

Programski kod 3.2. *Dio CSS koda početne stranice web aplikacije*

3.5.2. Izračun rizika i prikaz rezultata

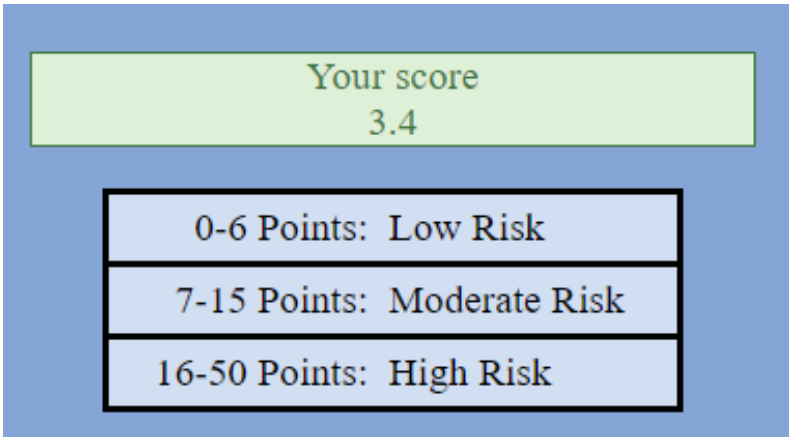
Za izračun rizika i prikaz rezultata koristio se JavaScript. Dio JavaScript koda za izračun rizika je prikazan kodom 3.3. U tome kodu se nalaze dvije funkcije: *check* i *validate*. Funkcija *validate* provjerava jesu li svi podatci ispravno uneseni. Ako jesu onda se izvršava funkcija *check* koja računa i prikazuje rizik, a ako podatci nisu pravilno uneseni onda se ispisuje poruka da podatci nisu pravilno uneseni. Rizik se računa pomoću izraza koji je zapisan kao komentar unutar koda 3.3. Za izračun rizika, vrijednosti dobi i težine se dobivaju unosom u tekstualno polje, a vrijednosti ostalih čimbenika rizika se dobivaju ovisno o tome koji odgovor je odabran (pokraj odgovora se nalazi vrijednost unutar zagrada, slika 3.3). Na slici 3.4 prikazan je izgled rezultata, a na slici 3.5 prikazana je poruka o pogrešnome unosu. Rezultat i poruka o pogrešno unesenim podatcima se prikazuju na način da JavaScript locira specifični element te ga učini vidljivim i po potrebi promjeni tekst koji se nalazi unutar tog elementa.



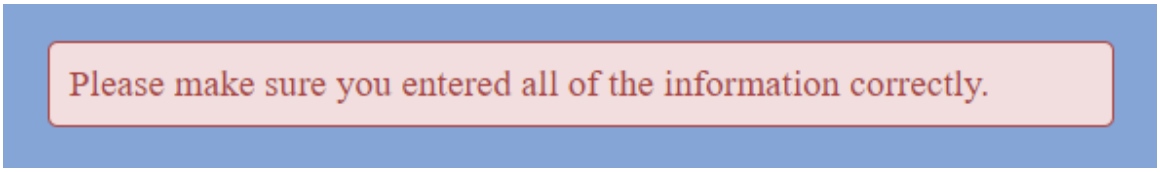
Slika 3.3. *Primjer prikaza vrijednosti ostalih čimbenika rizika*

```
function check(){
    if(!valid){
        return;
    }
    var question1 = document.quiz.question1.value;
    var question2 = document.quiz.question2.value;
    //...
    var score = 0;

//score = (3*age/10) + race + rheumArth + estr + fracHist - (weight(lbs)/10)

    score += (3*question1/10);

    if(question2 == "nonBlack"){
        score += 5;
    }
    //...
    document.getElementById("after_submit").style.visibility = "visible";
    document.getElementById("number_score").innerHTML = score.toFixed(1);
    document.quiz.hidden_score.value = score.toFixed(1);
}

function validate() {
    document.getElementById("after_submit").style.visibility = "hidden";
    document.getElementById("after_error").style.visibility = "hidden";

    if( document.quiz.question1.value == "" ) {
        document.getElementById("after_error").style.visibility = "visible";
        document.getElementById("error_message").innerHTML = errorMsg;
        valid = false;
        return;
    }
    //...
    valid = true;
    return;
}
var valid = true;
var errorMsg = "Please make sure you entered all of the information
correctly.";
```

Programski kod 3.3. *Dio JavaScript koda za izračun rizika*



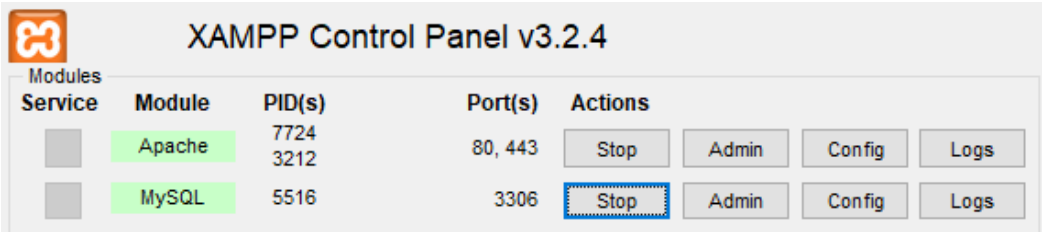
Slika 3.4. *Prikaz rezultata izračuna rizika*



Slika 3.5. *Prikaz poruke o pogrešnom unosu podataka*

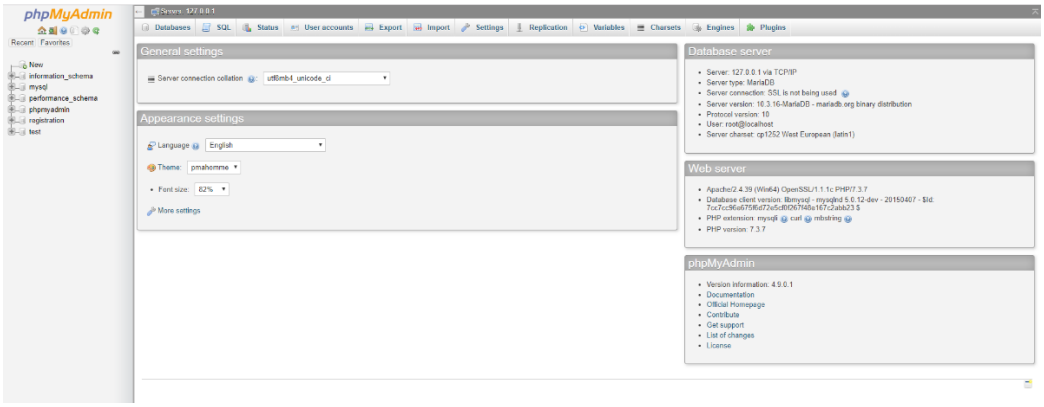
3.5.3 Registracija i prijava korisnika

Za registraciju i prijavu korisnika potrebna je baza podataka koja je kreirana pomoći MySQL-a. Kako bi se mogla napraviti baza podataka potrebno je prvo pokrenuti MySQL i Apache unutar XAMPP sučelja koje je prikazano na slici 3.6.



Slika 3.6. *XAMPP sučelje*

Nakon toga potrebno je otvoriti internetski preglednik i upisati u pretragu „localhost/phpmyadmin/“ kako bi se otvorila phpMyAdmin stranica pomoću koje će se napraviti baza podataka. Izgled web stranice phpMyAdmin je prikazana na slici 3.7. Na lijevoj strani se nalaze već postojeće baze podataka kao i opcija za kreiranje nove. Unutar novostvorene baze podataka potrebno je napraviti tablicu korisnika koja je prikazana na slici 3.8. Unutar tablice korisnika spremati će se njihovo korisničko ime, e-pošta, šifra i rizik.

