Primjena agilne metode za testiranje mobilnih aplikacija

Bizik, Tea

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet Osijek

Dvopredmetni diplomski studij informatologije i informacijske tehnologije

Tea Bizik

Primjena agilne metode za testiranje mobilnih aplikacija

Diplomski rad

Mentor: izv. prof. dr. sc. Boris Badurina

Osijek, 2021.god.

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet Osijek

Odsjek za informacijske znanosti

Dvopredmetni diplomski studij informatologije i informacijske tehnologije

Tea Bizik

Primjena agilne metode za testiranje mobilnih aplikacija

Diplomski rad

Područje društvenih znanosti, polje informacijskih i komunikacijskih znanosti,
grana informacijskih sustava i informatologije

Mentor: izv. prof. dr. sc. Boris Badurina

Osijek, 2021.god.

Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnog odnosno diplomskog rada.

IZJAVA

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napravio te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s napisanim izvorom odakle su preneseni.
Svojim vlastoručnim potpisom potvrđujem da sam suglasan da Filozofski fakultet Osijek trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta Osijek, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, datum 20.9.2021.

Tea Bizik, 0122221045 7
ime i prezime studenta, JMBAG

Sažetak

Cilj ovog rada je skrenuti pozornost na važnost i prednosti korištenja agilne metode na razini projekta. Također, cilj rada je i prikazati primjenu agilne metode u raznim razvojnim softverskim okruženjima. Kroz rad je predstavljena agilna metodologija te je napravljena usporedba tradicionalnog i agilnog načina vođenja projekata. Konkretno, agilna je metoda uspoređena s najpoznatijom metodom tradicionalnog vođenja projekata, Waterfall, koji je tipičan za određena područja inženjerskog dizajna. Nadalje, kroz rad je detaljno opisan glavni dio dokumentacije potrebne za testiranje, a riječ je o test scenariju i test slučaju te razlike i povezanost među istima. Sljedeće su opisane i primjerom potkrijepljene metode, vrste i razine testiranja. Za potrebe ovog rada opisane su osnovne metode testiranja, odnosno metoda crne, bijele i sive kutije te najpoznatije vrste testiranja. Osim što su najpoznatije vrste testiranja, objašnjene su i zbog toga što se niti jedno testiranje ne provodi izostavljajući iste. Kada se u radu predstavljaju razine testiranja, govori se o jediničnom, integracijskom, sustavnom te testiranju prihvatljivosti. Nakon toga slijedi poglavlje korisnička priča u kojoj se objašnjava odnos i uloga product owner-a i QA testera kroz samu ulogu i stvaranje korisničke priče. Da bi se agilna metoda mogla provesti, potrebno je usvojiti prethodno navedene pojmove i znanja. Zbog toga se na samom kraju teorijskog dijela rada navode najpoznatiji alati za provođenje agilne metodologije te se oprimjeruje svaki. Za potrebe istraživanja, odabrane su četiri tehnike kod agilnog testiranja te su iste provedene i prikazane na primjeru testiranja Facebook aplikacije na čijem se primjeru zasniva cijeli rad.

Ključne riječi: testiranje, agilna metoda, tradicionalna metoda, metode testiranja, vrste testiranja

Sadržaj

1. Uvod

Testiranje ili osiguranje kvalitete podrazumijeva sprječavanje pogrešaka i nedostataka proizvoda prije nego izađu na tržište ili ih kupci počnu koristiti. Zadaća svakog testera je identificiranje, analiziranje, prijavljivanje pogrešaka, pisanje dokumentacije te praćenje rješavanja nedostataka proizvoda. Svaki se proizvod testira postepeno, od početka razvoja. Jedna od prvih odluka s kojom se suočava svaki tester za svaku od implementacija projekata je "Koju razvojnu metodu koristiti?

Dvije osnovne, najpopularnije metodologije su Waterfall koji se naziva najpoznatijom metodom tradicionalnog pristupa te Agile, specifična vrsta brzog razvoja aplikacija i novija od Waterfall-a. Agilna metoda poznata je po svojoj fleksibilnosti, dok je Waterfall strukturirana metoda razvoja softvera. Odabir prave metodologije za provođenje projekta ovisi o preferencijama i prirodi svakog projekta. Neki projekti zahtijevaju iterativni postupak, a drugi sekvencijalni pristup. Glavna razlika između Agile i Waterfall metode je u tome što se projekti slapova dovršavaju sekvencijalno, dok se agilni projekti dovršavaju iterativno u ciklusu. I agilna metodologija i metodologija vodopada nose svoj niz prednosti i nedostataka. Sve u svemu, oboje mogu biti korisni za tim za razvoj softvera. Koja će metoda biti odabrana vrlo ovisi o vrsti projekta i okolnostima, no u ovom će se radu govoriti o agilnoj metodi.

Agilna metodologija ima dva ključna elementa: timski rad i vrijeme. Umjesto da kreira vremensku crtu za jedan veliki projekt razvoja softvera, agilni projekt dijeli na pojedinačne dijelove koji se mogu isporučiti. Te se faze s vremenskim okvirom nazivaju sprintevi i traju samo nekoliko tjedana. Kada se završi svaki sprint, za planiranje sljedeće koristi se povratna informacija iz prethodne faze. Kako bi se jedno testiranje uspješno izvršilo, potrebno je da se tester pripremi za testiranje i osigura neophodnu dokumentaciju, koja mu pomaže da testiranje provede što kvalitetnije. Kod dokumentacije su obavezni test scenarij i test slučaj. Test slučaj podrazumijeva skup radnji izvršenih za provjeru određenih značajki ili funkcionalnosti, dok Test scenarij podrazumijeva bilo koju funkcionalnost koja se može testirati.

Test scenarij i test slučaj su obavezna dokumentacija pri testiranju te se koriste neovisno o odabranoj metodi, vrsti ili razini testiranja proizvoda. Kroz rad će se detaljno objasniti svaka navedena metoda, vrsta i razina testiranja.