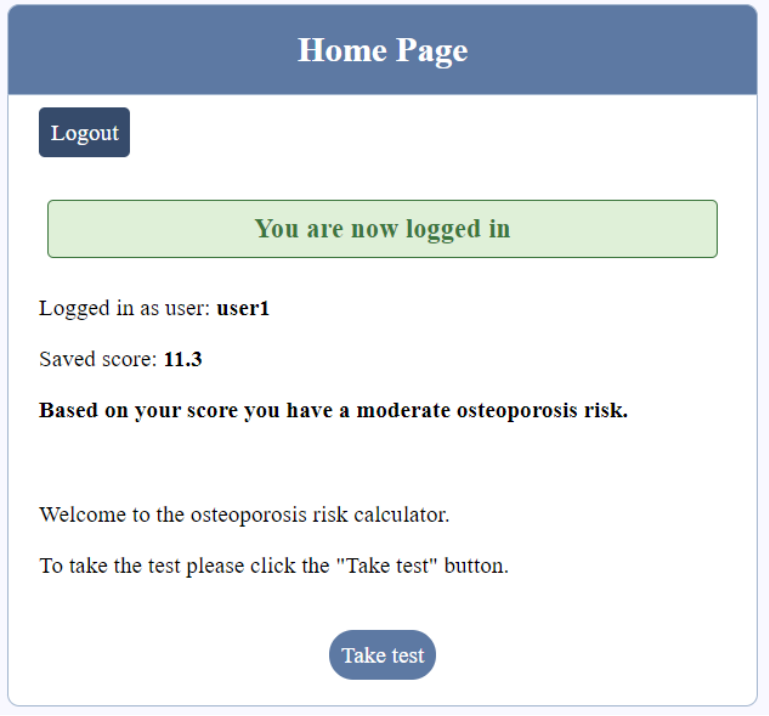


Slika 3.7. *Web stranica phpMyAdmin*

	#	Name	Type	Collation	Attributes	Null	Default
☐	1	id 🔑	int(11)			No	*None*
☐	2	username	varchar(255)	utf8mb4_croatian_ci		No	*None*
☐	3	email	varchar(255)	utf8mb4_croatian_ci		No	*None*
☐	4	password	varchar(255)	utf8mb4_croatian_ci		No	*None*
☐	5	score	varchar(255)	utf8mb4_croatian_ci		No	None

Slika 3.8. *Tablica korisnika*

Nakon što je stvorena baza podataka koristiti će se PHP za pristup i rad s njom. Kod za prijavu korisnika je prikazan programskim kodom 3.4. Unutar koda se prvo pokreće nova ili nastavlja trenutna sesija. Zatim se inicijaliziraju varijable te se spaja na bazu podataka. Postupak prijave će se pokrenuti tek kada se klikne na gumb za prijavu korisnika. Prvo se dohvaćaju vrijednosti uneseni unutar formi te se provjerava je li sve pravilno uneseno. Ako su svi uvjeti ispunjeni onda će se korisnik uspješno prijaviti, inače će se ispisati poruka o nastalim greškama. Nakon što se korisnik prijavi biti će usmjeren na početnu stranicu gdje će moći vidjeti informacije o svojem profilu (prikazano na slici 3.9). Registracija je ostvarena na sličan način kao prijava. Također je bitno napomenuti da prijavljeni korisnik može spremati novo izračunati rizik.



Slika 3.9. *Uspješna prijava korisnika*

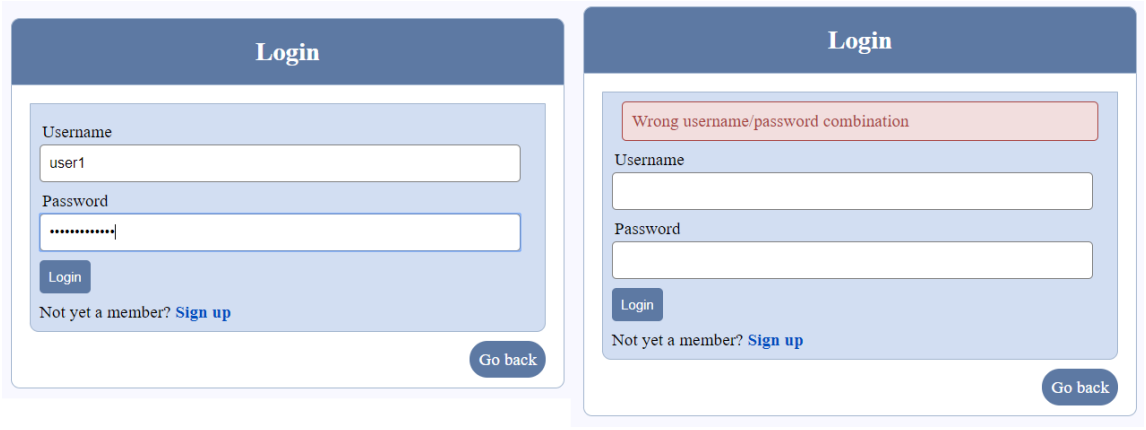
```php
<?php
session_start();

$username = "";
$email    = "";
$errors = array();
$score = "";

$db = mysqli_connect('localhost', 'root', '', 'registration');

if (isset($_POST['login_user'])) {
  $username = mysqli_real_escape_string($db, $_POST['username']);
  $password = mysqli_real_escape_string($db, $_POST['password']);

  if (empty($username)) {
    array_push($errors, "Username is required");
  }
  if (empty($password)) {
    array_push($errors, "Password is required");
  }

  if (count($errors) == 0) {
    $password = md5($password);
    $query = "SELECT * FROM users WHERE username='$username' AND
password='$password'";
    $results = mysqli_query($db, $query);

    $query2 = "SELECT score FROM users WHERE username='$username'";
    $tempScore = mysqli_query($db, $query2);
    $row = mysqli_fetch_array($tempScore);;

    if (mysqli_num_rows($results) == 1) {
      $_SESSION['username'] = $username;
      $_SESSION['success'] = "You are now logged in";
      $_SESSION['score'] = $row[0];
      header('location: ../index.php');
    }else {
      array_push($errors, "Wrong username/password combination");
    }
  }
}
?>
```

Programski kod 3.4. *PHP kod za prijavu korisnika*

3.5.4. Primjeri korištenja web aplikacije

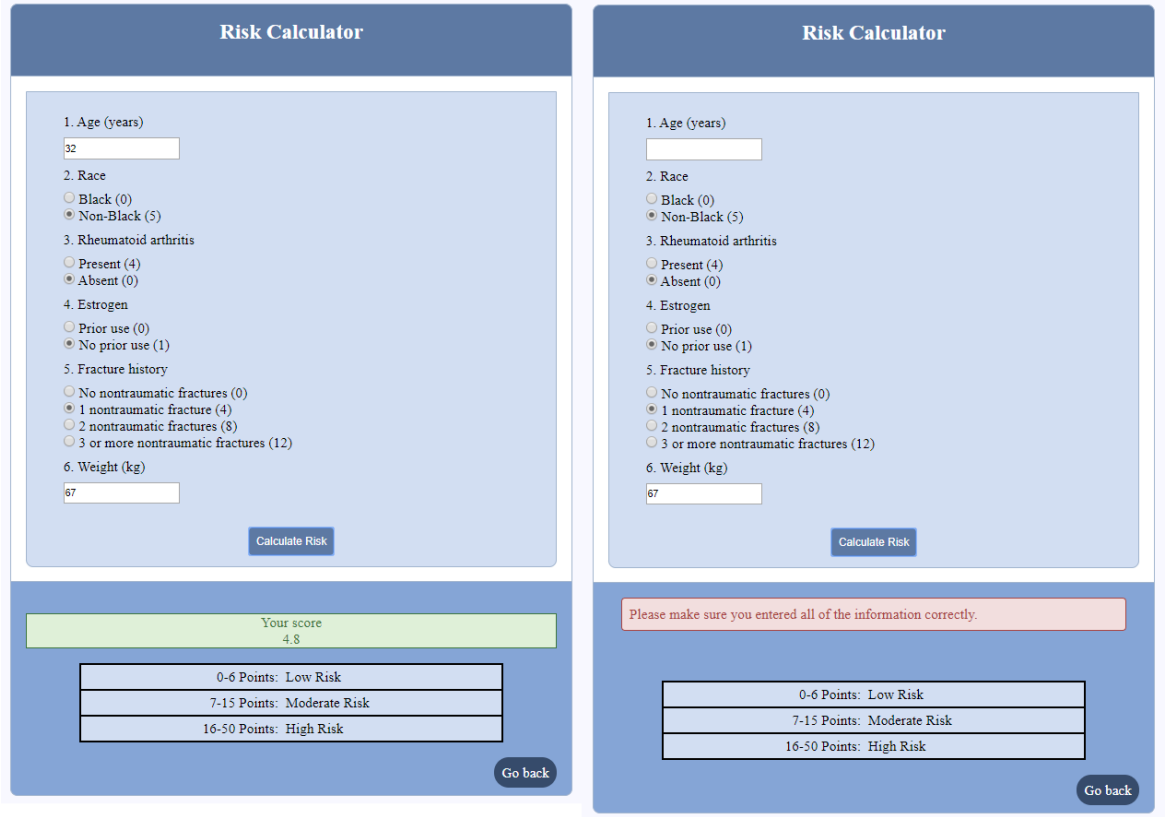
Na slici 3.10 je prikazan primjer krivo unesene šifre pri prijavi korisnika. U srednjem dijelu koda 3.4 se vidi razlog pojave greške. Šalje se zahtjev bazi podataka da dohvati sve korisnike čije se korisničko ime i šifra poklapaju s unesenim korisničkim imenom i šifrom. Zatim se provjerava koliko je korisnika pronađeno. Ako je pronađen specifično jedan korisnik onda će prijava biti

uspješna, a ako nije nađen niti jedan korisnik ili više od jednog korisnika ispisati će se poruka o nastaloj grešci.



Slika 3.10. *Unos krive šifre (lijevo) i rezultat krivo unesene šifre (desno)*

Na slici 3.11 su prikazani rezultati pravilno unesenih parametar u kalkulator za rizik i rezultati kad jedan od parametara nije unesen.



Slika 3.11. *Pravilno uneseni parametri u kalkulator rizika (lijevo) i jedan prazan parametar u kalkulatoru rizika (desno)*

Zadnji primjer je prikazan na slici 3.12. Na toj slici je prikazan rezultat za pokušaj registracije korisnika pomoću imena i e-pošte koja se već nalazi u bazi podataka. (Kao podsjetnik za prikaz rezultata uspješne prijave ili registracije korisnika može se provjeriti slika 3.9)

Slika 3.12. *Unos korisničkog imena i e-pošte koji su već registrirani (lijevo) i rezultat unosa već postojećeg korisničkog imena i e-pošte (desno)*

4. ISPITIVANJE WEB APLIKACIJE ZA PROCJENU RIZIKA OBOLIJEVANJA OD OSTEOPOROZE

4.1. Selenium

Selenium [11] je alat otvorenog koda koji omogućuje automatizirano ispitivanje web aplikacija na različitim internetskim preglednicima. Sastoji se od četiri dijela, a to su Selenium Integrated Development Enviorment (IDE), Selenium Remote Control (RC), Selenium WebDriver i Selenium Grid.

Selenium je originalno razvio Jason Huggins 2004. godine dok je ispitivao unutarnju aplikaciju ThoughtWorks-a. Shvatio je kako može puno bolje iskoristiti svoje vrijeme umjesto da ručno ispituje aplikaciju kada god napravi neku promjenu. Razvio je JavaScript biblioteku koja je mogla pokretati interakcije s aplikacijom što mu je omogućilo automatsko ponavljanje testova na više internetskih preglednika. Ta biblioteka je na kraju postala Selenium Core. Kako bi se riješio problem pravila istog porijekla, koje sprječava JavaScript kod da pristupi elementima drugačije domene od one na kojoj je kod pokrenut, razvio se Selenium RC, tj. Selenium 1. Njega je razvio inžinjer ThoughtWorks-a Paul Hammant tako što je napravio server koji služi kao HTTP posrednički poslužitelj (engl. *proxy*) kako bi „prevario“ preglednik da vjeruje kako su Selenium Core i web aplikacija koja se ispituje na istoj domeni. Kako su web aplikacije postajale sve snažnije i restriktivnije s JavaScriptom Simon Stewart je 2006. godine odlučio napraviti WebDriver koji je bio prvi višeplatformski *framework* za ispitivanje koji je mogao upravljati preglednikom na razini operacijskog sustava. Iste godine japanac Shinya Kasatani je razvio Firefox proširenje (engl. *extension*) po imenu Selenium IDE koji može automatizirati preglednik pomoću svojstva snimanja i reproduciranja (engl. *record-and-playback*). Idući napredak bio je 2008. godine kada je Philippe Hanrigou je napravio Selenium Grid koji je omogućio pokretanje više Selenium testova odjednom na više lokalnih i/ili udaljenih sistema čime je bitno smanjeno vrijeme izvođenja testova. Također 2008. godine dogodio se najbitniji događaj u razvoju Seleniuma, spajanje Selenium RC-a i WebDrivera kako bi nastao Selenium WebDriver, tj. Selenium 2.

Selenium RC je dugo vremena bio glavni Selenium-ov projekt zato što je on prvi alat za automatizirano ispitivanje web aplikacija koje je omogućilo korisnicima da koriste programski jezik

koji žele. Kao što je prijašnje navedeno u odlomku o povijesti Selenium-a on koristi server koji služi kao HTTP posrednički poslužitelj kako bi riješio problem pravila istog porijekla. Komponente Selenium RC su:

1. Selenium Server koji pokreće i gasi internetske preglednike, interpretira i pokreće Selenese naredbe (skup Selenium komandi) i služi kao HTTP posrednički poslužitelj koji presreće i potvrđuje HTTP poruke koje se izmjenjuju između preglednika i AUT-a (engl. *application under test*).
2. Klijentske biblioteke koje pružaju sučelje između svakog programskog jezika i Selenium RC Server-a.

Selenium IDE je alat korišten za snimanje, uređivanje, debugiranje (engl. *debugging*) i reproduciranje funkcionalnih testova. On je originalno implementiran kao Firefox proširenje, ali danas je također implementiran kao i Chrome proširenje. Pomoću Selenium IDE moguće je snimati i izvoziti (engl. *export*) testove u bilo kojem od podržanih programskih jezika kao npr. Java, Ruby, JavaScript, PHP itd. Koristan je za učenje koncepata automatiziranog ispitivanja i Selenium-a kao što su: Selenese, pokretanje prilagođenih JavaScript kodova pomoću *runScript*, izvoženje testnih slučajeva u različitim formatima itd. Također je idealan za početnike zato što za stvaranje testova je potrebno vrlo malo predznanja iz programiranja.

Selenium Grid je zasnovan na *hub-node* arhitekturi u kojoj je *hub* samo jedan uređaj na kojem pokrećemo test, ali će se test izvršavati na različitim uređajima koji se zovu *nodes*. On se koristi kada se žele pokretati testovi na različitim internetskim preglednicima, operacijskim sustavima i uređajima sve u isto vrijeme čime se smanjuje ukupno vrijeme potrebno za izvršavanje svih testova. Prije se Selenium Grid koristio zajedno sa Selenium RC kako bi se pokretalo više testova na udaljenim uređajima, ali danas ljudi smatraju da je Selenium WebDriver bolji od RC te iz tog razloga Grid može raditi s WebDriver-om kao i s RC-om.

Slenium WebDriver je poboljšana inačica Selenium RC koji je nastao spajanjem Selenium RC-a i WebDriver-a. On podržava WebDriver API (engl. *application programming interface*) i njegove temeljne tehnologije zajedno sa Selenium RC tehnologijom ispod WebDriver API-a za maksimalnu fleksibilnost pri prebacivanju (engl. *porting*) svojih testova. Za razliku od Selenium RC Selenium WebDriver se ne oslanja na JavaScript za automatizaciju nego upravlja preglednikom tako da

komunicira direktno s njim. Svaki preglednik ima svoj *driver* za pokretanje testova. Selenium WebDriver također bolje podržava dinamičke web stranice gdje se elementi stranice mogu mijenjati bez ponovnog učitavanja stranice. On je trenutno najkorišteniji Selenium alat.