Korisničke priče važan su dio agilnog pristupa i metodologije. Otvaraju prostor za raspravu i predstavljaju namjere svim dionicima kako bi svi koji sudjeluju u razvoju proizvoda razumjeli što i zašto treba učiniti. Korisničke priče može napisati klijent, product owner ili razvojni tim. Product owner donosi odluke za tim, testeri provode zadano, a product owner ostaje odgovoran.

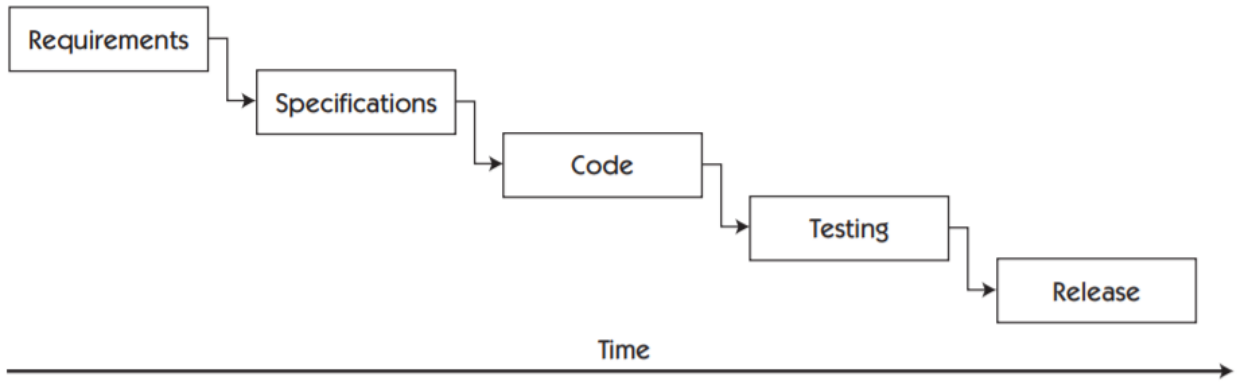
Postoji nekoliko agilnih metoda u praksi kojima se testeri koriste da bi izvršili zadatke i osigurali kvalitetan proizvod, a u ovom će se radu prikazati kako se agilna metoda primjenjuje u alatima za provođenje iste kao što su Scrum, Kanban i JIRA.

2. Agilna metoda

Jedna od prvih odluka s kojom se suočavaju zaposlenici prije samog početka provođenja određenog projekta jest „Koju razvojnu metodu trebamo koristiti?" Pojam razvojne metode ukratko se može opisati kao način organiziranja rada na razvoju softvera. Naravno, agilna metoda nije jedina razvojna metoda, ali se značajno razlikuje od svih ostalih.

Postoje dvije osnovne skupine, a to su tradicionalna i agilna razvojna metoda. Najpoznatiji model tradicionalne metode je waterfall, a opisat će se radi uočavanja razlike između agilne metode i kao odgovor zašto i kako je nastala agilna metoda

Između 70-ih i 90-ih godina 20. stoljeća nastale su opširne, tradicionalne metode.[1] Model waterfall, kako samo ime govori – slap – kada voda jednom poteče preko ruba, teče prema dolje i nikada se ne okreće natrag prema vrhu, odnosno, u waterfall metodologiji nema mogućnosti vraćanja iz jedne faze u prethodnu.[2] U stvarnom projektu koristeći se tradicionalnom metodom, svaka se faza završava prije nego što sljedeća započne. (Slika 1.)



Slika 1. Tradicionalna metoda

[1] Stefanović, Kristina; Tošić, Goran. Agilne metodologije razvoja softvera: osnovne agilne metodologije, str. 3. URL: http://files.kristina-stefanovic.webnode.com/200000105-0059b016e2/Survey.pdf (2021-01-09)
[2] Kannan, Vaishnavi; Jhajharia Smita; dr. Verma, Seema. Agile vs waterfall: A Comparative Analysis. // International Journal of Science, Engineering and Technology Research (IJSETR) 3, 8(2014), str.1. URL: http://ijsetr.org/wp-content/uploads/2014/10/IJSETR-VOL-3-ISSUE-10-2680-2686.pdf (2021-01-09)

Stoga se waterfall koristi kada je slika konačnog proizvoda jasna i dostupna, a zahtjevi su dobro definirani i ne podliježu promjenama.[3] U današnjoj svakodnevici ovom metodom projekti se puno rjeđe provode u IT industrijama od agilne. Najčešći i najlogičniji primjer primjene iste je u građevinskom sektoru. Primjerice, građevinari ne mogu obojati zidove ako ih prethodno nisu sazidali. Isto tako, ako su zidovi već sazidani, vraćanje na izgradnju temelja je nepotrebno. Ovakav primjer govori o važnosti pomnog planiranja svih faza i aktivnosti prije samog provođenja projekta. Tradicionalna metoda loš je odabir za velike projekte i projekte koji su tijeku jer zbog svoje linearnosti ne postoji mogućnost uvođenja promjena. Na početku projekta klijent objavljuje svoje zahtjeve i ima uvid u proizvod tek kada su sve faze gotove. Tu nastaje problem jer u vremenskom periodu u kojem se sve faze odrađuju jedna po jedna, druga osoba može doći do iste ili slične ideje i razviti ju prije navedenog klijenta. Primjerice, da se Facebook razvijao tradicionalnom metodom, u međuvremenu bi postojala mogućnost da se netko tko nije Mark Zuckerberg, vlasnik i kreator Facebook-a, dosjetio da izradi društvenu mrežu poput Facebooka i objavio ju prije nego Mark Zuckerberg. Također, u istom vremenskom razdoblju neke tehnologije mogu postati zastarjele ili neupotrebljive te da bi se uvele promjene potrebno je projekt započeti ponovno od prve faze i zbog toga se nedostatak iteracija i komunikacije tijekom provođenja projekta smatra nedostatkom te ujedno i razlogom zašto se sve veći broj IT industrija odlučuje za agilnu metodu.