4.2. Plan ispitivanja

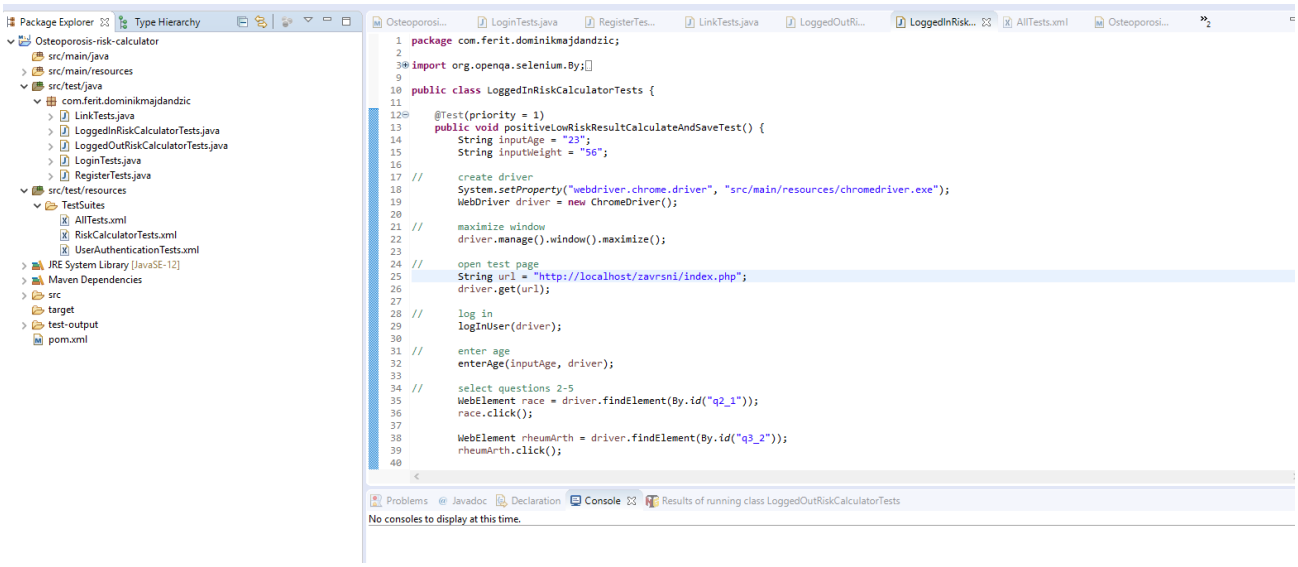
Web aplikacija za procjenu rizika obolijevanja od osteoporoze će se ispitivati pomoću Selenium Webdriver-a i Selenium IDE-a. Pomoću WebDriver-a je potrebno napisati kod za testne scenarije,a pomoću IDE-a je potrebno snimiti testne scenarije.

4.2.1. Testni scenariji

Prvo za što se trebaju napisati testovi su testovi za navigaciju kroz aplikaciju. Potrebno je osigurati da svi linkovi rade ispravno. Iduće što će se ispitivati je prijava korisnika. Tu je potrebno ispitivati slučajeve kada se korisnik uspješno prijavi i slučajeve kada korisnik ne unese pravilne podatke. Nakon prijave, potrebno je ispitivati registraciju. Za registraciju također treba ispitivati slučajeve kada se korisnik pravilno registrira, ali za to je potrebno napraviti generator nasumičnih znakova kako bi mogli test pokretati više puta. Osim toga, potrebno je ispitivati sve slučajeve kada korisnik ne unese pravilno podatke potrebne za registraciju. Također je potrebno ispitivati mogućnost odjave korisnika. Na kraju je potrebno ispitivati kalkulator rizika. Zato što on dobiva dodatnu funkcionalnost (spremanje rezultata) kada je korisnik prijavljen, potrebno ga je ispitivati kada je korisnik prijavljen i odjavljen. Tu će se izraditi razni testni scenariji sa valjanim unosima i ne valjanim unosima. Nakon što su svi testovi napisani potrebno ih je spojiti u testne pakete radi lakšeg pokretanja.

4.3. Ispitivanje pomoću Selenium WebDriver-a

Prije samog pisanja testnih scenarija potrebno je pripremiti radnu okolinu. Za pisanje testnih scenarija će se koristiti Java pa je potrebno instalirati JDK (engl. *Java Development Kit*). Također je potrebno instalirati Apache Maven [29] koji će olakšati upravljanje projektom (neće biti potrebno tražiti i skidati potrebne .jar datoteke jednu po jednu nego će to sve odraditi Maven). Kao *framework* će se koristiti Eclipse IDE [30] uz TestNG eclipse proširenje [31] za prikaz rezultata. Na slici 4.1 prikazano je sučelje Eclipse IDE-a. Na lijevoj strani se nalazi navigacija, na sredini kod, a dolje se nalaze konzola i TestNG rezultati.



Slika 4.1. *Eclipse IDE korisničko sučelje*

4.3.1. Pisanje testnih scenarija

Primjer testnog scenarija je prikazan programskim kodom 4.1. Prvo je potrebno pokrenuti web preglednik za što nam je potreban određeni driver. No, nije potrebno samo imati driver web preglednika, potrebno je i imati sam preglednik na uređaju na kojemu se ispituje. Nakon otvaranja preglednika potrebno je otvoriti stranicu koja se ispituje. Zatim se traže elementi na stranici koji su potrebni za daljnje ispitivanje. U ovom primjeru unutar metode enterUsername se dohvaća element kojemu je vrijednost name = username. Osim po imenu elementa, može se locirati pomoću *xpath*-a, *id*-a i *class*. Nakon što su dohvaćeni traženi elementi s njima se može upravljati (npr. unos teksta). Upravljanjem elemenata se imitira način na koji bi korisnik koristio web aplikaciju. Na kraju je potrebno potvrditi da je test uspješan. U ovome slučaju (uspješna prijava korisnika) potrebno je potvrditi da je korisnik preusmjeren natrag na početnu stranicu, da se na njoj vidi gumb za odjavu, da je prikazana poruka o uspješnoj prijavi i da je prikazano točno korisničko ime. Na kraju se zatvara preglednik. Svi ostali testni scenariji se nalaze u prilogu 3.

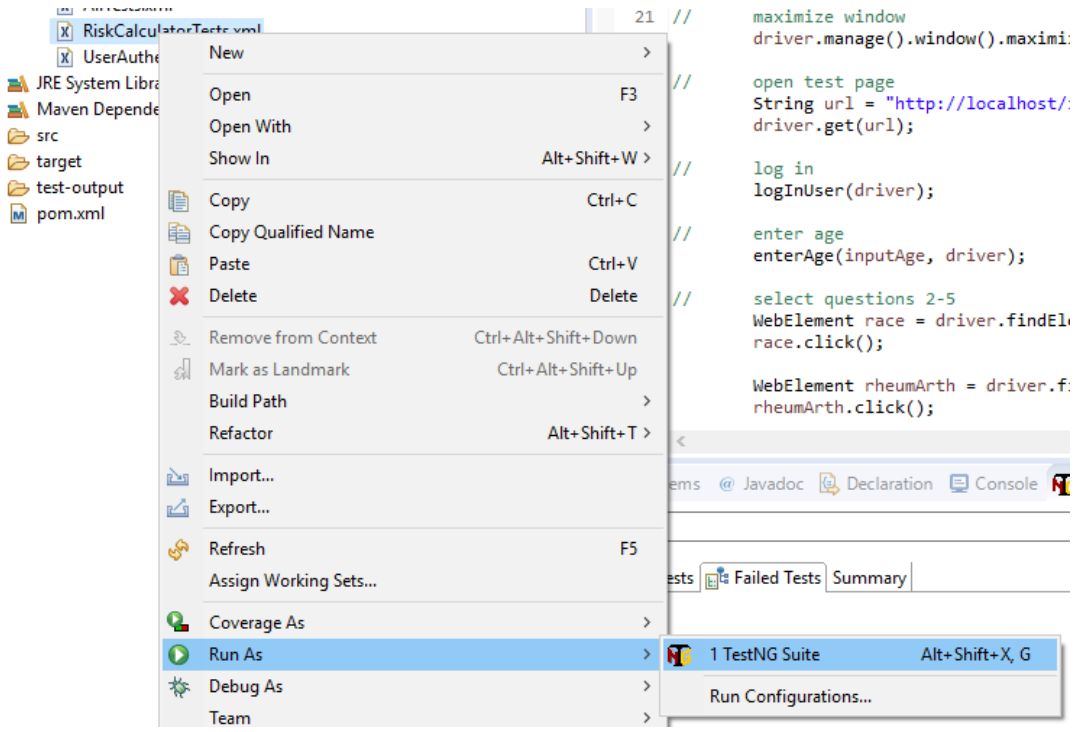
```
@Test(priority = 1)
	public void positiveLoginTest() {
		String inputUsername = "user1";
		String inputPassword = "pass123";

//		create driver
		System.setProperty("webdriver.chrome.driver",
"src/main/resources/chromedriver.exe");
		WebDriver driver = new ChromeDriver();

//		open test page
		String url = "http://localhost/zavrsni/user_authentication/login.php";
		driver.get(url);

//		enter username
		enterUsername(inputUsername, driver);

//		enter password
		enterPassword(inputPassword, driver);

//		click login button
		clickLogInButton(driver);

//		verification
//		new url
		String expectedUrl = "http://localhost/zavrsni/index.php";
		validateUrl(driver, expectedUrl);

//		logout btn visible
		validateLogOutButton(driver);

//		successful login message
		String expectedMessage = "You are now logged in";
		validateSuccessMessage(driver, expectedMessage);

//		username displayed
		validateUsernameDisplayed(inputUsername, driver);

//		close browser
		driver.quit();
	}

private void enterUsername(String inputUsername, WebDriver driver) {
		WebElement username = driver.findElement(By.name("username"));
		username.sendKeys(inputUsername);
	}
```

Programski kod 4.1. *Dio koda testnog scenarija za uspješnu prijavu korisnika*

4.3.2. Pokretanje testova i prikaz rezultata

Unutar testne klase svaki test se označava sa „@Test“ kako bi TestNG znao koje su metode testovi tako da ih pokrene. Mogu se dodati i neki dodatni uvjeti kao prioritet izvođenja (programski kod 4.1). Testovi će se pokretati i prikazivati pomoću TestNG. Testovi se pokreću desnim klikom na

testnu klasu ili testnu grupu te klikom na „run as TestNG test/suite“. Na slici 4.2 je prikazano pokretanje testne grupe.

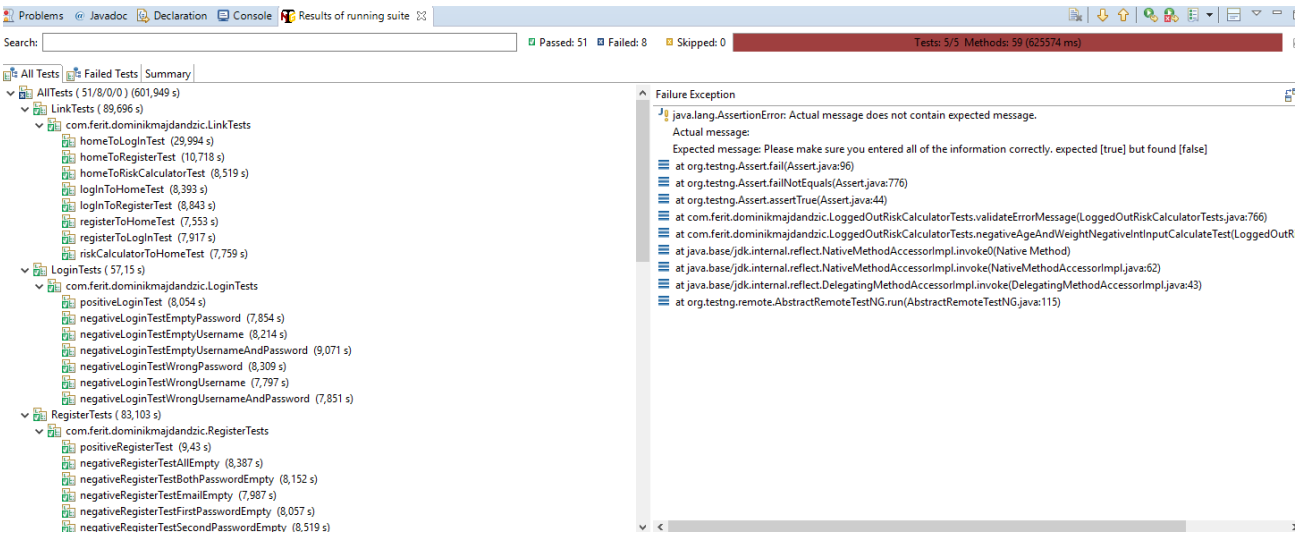


Slika 4.2. *Pokretanje testne grupe*

Nakon pokretanja testne grupe, početi će se redom izvoditi testovi te će se rezultati prikazati u konzoli i TestNG-u. Na slici 4.3 je prikazan izgled rezultata u konzoli, a na slici 4.4 je prikazan izgled rezultata u TestNG-u. Može se uočiti kako TestNG ima više informacija o provedenim testovima od konzole. U gornjem dijelu se može vidjeti broj testova koji su uspjeli, koji nisu uspjeli i koji su preskočeni. Također pokraj imena testova ili testnih grupa moguće je vidjeti ukupno vrijeme izvođenja testova. Iz slike 4.4 se može iščitati da je za pokretanje svih 59 testova bilo potrebno 10 minuta što je znatno smanjilo vrijeme provođenje testove jer bi ručnim ispitivanjem to trajalo minimalno 30 minuta..

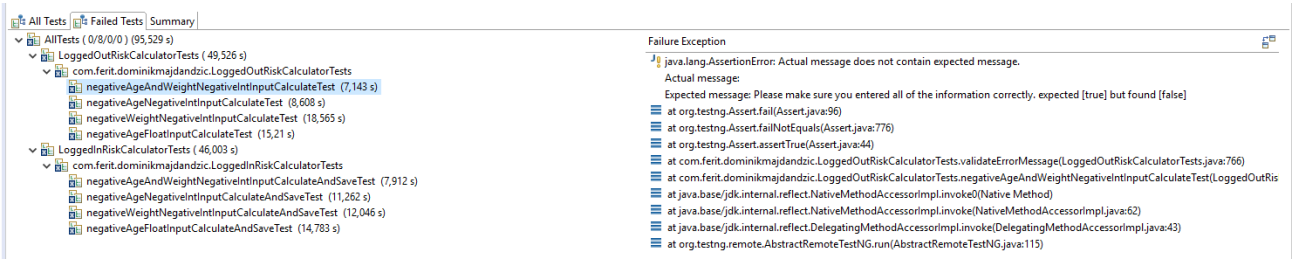
```
===============================================
AllTests
Total tests run: 59, Failures: 8, Skips: 0
===============================================
```

Slika 4.3. *Prikaz rezultata testne grupe AllTests u konzoli*



Slika 4.4. *Prikaz rezultata testne grupe AllTests u TestNG-u*

Pomoću TestNG mogu se odvojeno prikazati rezultati samo neuspjelih testova, gdje se na lijevoj strani može odabrati specifični test, a na desnoj strani se može vidjeti koji dijelovi testa nisu uspjeli. To je prikazano na slici 4.5. Analizom te slike može se uočiti da se odabrani test nije uspio izvršiti jer se nije moglo potvrditi da se određeni tekst nalazi na stranici, tj. taj tekst ne postoji.