Zbog načina na koji su tradicionalne metode pokušale nametnuti formaciju kojom se softver osmišljava, dokumentira, razvija i testira, krajem 90-ih godina 20. stoljeća, grupa projektanata (Ken Beck, Alstair Cockburn i dr. osnivaju Agile Alliance) koja se protivila takvoj nametnutoj formaciji, postaje začetnik agilne razvojne metode te 2001. godine donose *Agile Software Development Manifesto* pokušavajući da naglase ulogu koju fleksibilnost može odigrati u spretnijem i bržem razvoju softvera. U Manifestu su zabilježeni novi principi agilne metode te je uspostavljen novi sistem vrijednosti.[4]

U engleskom rječniku riječ "Agile" znači sposobnost brzog i jednostavnog premještanja.[5] Agilna metoda je timski pristup razvoju i kao takav naglašava brzu isporuku proizvoda u kompletnim funkcionalnim komponentama. Pomaže projektnim timovima da se na kontroliraniji

[3] Kannan, Vaishnavi; Jhajharia Smita; dr. Verma, Seema. Nav. dj., str. 2.
[4] Stefanović, Kristina; Tošić, Goran. Nav. dj.,str.3.
[5] ISTQB Agile Tester, str. 4. (2021-01-10)

način nose s mnogim najčešćim zamkama projekta poput troškova, predvidljivosti rasporeda i uspona opsega. U tradicionalnim metodama razvoja softvera poput modela waterfall, projekt može potrajati nekoliko mjeseci ili godina, a kupac možda neće moći vidjeti krajnji proizvod do završetka projekta. Za razliku od toga, agilni projekti imaju sprinteve o kojima će se u narednim poglavljima više govoriti, te iteracije, odnosno ponavljanja koja su kraća, a mogu varirati od dva tjedna do dva mjeseca tijekom kojih se razvijaju i klijentu postepeno isporučuju značajke određene propisanim zahtjevima.[6] To znači da klijent nakon svake isporuke može promijeniti zahtjeve ili dodati nove te biti u uključen u tijek provođenja projekta. Ukratko, agilni razvoj softvera odnosi se na skupinu metoda temeljenih na iterativnom razvoju, gdje se zahtjevi i rješenja razvijaju kroz suradnju samoorganizirajućih višefunkcionalnih timova. Agilna metoda predstavlja set principa koji promoviraju fleksibilnost i prilagodljivost čestim promjenama, za razliku od Waterfall metode.

Lagani procesni okvir za agilni razvoj softvera, ujedno i podskup agilne metode, te najčešće korišteni je Scrum koji olakšava međusobne komunikacije između timova.[7] Nakon što se razjasne sva potrebna polja i znanja vezana za agilno testiranje, rad će detaljnije predstaviti Scrum metodu kao i alate koji se koriste kako bi se implementirala agilna metoda, odnosno, govorit će se o alatima Kanban i JIRA.

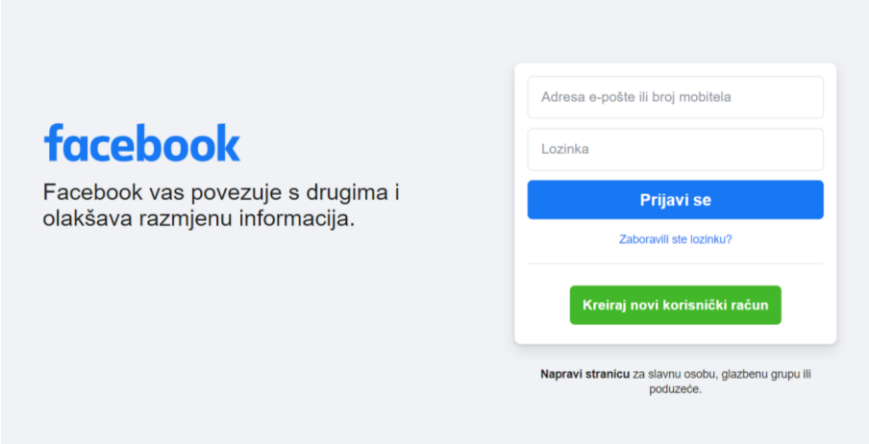
[6] Isto.
[7] Gregory, Janet; Crispin, Lisa. Agile testing: a practical guide for testers and agile teams. Str.42. URL: http://index-of.co.uk/Software-Testing/AGILE_TESTING_-_A_PRACTICAL_GUIDE_FOR_TESTERS_AND_AGILE_TEAMS.pdf (2021-01-12)

3. Dokumentacija za testiranje

Potrebno je da se tester pripremi za testiranje razvojnih aplikacija i osigura neophodnu dokumentaciju, koja će mu pomoći da testiranje provede što kvalitetnije kako bi se jedno testiranje uspješno izvršilo. Dokumentacija[8] koja je testeru potrebna razlikuje se u razinama detalja koje sadrži, počevši od najopćenitijih uputa za testiranje, konkretnih zadataka, do onih najpreciznijih, kao što su naredbe koji treba pokrenuti. U sljedećim će se potpoglavljima predstaviti test scenarija i test slučaj ispitivanja jer podrazumijevaju glavni dio dokumentacije o testiranju.