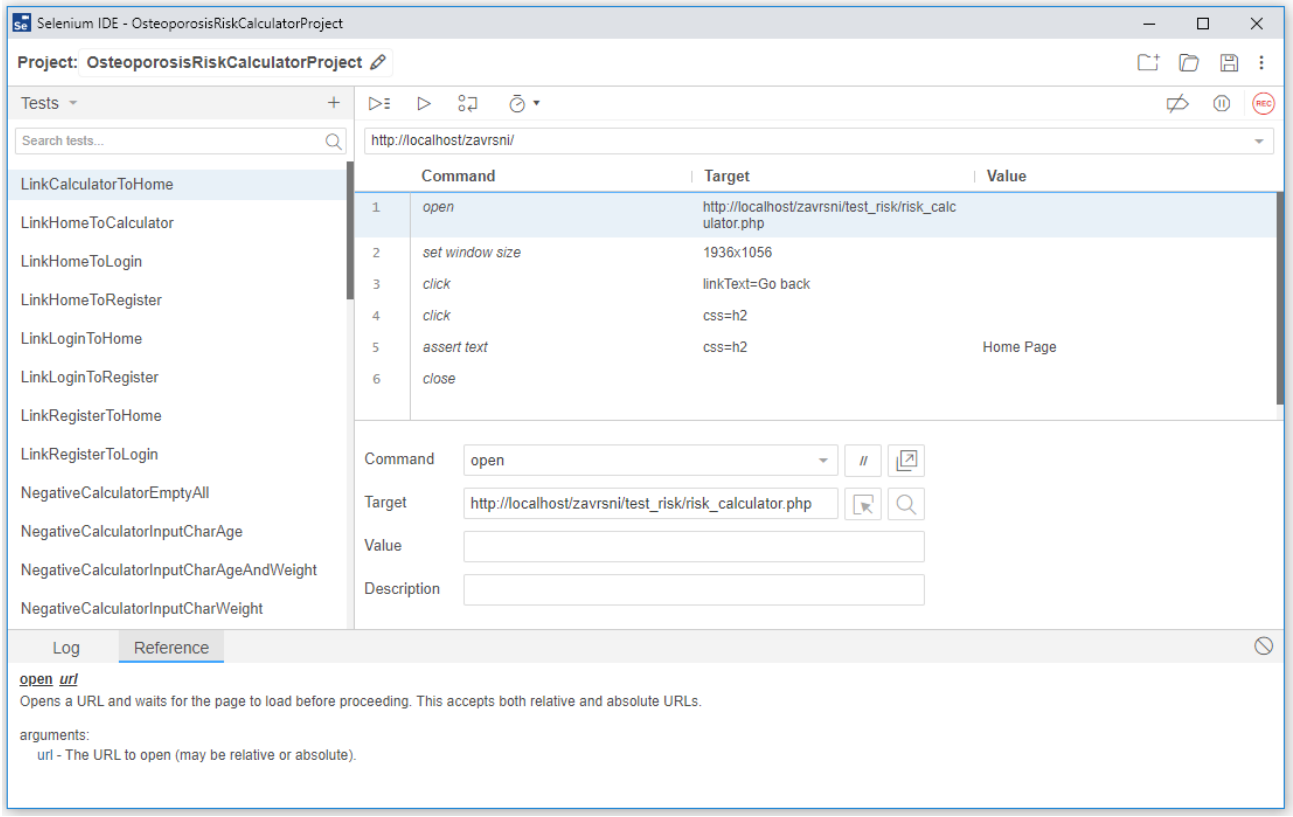


Slika 4.5. *Neuspjeli testovi testne grupe AllTests*

4.4. Ispitivanje pomoću Selenium IDE-a

Za korištenje Selenium IDE-a potrebno je imati instaliran Google Chrome ili Mozilla Firefox web preglednik na računalu na kojemu se ispituje web aplikacija. Zatim se na taj preglednik treba skinuti Selenium IDE proširenje koje se može naći na web trgovinama već navedenih preglednika. Za pokretanje je potrebno otvoriti web preglednik te među proširenjima pronaći Selenium IDE i kliknuti na njega. Na slici 4.6 je prikazano korisničko sučelje Selenium IDE-a. Na lijevoj strani se nalazi navigacija gdje se mogu vidjeti postojeći testovi, testne grupe i pokrenuti testovi. Na donjem dijelu sučelja se nalazi konzola gdje će se ispisivati koraci izvođenja i rezultati pojedinih testova ili testnih grupa. Osim konzole, na donjem dijelu se nalazi opis naredbe koja je trenutno odabrana. Na

sredini (glavnom dijelu) sučelja nalaze se koraci snimljenog testa. Desno iznad koraka snimljenog testa se nalaze opcije za snimanje testa, za pauziranje pri pojavi greške te opcija za onesposobljavanje prijelomnih točaka (engl. *breakpoint*). Dok se lijevo iznad koraka snimljenog testa nalaza opcije za pokretanje testa, pokretanje testne grupe, izvođenje testa liniju po liniju i opcija za podešavanje brzine izvođenja testova.

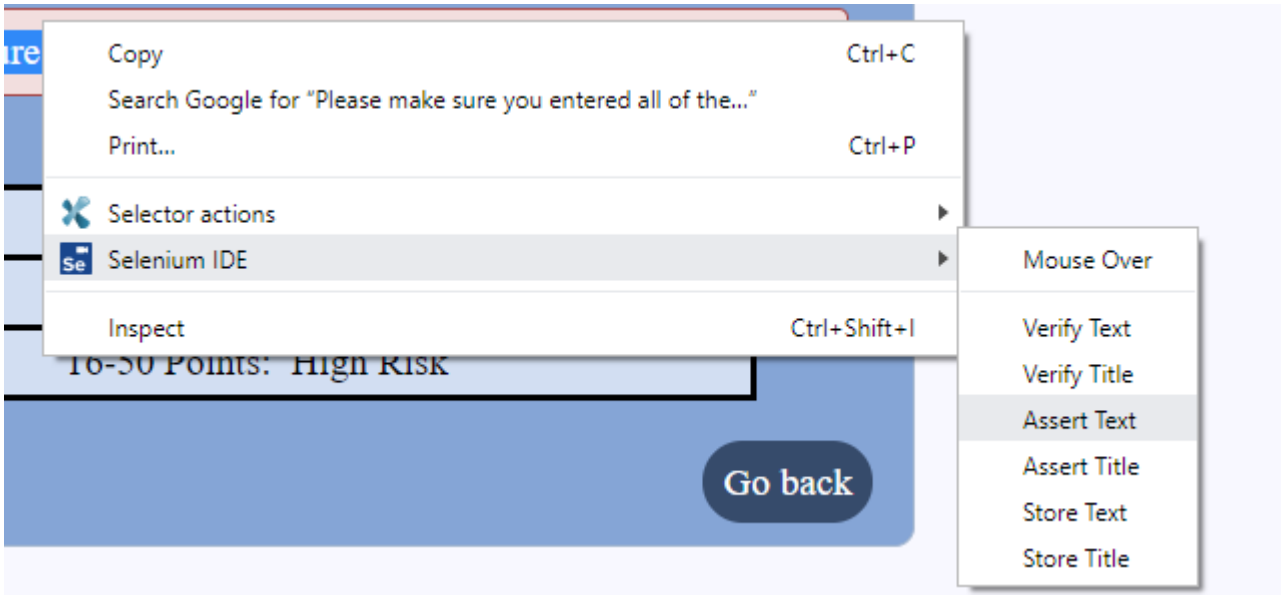


Slika 4.6. *Korisničko sučelje Selenium IDE-a*

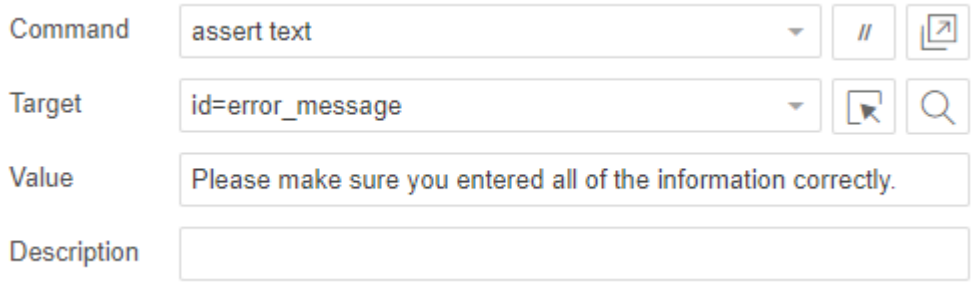
4.4.1. Snimanje testova

Nakon što se pokrene snimanje testa otvara se zadana početna web stranica u pregledniku. Nakon što se otvorila početna stranica svaka radnja na njoj će se spremati kao korak testa (npr. klikovi, unos teksta itd.). Tijekom snimanja testova, unutar preglednika desnim klikom se može odabrati različite opcije za potvrdu ili spremanje teksta (slika 4.7), pomoću kojih se može potvrditi promjena dinamičkih elemenata na stranici. Nakon što su se snimile sve željene radnje potrebno je zaustaviti snimanje. Naravno, osim samog snimanja testa moguće je i ručno ubaciti i uređivati korake za izvođenje. Potrebno je odabrati naredbu koja se želi izvesti, element na koji se ona primjenjuje te po

potrebi vrijednost koja će se koristiti (npr. tekst koji se želi unijeti). Primjer ručnog unošenja koraka izvođenja testa je prikazan na slici 4.8.