3.1. Test scenarij

Test scenarij predstavlja svaku funkcionalnost s kojom se korisnik sreće tijekom korištenja softvera. [9] Drugim riječima, opisuje radnju koju korisnik može poduzeti s web mjestom ili aplikacijom. Također može opisati situaciju u kojoj se korisnik može naći dok koristi istu. Testeri često "izmišljaju" podatke o testiranju, ali podatci su obično jednostavni stoga se rezultati mogu lako provjeriti. Prilikom testiranja različitih scenarija, podaci i protok moraju biti realni. Primjerice,među prvim funkcionalnostima Facebook društvene mreže s kojom se njegovi korisnici susreću prilikom prijave na istu je *"Prijavi se"* na osobni račun. *"Prijavi se"* je funkcionalnost koja omogućuje da samo korisnik koji ima prethodno registriran račun na Facebook-u može pristupiti svojem računu uz unos ispravnog e-maila i zaporke. (Slika 2.)



Slika 2. Facebook Login

[8] Specifikacija koja opisuje softver.
[9] Guru99. URL: https://www.guru99.com/ (2021-02-01)

Primjer test scenarija za navedenu funkcionalnost može glasiti: Ispitati funkcionalnost prijavljivanja na svoj Facebook račun. Primjer istog test scenarija može glasiti i: Istražiti funkcionalnost gumba "*Prijavi se*" na FB. Način na koji će se test scenarij formulirati razlikuje se od testera do testera. Važno je da je test scenarij dobro definiran kako bi pomogao testeru da na temelju njega kreira detaljne test slučajeve o kojima će se govoriti u sljedećem potpoglavlju. Tako se omogućuje potpuna pokrivenost te funkcionalnosti. Ovisno o tome koliko je funkcionalnost složena, iz jednog se test scenarija može definirati jedan ili više test slučajeva koje će tester koristiti tijekom testiranja. S obzirom na to da je navedena funkcionalnost prilično jednostavna jer ne sadrži velik broj podataka koji se trebaju provjeriti. Iz takvog test scenarija može nastati tek nekoliko test slučajeva jer u okviru te funkcionalnosti postoji jedan gumb, gumb "*Prijavi se*", pomoću kojeg se izvršava samo jedna naredba.

Test scenarij odnosi se i na složenije funkcionalnosti. Takve funkcionalnosti koriste više ulaznih podataka. U tom je slučaju potrebno ispisati i istestirati mnogo više test slučajeva u odnosu na prethodni primjer. Jedna od takvih stranica može izgledati kao stranica sa slike. (Slika 3.)

Slika 3. Registracija Facebook korisnika

Na ovoj stranici potrebno je unijeti puno više detalja u odnosu na Facebook Prijavi se stranicu. U ovom slučaju bi se od korisnika tražilo da popuni sljedeća polja: ime, prezime, broj

mobitela ili adresa e-pošte i lozinka. Također, od korisnika se traži da unese dan, mjesec i godinu rođenja te da odabere spol. Dodatni podaci koji se traže od korisnika zahtijevaju da tester ispita puno više mogućnosti za eventualne pogreške nastale za vrijeme programiranja. Test scenarij za ovaj primjer mogao bi glasiti: Ispitati funkcionalnost registracije korisnika.

Test scenarij je u odnosu na test slučaja manje detaljan. On testera uvodi u zadatak koji je pred njim i predstavlja neku vrstu cilja koji treba ostvariti testiranjem. Test slučajevi predstavljaju podskup skeniranih scenarija, a potonji se temelji na funkcionalnosti i odnosi se na tijek rada. Pisanje test scenarija pomaže u prepoznavanju najkritičnijeg dijela sustava. Glavni razlog pisanja istih je provjera potpune funkcionalnosti sustava. Jednom kada su napisani, lako je odrediti testne primjere za njih. Također, pomažu u osiguravanju usklađenosti poslovnih procesa prema definiranom tijeku.[10] Nakon što su ovi testni scenariji finalizirani, testni se slučajevi mogu lako izvesti iz testnih scenarija.

3.2. Test slučaj

Testni slučajevi jedan su od najvažnijih dijelova životnog ciklusa razvoja softvera koji je odgovoran za izvedivost programa. Može se reći da je test slučaj niz aktivnosti koje se provode kako bi se potvrdilo radi li funkcionalnost koja se testira na očekivan način.[11]

Kroz prethodno potpoglavlje test scenarij definiran je kao funkcionalnost s kojom se korisnik sreće dok koristi softver. Ako je testni slučaj "način", tada je scenarij testiranja "što". To je niz mnogih testnih slučajeva koji se moraju izvršiti jedan za drugim kako bi provjerili funkcionalnost aplikacije. Sada će se kroz test slučaj opisati koje aktivnosti će se provesti da bi se ta funkcionalnost testirala. Tijekom pisanja test slučajeva potrebno je pratiti prethodno zadanu formu. Forma pisanja test slučajeva razlikuje se od jedne do druge IT tvrtke, no najčešće je forma istoga u obliku Excel tabelice, no također je moguće koristiti i softvere za upravljanje test slučajevima koji, uz brojne druge mogućnosti, pružaju mogućnost da se test slučaj automatski kreira i prati. Neovisno o tome u kojoj se formi pišu test slučajevi, potrebno je znati da se sastoji uglavnom od sljedećih stavki: (Slika 4.)

[10] EDUCBA. URL: https://www.educba.com/test-cases-vs-test-scenario/ (2021-02-05)
[11] Lučić, Mijo. Prakse testiranja programskih proizvoda, 2019., str. 18.

Test case ID	Test case description	Prerequisites	Test steps	Test data	Expected Result	Actual Result	Status	Created By	Date of creation	Executed By	Date of execution
TC001	The objective of this test case is to verify the 'Login' of Gmail account	1. User is authorized 2. Has an account in Gmail	1. Enter valid username 2. Enter valid password 3. Click on 'Login' button	1. User account should be present in Gmail	1. User should be able to login his Gmail account with his valid credentials 2. 'Invalid username or password' should get displayed if the username and password are not valid	1. If the valid credentials are entered then the user will be able to login his / her account 2. If invalid credentials are entered then nothing happens(the expected message is not displayed)	Fail	Rajesh	1/1/2016	Umesh	1/2/2016

Slika 4. Jednostavan primjer test slučaja

1) Test Case ID – jedinstveni broj test slučaja
2) Test Case Description – opis test slučaj zadatka
3) Pre-conditions – preduvjeti koje je potrebno ispuniti prije početka testiranja
4) Test Steps – koraci po kojima se radi testiranje
5) Test Data – podaci koje unosimo prilikom testiranja
6) Expected results – očekivani rezultati testiranja
7) Actual results – dobiveni rezultati testiranja
8) Test Status (pass/fail) – status testiranja (je li test bio uspješan ili ne)
9) Created by – ime odgovorne osobe koja je napisala test slučaj
10) Date of creation – datum kada je test slučaj napisan[12]

Kako bi se što bolje razumjelo kako se kreira test slučaj, u nastavku će se koristiti primjeri iz prethodnog poglavlja.

Primjer 1. Testiranje Facebook prijave

Test scenarij iz navedenog primjera glasi: *Ispitati funkcionalnost prijavljivanja na svoj Facebook račun.* Na početnoj Facebook stranici nalazi se dio za prijavljivanje. Isti se sastoji od polja za unos email adrese, polja za unos lozinke i gumba za prijavljivanje. Kako bi se ispitali svi

[12] Try QA. URL: http://tryqa.com/ (2021-02-27)

mogući slučajevi koji se u praksi mogu dogoditi, za svaki od mogućih događaja piše se poseban test slučaj.

Cilj ovog primjera je ispitati hoće li softver zabraniti prijavljivanje kada korisnik unese pogrešne podatke. U nastavku se nalazi tablica svih test slučajeva koji se mogu kreirati za testiranje Facebook prijave.

TC ID	TC Description	Pre-conditions	Test Steps	Test Data	Expected Results	Actual Results	Test Status	Created by	Date of creation
1	Testirati funkcionalnost prijave s točnim email-om i točnom lozinkom.	Otvoriti https://www.facebook.com/	1. Unesi korisničko ime. 2. Unesi lozinku. 3. Klikni "Prijavi se".	Email: kor12 Lozinka: loz123	Uspješna prijava	Uspješna prijava	✓	Tea Bizik	27.2.2021.
2	Testirati funkcionalnost prijave s točnim email-om i netočnom lozinkom.	Otvoriti https://www.facebook.com/	1. Unesi korisničko ime. 2. Unesi lozinku. 3. Klikni "Prijavi se".	Email: kor12 Lozinka: loz987	Neuspješna prijava	Neuspješna prijava	✓	Tea Bizik	27.2.2021.
3	Testirati funkcionalnost prijave s netočnim email-om i točnom lozinkom.	Otvoriti https://www.facebook.com/	1. Unesi korisničko ime. 2. Unesi lozinku. 3. Klikni "Prijavi se".	Email: kor85 Lozinka: loz123	Neuspješna prijava	Neuspješna prijava	✓	Tea Bizik	27.2.2021.
4	Testirati funkcionalnost prijave s netočnim email-om i netočnom lozinkom.	Otvoriti https://www.facebook.com/	1. Unesi korisničko ime. 2. Unesi lozinku. 3. Klikni "Prijavi se".	Email: kor85 Lozinka: loz987	Neuspješna prijava	Neuspješna prijava	✓	Tea Bizik	27.2.2021.
5	Testirati funkcionalnost prijave s točnim email-om i bez lozinke.	Otvoriti https://www.facebook.com/	1. Unesi korisničko ime. 2. Ne unosi lozinku. 3. Klikni "Prijavi se".	Email: kor12 Lozinka:	Neuspješna prijava	Neuspješna prijava	✓	Tea Bizik	27.2.2021.
6	Testirati funkcionalnost prijave bez email-a i točnom lozinkom.	Otvoriti https://www.facebook.com/	1. Ne unosi korisničko ime. 2. Unesi lozinku. 3. Klikni "Prijavi se".	Email: Lozinka: loz123	Neuspješna prijava	Neuspješna prijava	✓	Tea Bizik	27.2.2021.

7	Testirati funkcionalnost prijave bez email-a i bez lozinke.	Otvoriti https://www.facebook.com/	1. Ne unosi korisničko ime. 2. Ne unosi lozinku. 3. Klikni "Prijavi se".	Email: Lozinka:	Neuspješna prijava	Neuspješna prijava	✓	Tea Bizik	27.2.2021.

U tablici iz primjera uočava se da je svih sedam test slučaja provedeno uspješno.

Primjer 2. Testiranje Facebook registracije

Test scenarij za primjer registracije novih korisnika Facebook-a glasi: *Ispitati funkcionalnost registracije korisnika*. Ovaj je primjer složeniji od prethodnog jer se od korisnika očekuje da unese više podataka te je zato potrebno testirati više test slučajeva. Tablica u nastavku prikazat će neke od mogućih test slučajeva za scenarij registracije korisnika.

TC ID	TC Description	Pre-conditions	Test Steps	Test Data	Expected Results	Actual Results	Test Status	Created by	Date of creation
1	Testirati funkcionalnost registracije bez unošenja podataka.	Otvori stranicu za registraciju novih korisnika Facebook-a.	1. Ne unosi ime. 2. Ne unosi prezime. 3. Ne unosi email. 4. Ne unosi lozinku. 5. Ne unosi datum rođenja. 6. Ne unosi spol.	Ime: Prezime: Email: Lozinka: Datum rođenja: Spol:	Neuspješna registracija	Neuspješna registracija	✓	Tea Bizik	27.2.2021.
2	Testirati funkcionalnost registracije unošenjem imena, prezimena, email-a, lozinke i spola te unošenjem datuma rođenja za osobu mlađu od 13 godina.	Otvori stranicu za registraciju novih korisnika Facebook-a.	1.Unesi ime. 2. Unesi prezime. 3. Unesi email. 4. Unesi lozinku. 5. Unesi datum rođenja. 6. Odaberi spol.	1.Ime: Ana 2. Prezime: Anić 3. Email: anaanic987@gmail.com 4. Lozinka: loz123 5. Datum rođenja: 01.01.2012. 6. Spol: Žensko	Neuspješna registracija	Neuspješna registracija	✓	Tea Bizik	27.2.2021.