Slika 4.7. *Prikaz opcija za potvrdu i spremanje teksta*



Slika 4.8. *Prikaz ručnog unošenja koraka izvođenja testa*

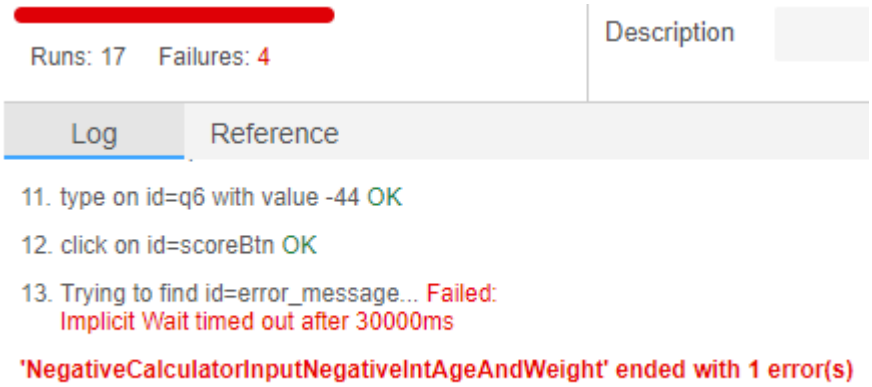
4.4.2. Pokretanje testova i prikaz rezultata

Moguće pokretati pojedinačni test ili testnu grupu. Osim pokretanja cijelog testa odjednom moguće je prolaziti kroz test korak po korak klikom na „*Step over current commant*“. Kao što je prije navedeno moguće je podesiti brzinu izvođenja testa jer je moguće da se neki testovi neće uspješno izvršiti zbog prebrzog prolaženja kroz korake. Također je moguće postaviti prijelomne točke na specifične korake. One će izvođenje testa pauzirati kada se dođe do označenog koraka. Kako bi se nastavio test može se ponovno kliknuti na pokretanje testa ili se može nastaviti prolaskom korak po

korak. Izgled prijelomne točke je prepoznatljiv po plavoj boji rednog broja koraka i prikazan je na slici 4.9.

Slika 4.9. *Izgled prijelomne točke*

Tijekom izvođenja testa postoji implicitno vrijeme čekanja. To je vrijeme koje će se pokušavati izvršiti jedan korak testa. Ako se neki od koraka testa ne može izvršiti (npr. nije moguće pronaći specifični tekst jer on ne postoji), onda će se test prekinuti i u rezultatima će biti prikazano koji korak se nije uspio izvršiti. Rezultati pokrenute testne grupe su prikazani na slici 4.10. U gornjem dijelu se vidi ukupni broj pokrenutih testova i broj neuspjelih testova. Pregledom kroz konzolu može se vidjeti za svaki korak svakog testa je li on uspješno izveden. Može se zaključiti da je prikaz rezultata WebDriver-a pomoću TestNG bolji od prikaza rezultata IDE-a. To je zato što kod TestNG prikazuje više informacija kao što su vrijeme izvođenja testova i razlog zašto test nije bio uspješan. Također TestNG ima uredan pregled svih testova dok se kod IDE-a mora listati kroz konzolu da bi se došlo do testa koji se traži.

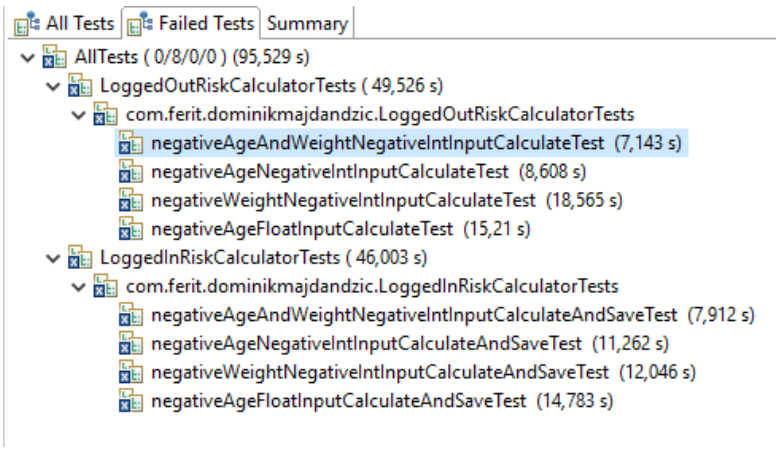


Slika 4.10. *Rezultati testne grupe*

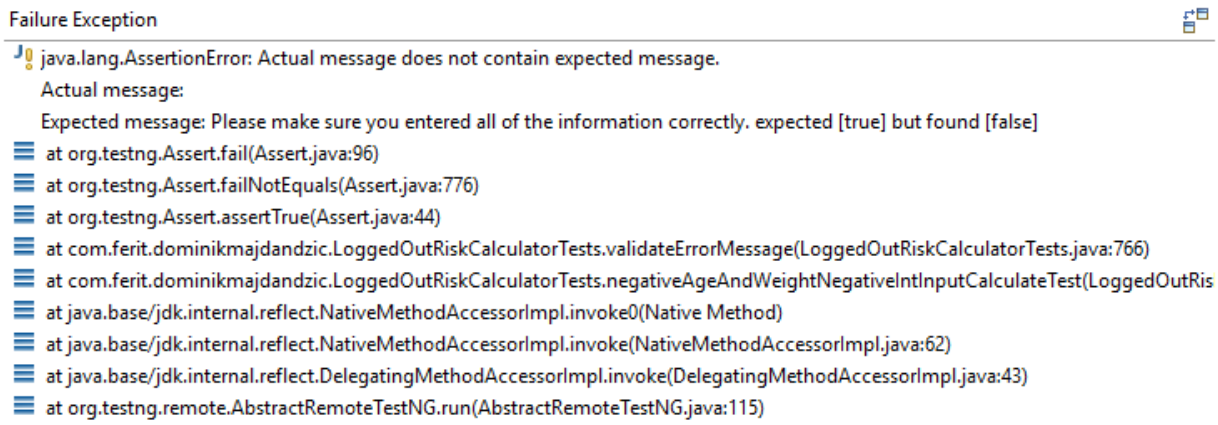
4.5. Poboljšanja web aplikacije na osnovi rezultata ispitivanja

Za analizu rezultata koristiti će se rezultati ispitivanja pomoći Selenium Webdriver-a zato što je on češće korišten alat i svi neuspjeli testovi su uredno prikazani na jednom mjestu. Rezultati, specifično neuspjeli testovi, testne grupe *AllTests* (u njoj se nalaze svi testni slučajevi, slika 4.4) su prikazani na slici 4.11. U rezultatima se mogu vidjeti nazivi svih testova koji se nisu uspjeli izvršiti. Klikom na

neki od tih testova, sa desne strane se prikazuju dijelovi koji se nisu uspjeli izvesti. Ti dijelovi za test unosa negativne dobi i težine su prikazani na slici 4.12. Moguće je vidjeti točno u kojim datotekama unutar cijelog projekta i u kojim linijama koda tih datoteka je došlo do greške.



Slika 4.11. *Prikaz neuspjelih testova testne grupe AllTests*



Slika 4.12. *Prikaz detalja neuspjelog testa unosa negativne dobi i težine*

Analizom dobivenih rezultata (gledajući nazive testova) lako je utvrditi da se problem nalazi u izračunu rizika. Izračun rizika ne bi trebao raditi sa negativnim brojevima kao ulazima za dob i težinu korisnika ili sa decimalnim brojevima kao ulazima za dob. No, umjesto prikaza poruke o nepravilno unesenim podatcima, prikazuje se izračunati rizik. Taj problem nije teško riješiti. Jednostavno se unutar JavaScript koda za izračun rizika treba ubaciti uvjet da ako se unesena dob i težina ne nalaze unutar razumnih granica neka se ispiše poruka o nepravilno unesenim podatcima. Također za dob se može umjesto unosa brojeva koristiti padajući izbornik sa godinama rođenja. Svi ostali testovi su pravilno izvedeni te ih nije potrebno detaljno analizirati.