U tablici iz primjera uočava se da su oba test slučaja provedena uspješno, no isto tako vidljivo je da ima prostora za kreiranje još velikog broja test slučaja. Kada se test slučajevi napišu za izvršenje testa, rad testera je bolje organiziran i pojednostavljen.

4. Metode, vrste i razine testiranja

4.1. Metode testiranja

U svakodnevnoj praksi testiranja postoje tri osnovne metode koje se nazivaju: metoda crne kutije, metoda bijele kutije i metoda sive kutije. Ove se tri metode međusobno razlikuju u potrebnim razinama poznavanja softvera od strane testera, odakle i dolaze njihovi nazivi. Za testiranje se najčešće koristi pristup bijele kutije ili pristup crne kutije.[13]

Metoda crne kutije podrazumijeva da tester ne poznaje strukturu softvera niti kod koji pokreće softver, a vrlo često nije upoznat ni s korisničkim sučeljem što znači da mu nije poznat raspored naredbi i izgled web stranica. Cilj ove metode nije da obuhvati sve moguće situacije, već da tester stekne dojam o softveru sličnom onom koji će imati korisnik. Njegov zadatak je da ocijeni dopadljivost softvera, otkrije eventualne mane u rasporedu elemenata na web stranici i slično. Program se prilikom testiranja tretira kao crna kutija nepoznata sadržaja kod koje su vidljivi samo ulazi i izlazi, a funkcionalnost je određena promatranjem dobivenih izlaznih podataka na temelju odgovarajućih poznatih ulaza.[14]

Na primjeru koji se proteže kroz ovaj rad objasnit će se ova metoda. Zadatak je prijaviti se na svoj Facebook račun. Kada se otvori početna stranica Facebook-a, odjeljak za prijavu se ne nalazi na vidljivom mjestu te korisnik mora skrolati do kraja stranice da bi došao do dijela za prijavu, i tako svaki put. To nije baš ugodno korisničko iskustvo. Otkrivanje ovakvih nepravilnosti ne zahtijeva da tester poznaje kod niti da izvršava velik broj test slučajeva već je ovom metodom dovoljno da opiše što bi trebalo popraviti iz perspektive korisnika.

Metoda bijele kutije naziva se i unutarnjom strukturom softvera za izvođenje test slučajeva jer podrazumijevaju mogućnost gledanja u sustav za razliku od crnih kutija.[15] Tester je u slučaju metode bijele kutije zainteresiran za konkretan kod koji pokreće softver, a vrlo često ne ni cijeli kod već samo jedan njegov dio. On tada ispituje radi li taj jedan mali dio softvera na očekivani

[13] Martinović, Dragan. Testiranje programske podrške. URL: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=6512, str. 289. (2021-02-11)
[14] Isto.
[15] Graham, Dorothy…[et al.]. Foundations of Software Testing: ISTQB certification. URL: https://www.utcluj.ro/media/page_document/78/Foundations%20of%20software%20testing%20-%20ISTQB%20Certification.pdf, str. 87. (2021-02-12)

način, izvršavaju li se linije koda ispravnim redoslijedom i daju li rezultat koji se od njih očekuje. To je razlog zašto testeri moraju odlično poznavati kod kada koriste ovu metodu.[16] Osim testera, metodu bijele kutije koriste i programeri koji najbolje poznaju kod jer su autori istoga. Koriste ju kako bi ispitali radi li dio koda koji su upravo kreirali ispravno, a onda nastavljaju s pisanjem novih linija programskog koda.

Na primjeru, tester bez poznavanja unutarnjih struktura web mjesta, testira web stranice pomoću preglednika; pružanje ulaza (klikovi, pritisci tipki) i provjera rezultata prema očekivanom ishodu.

Metoda sive kutije kombinira dvije prethodne navedene metode pa tako tester ne mora poznavati cjelokupan kod, ali vrši testiranje primjenjujući detaljnije test slučajeve u odnosu na metodu bijele kutije. U testiranju crne kutije, ispitivač je nepoznat unutarnja struktura predmeta koji se testira, a u testiranju bijele kutije unutarnja struktura je poznata. U ispitivanju sive kutije, unutarnja je struktura djelomično poznata.[17]

Primjerice, testiranje metodom sive kutije, tester može započeti klikom na hyperlink kako bi provjerio otvara li novu stranicu. Tada tester provjerava upućuje li HTML kod na točan URL koristeći ispravnu sintaksu. Na kraju, tester ponovno provjerava korisničko sučelje kako bi potvrdio da preglednik preusmjerava na ispravan URL. Kada bi se tester koristio samo metodom bijele kutije, samo bi provjerio je li HTML pravilno kodiran i usmjerava li na točan URL koristeći ispravnu sintaksu. Testiranjem metodom crne kutije tester bi samo kliknuo na hyperlink i provjerilo bi se preusmjerava li preglednik na novi URL.

4.2. Vrste testiranja

Ovo će se poglavlje osvrnuti na neke od mnogobrojnih vrsta testiranja i dati primjere za svaki od njih. Svaki se tester redovno u praksi susreće s ovim testovima, stoga će se kroz praktične primjere prikazati kada se isti koriste. Testirati se može mrežno mjesto, mobilna aplikacija, desktop aplikacija i brojni drugi softveri, neke testovi se za neke softvere provode, dok drugi nisu

[16] Martinović, Dragan. Nav. dj., str. 290.
[17] Gray Box Testing, 2020. URL: http://softwaretestingfundamentals.com/gray-box-testing/ (2021-5-18)

primjenjivi. Unatoč tome, postoje i oni testovi koji se moraju provesti bez obzira koji softver je u pitanju, a to su test performanci, stress test i test sigurnosti

4.2.1. Test performansi

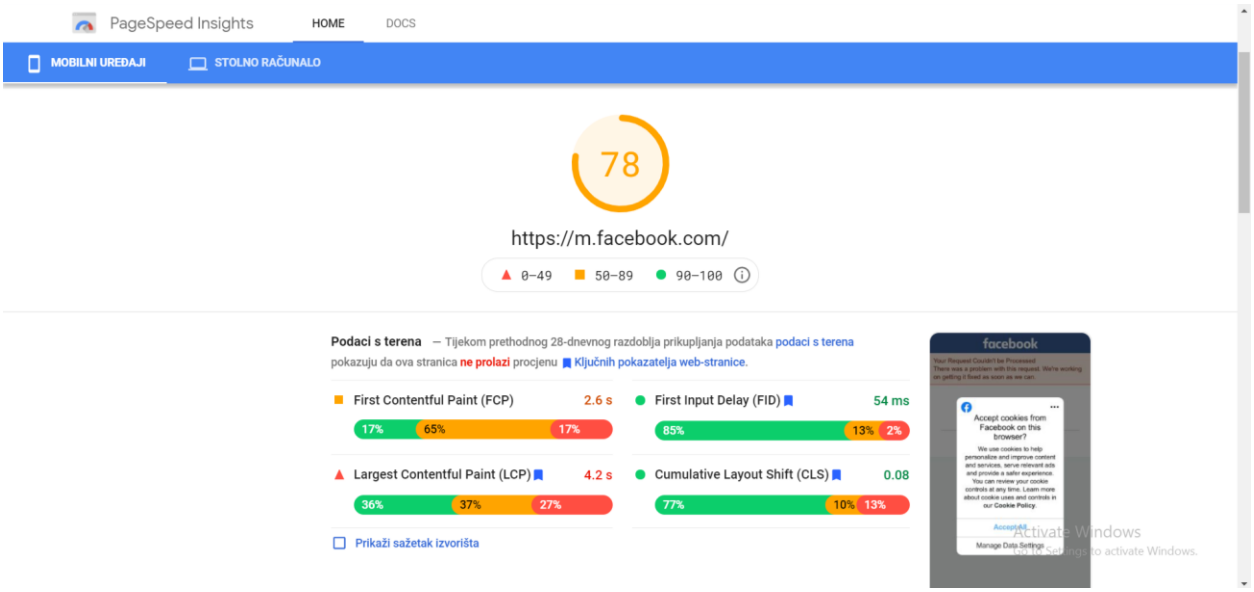
Test performansi provodi se kako bi se ispitalo kolika je brzina reakcije softvera, koliko se efikasno koriste resursi aplikacije te je li kapacitet softvera zadovoljavajuć. Prilikom ovog testiranja ne traže se sustavne pogreške, već se ispituje postoje li određene točke koje stvaraju zastoje. Ovaj se test također provodi kako bi se osiguralo poštivanje ciljeva i zahtjeva izvedbe te kako bi se pomoglo dionicima donositi informirane odluke povezane s ukupnom kvalitetom aplikacije koja se testira.[18]

Primjer koji se uzeti za ovo testiranje je ispitivanje kojom brzinom aplikacija za pametni telefon troši njegovu bateriju. Ako se primijeti da je aplikacija veliki potrošač baterije bilo bi dobro napraviti izmjene koje će smanjiti potrošnju.

Još jedan od klasičnih primjera testa performansi je brzina učitavanja web stranice. U slučaju da se web stranica vrlo sporo učitava treba preispitati zašto dolazi od ovakvog funkcioniranja web stranice. Jedan od mogućih razloga je da fotografije na web stranici zauzimaju veliki memorijski prostor, pa ih zato preglednik sporo učitava.

Danas je dostupan velik broj web alata koji vrše analizu performansi web mjesta. Jedan od njih je PageSpeed Insights[19] kreiran od strane Googlea. Služenje istim je vrlo jednostavno. Potrebno je samo u prazan prostor na početku stranice unijeti URL adresu stranice na kojoj želimo provesti test performanci i on će prikazati je li web mjesto optimizirano i u kojoj mjeri. U fokusu ovoga rada je Facebook mrežno mjesto, odnosno aplikacija, stoga će se za potrebe prikazivanja testa performance u prazan prostor unijeti URL iste te će se prikazati korisničko sučelje PageSpeed Insights mrežnog mjesta koji prikazuje je li mrežno mjesto koje smo analizirali optimizirano i u kojoj mjeri. (Slika 5.)

[18] Gregory, Janet; Crispin, Lisa. Nav. dj., str. 234.
[19] PageSpeed Insights. URL: https://developers.google.com/speed/pagespeed/insights/?hl=hr (2021-03-02)



Slika 5. Test performance Facebook mrežnog mjesta

Rezultati mjerenja i rezultat ocjene obojeni su prema sljedećim rasponima: 0-49 (crvena): loše, 50-89 (narančasta): potrebno je poboljšanje te 90-100 (zeleno): dobro. Da bi pružile dobro korisničko iskustvo, web stranice bi se trebale truditi da imaju dobar rezultat (90-100). "Savršeni" rezultat od 100 izuzetno je izazovno postići i ne očekuje se. Primjerice, za ocjenu od 99 do 100 potrebno je približno isto toliko poboljšanja metrike kao i za ocjenu 90 do 94.[20] Aplikacija Facebook u danome trenutku optimizirano je u mjeri 78 (narančasta), što znači da je potrebno poboljšanje.