5. ZAKLJUČAK

Izrađena web aplikacija za procjenu rizika obolijevanja od osteoporoze je primjer primjene aplikacija za medicinske svrhe. Ona pomoću unesenih podataka korisniku daje uvid o procijenjenom riziku. Aplikacija se sastoji od baze podataka korisnika, u kojoj se spremaju rezultati procjene rizika korisnika, i od programa koji procjenjuje rizik. Iako je aplikacija relativno jednostavna, postoji velika količina različitih slučajeva s različitim unosima koje je potrebno ispitivati. Ispitivanje se postiglo automatiziranim ispitivanjem pomoću alata Selenium. Za pisanje testnih scenarija pomoću Selenium Webdrivera i snimanje testnih scenarija pomoću Selenium IDE je potrebno uložiti pristojnu količinu vremena, ali nakon što su testni scenariji definirani za njihovo izvođenje je potrebno znatno manje vremena nego kada bi se ručno ispitivalo. To je vidljivo i iz samih rezultata ispitivanja. Samim time što Selenium ubrzava proces ispitivanja aplikacije, također ubrzava i olakšava sam proces razvoja aplikacije. Moguća poboljšanja web aplikacije mogla bi se baviti uvođenjem novih čimbenika rizika i razvojem novog izraza za procjenu rizika obolijevanja od osteoporoze. Naravno, za novonastalu poboljšanu aplikaciju biti će potrebno urediti trenutne i izraditi nove testne scenarije za automatizirano ispitivanje.

LITERATURA

[1] „Technopedia: Manual Testing“, https://www.techopedia.com/definition/29843/manual-testing [15.9.2019.]

[2] „Technopedia: Automated Website Testing“, https://www.techopedia.com/definition/30007/automated-website-testing [15.9.2019.]

[3] „Software Testing Fundamentals: Unit Testing“, http://softwaretestingfundamentals.com/unit-testing/ [15.9.2019.]

[4] „Software Testing Fundamentals: Functional Testing“, http://softwaretestingfundamentals.com/functional-testing/ [15.9.2019.]

[5] „Software Testing Fundamentals: Integration Testing“, http://softwaretestingfundamentals.com/integration-testing/ [15.9.2019.]

[6] „Software Testing Fundamentals: Performance Testing“, http://softwaretestingfundamentals.com/performance-testing/ [15.9.2019.]

[7] „Software Testing Fundamentals: Acceptance Testing“, http://softwaretestingfundamentals.com/acceptance-testing/ [15.9.2019.]

[8] „Software Testing Fundamentals: System Testing“, http://softwaretestingfundamentals.com/system-testing/ [15.9.2019.]

[9] „R. Vogels, Usersnap: A 6-Step Guide to Web Aplication Testing“, https://usersnap.com/blog/web-application-testing/?fbclid=IwAR1jtUiIHdAEyV0-j0Kr1i4pqOmeFkxJ-f-DILqzfA-gF3nzqv9otkqAQ1E [15.9.2019.]

[10] „Elena K., RubyGarage: Using automated testing in web development“, https://rubygarage.org/blog/using-automated-testing-in-web-development [15.9.2019.]

[11] „SeleniumHQ“, https://www.seleniumhq.org/ [15.9.2019.] [15.9.2019.]

[12] „Smartbear: TestComplete“, https://smartbear.com/product/testcomplete/overview/ [15.9.2019.]

[13] „IBM: Rational Functional Tester“, https://www.ibm.com/us-en/marketplace/rational-functional-tester [15.9.2019.] [15.9.2019.]

[14] „eggplant“, https://eggplant.io/ [15.9.2019.] [15.9.2019.]

[15] „Ranorex“, https://www.ranorex.com/ [15.9.2019.] [15.9.2019.]

[16] „Apache JMeter“, https://jmeter.apache.org/ [15.9.2019.] [15.9.2019.]

[17] „MayoClinic: Osteoporosis“, https://www.mayoclinic.org/diseases-conditions/osteoporosis/symptoms-causes/syc-20351968?fbclid=IwAR2BRN8OOvD40PicCEp17uUdpLthGi2g_o_QVxw-is3RO5DjxTm30XOTx6E [15.9.2019.]

[18] “Medscape: Osteoporosis Risk SCORE”, https://reference.medscape.com/calculator/osteoporosis-risk-score [15.9.2019.] [15.9.2019.]

[19] „*Lydick E, Cook K, Turpin J, et. al. Development and validation of a simple questionnaire to facilitate identification of women likely to have low bone density*“, Am J Manag Care. 1998 Jan;4(1):37-48. [15.9.2019.]

[20] „*Geusens P, Hochberg MC, van der Voort DJ, et. al. Performance of risk indices for identifying low bone density in postmenopausal women*“, Mayo Clin Proc. 2002 Jul;77(7):629-37. [15.9.2019.]

[21] „QFracture“, https://qfracture.org/ [15.9.2019.]

[22] “FRAX: Calculation tool”, https://www.sheffield.ac.uk/FRAX/tool.aspx [15.9.2019.] [15.9.2019.]

[23] “IOF: IOF risk check”, https://www.iofbonehealth.org/iof-one-minute-osteoporosis-risk-test [15.9.2019.]

[24] „Technopedia: Hypertext Markup Language“, https://www.techopedia.com/definition/1892/hypertext-markup-language-html [15.9.2019.]

[25] „Technopedia: Cascading Style Sheet (CSS)“, https://www.techopedia.com/definition/26268/cascading-style-sheet-css [15.9.2019.]

[26] „Technopedia: JavaScript (JS)“, https://www.techopedia.com/definition/3929/javascript-js [15.9.2019.]

[27] „Technopedia: MySQL“, https://www.techopedia.com/definition/3498/mysql [15.9.2019.]

[28] „Technopedia: PHP: HypertextPreprocessor (PHP)“, https://www.techopedia.com/definition/24406/php-hypertext-preprocessor-php [15.9.2019.]

[29] „Apache Maven Project“, https://maven.apache.org/ [15.9.2019.]

[30] „Eclipse foundation“, https://www.eclipse.org [15.9.2019.]

[31] „TestNG: TestNG Eclipse plug-in“, https://testng.org/doc/eclipse.html [15.9.2019.]

SAŽETAK

Web aplikacija razvijena u ovome radu služi da korisniku procjeni rizik obolijevanja od osteoporoze. Pomoću nje, korisnik može izračunati rizik da će u budućnosti oboljeti od osteoporoze. To se čini unosom šest čimbenika rizika: dob, rasa, težina, broj prijeloma koji nisu uzrokovani traumom, prisutnost reumatoidnog artritisa i prijašnje korištenje estrogena. Ti se podatci zatim obrađuju pomoću zadanog izraza te se dobiva rezultat. Korisniku je također omogućena registracija i prijava u sustav čime dobiva dodatnu opciju da spremi svoje rezultate. Sve funkcionalnosti web aplikacije su ispitane na dva načina pomoću alata Selenium. Prvi način je pomoću Selenium WebDrivera-a gdje su se testni scenariji pisali pomoću Java programskog jezika, a drugi način je pomoću Google Chrome ili Mozilla Firefox proširenja Selenium IDE gdje se testni scenariji generiraju snimanjem radnji korisnika. Na temelju praktičnog dijela rada opisani su razvoj i automatizirano ispitivanje web aplikacije pomoću Selenium-a.

Ključne riječi: automatizirano ispitivanje, osteoporoza, Selenium, web aplikacija

ABSTRACT

Title: Automated testing of web application for osteoporosis risk assessment by using Selenium tool

The web application developed in this paper serves to evaluate a users risk of osteoporosis. With it, the user can calculate the risk of developing osteoporosis in the future. This is done by inputting six risk factors: age, race, weight, number of nontraumatic fractures, presence of rheumatoid arhritis and previous use of estrogen. This data is then processed using a predefined expression followed by the display of the results. The user also has the ability to register and log in into the site, which gives him an additional option to save his results. All the functionality of the web application has been tested in two different ways using the Selenium tool. Firstly using Selenium Webdriver where the test scripts are written using the Java programming language and secondly using the Google Chrome or Mozilla Firefox Selenium IDE extension where the test scripts are generated by recording a users actions. The development and automated testing of a web application using Selenium are described based on the practical part of the paper.

Keywords: automated testing, osteoporosis, Selenium, web application

ŽIVOTOPIS

Dominik Majdandžić rođen je 6. prosinca 1997. godine u Osijeku. Pohađao je OŠ Augusta Šenoe u razdoblju od 2004. do 2012. godine. 2012. godine upisuje III. Gimnaziju u Osijeku gdje je sudjelovao na županijskom natjecanju iz programiranja. 2016. godine završava srednju školu i upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija.

Potpis: _____

PRILOZI (na CD-u)

Prilog 1. Word i PDF dokument rada

Prilog 2. Programski kod web aplikacije

Prilog 3. Testni scenariji web aplikacije