Nadalje, iz rezultata provedenog testa performance koji se vidi na slici 5., može se iščitati da su pokazatelji procjene ovog mrežnog mjesta podatci s terena prikupljeni tijekom prethodnog 28-dnevnog razdoblja prikupljanja podataka. Podaci s terena pokazuju da ova stranica ne prolazi procjenu *Ključnih pokazatelja web-stranice*."[21] Oni se koriste kao dopunski mjerni podaci za temeljne web vitalne značajke kako bi pomogli u bilježenju većeg dijela iskustva ili dijagnosticiranju određenog problema. Trenutni set ključnih pokazatelja web-stranice za 2020. godinu usredotočen je na tri aspekta korisničkog iskustva - učitavanje, interaktivnost i vizualnu

[20] Lighthouse performance scoring: How Lighthouse calculates your overall Performance score, 2019. URL: https://web.dev/performance-scoring/ (2021-04-28)
[21] Ključni pokazatelji web stranice. URL: https://web.dev/vitals/ (2021-04-28)

stabilnost te uključuje mjerne podatke i njihove odgovarajuće pragove o kojima će se više govoriti u nastavku. (Slika 6.)

Slika 6. Ključni pokazatelji web stranice

First Contentful Paint (u nastavku FCP) je važna mjera usmjerena na korisnika za mjerenje uočene brzine učitavanja jer označava prvu točku na vremenskoj traci učitavanja stranice na kojoj korisnik može vidjeti bilo što na zaslonu. Brzi FCP pomaže uvjeriti korisnika da se nešto događa. FCP mjeri vrijeme od kada se stranica počinje učitavati do trenutka kada se bilo koji dio sadržaja stranice prikaže na zaslonu.[22] U trenutku testiranja Facebook aplikacije testom performance 17% učitavanja ove aplikacije ima dobar (<1 s) FCP, 65% ima (1 s ~ 3 s), što bi trebalo poboljšati, te 17% ima slabi (>3 s) FCP.

Largest Contentful Paint (u nastavku LCP) mjeri performanse učitavanja. Da bi se osiguralo dobro korisničko iskustvo, LCP bi se trebao pojaviti u roku od 2,5 sekunde od početka učitavanja stranice.[23] Brzi LCP pomaže uvjeriti korisnika da je stranica korisna. Promatrana je i testirana aplikacija Facebook. 36% učitavanja ove aplikacije ima dobar (<2.5 s) LCP, 37% ima (2.5 s ~ 4 s) LCP, što treba poboljšati, te 27% učitavanja ima slabi (>4 s) LCP.

First Input Delay (u nastavku FID) mjeri interaktivnost. Da bi se osiguralo dobro korisničko iskustvo, stranice trebaju imati FID od 100 milisekundi ili manje.[24] FID je kašnjenje između trenutka kada se klikne ili dodirne nešto poput URL adrese ili gumba i vremena u kojem preglednik reagira na navedenu akciju i započne je obrađivati. To znači da nizak FID pomaže da

[22] First Contentful Paint (FCP) URL: https://web.dev/fcp/ (2021-05-03)
[23] Walton, Philip. Web Vitals, 2020. URL: https://web.dev/vitals/ (2021-05-03)
[24] Walton, Philip. Nav. dj.(2021-05-03)

stranica bude upotrebljiva. Za Facebook testiranu aplikaciju 85% učitavanja ima dobar (<100 ms) FID, 13% ima (100 ms ~ 300 ms), što treba poboljšati, te samo 2% (>300 ms) ima slabi FID.

Cumulate Layout Shift (u nastavku CLS) mjeri vizualnu stabilnost. Da bi se osiguralo dobro korisničko iskustvo, stranice trebaju imati CLS od 0,1. ili manje, odnosno, nizak CLS osigurava da stranica bude divna.[25] Također, CLS važno je mjerilo jer pomaže kvantificirati koliko često korisnici doživljavaju neočekivane pomake izgleda. 77% učitavanja Facebook aplikacije ima dobar (<0.1) CLS, 10% ima (0.1 ~ 0.25), pto treba poboljšati, te 13% ima slabi (0.25) CLS.

Za potrebe provođenja testa performansa za Facebook aplikaciju u oom radu korišteno je web mjesto *PageSpeed Insight* kako bi se prikazale glavne značajke ovakvog testiranja, no internet je pun i drugih alata za provođenje iste vrste testova. Neki od najpoznatijih su *GTmetrix, Pingdom, Webpage Test, KeyCDN Site Speed Test* i mnogi drugi.

4.2.2. Stres test

Stres test koristimo kada želimo saznati do koje granice će softver istrpjeti istovremeni broj korisnika, odnosno stress test je onaj koji ide izvan uobičajene očekivane upotrebe sustava (da vidimo što će se dogoditi izvan očekivanja dizajna), s obzirom na opterećenje ili zapreminu.[26] Aktivni korisnici raznih softvera na internetu znaju reći: "Pao je sustav." To znači da je softver u danom trenutku bio preopterećen te nije mogao izvršiti toliki broj zahtjeva dobivenih od strane korisnika.

Stress test će se razmotriti kroz sljedeće primjere:

a) Internetska stranica za kupnju ulaznica za koncerte u najavi može svjedočiti velikom porastu prometa ili kad najavi prodaju.
b) Na vodećim internet portalima spominje se poznata osoba, a članka nudi i link koji vodi na profil društvene mreže te osobe. Promet profila automatski se povećava.

Kako bi se ovakve situacije spriječile, testeri vrše ispitivanje do koje granice će softver funkcionirati bez problema. Nužno je provesti stress test kako bi se prilagodili tako abnormalnim

[25] Isto.
[26] Graham, Dorothy…[et al.]. Nav. dj., str.183. (2021-05-03)

skokovima prometa. Nepridržavanje ovog naglog prometa može rezultirati gubitkom prihoda i ugleda.

Testiranje otpornosti na stres također je izuzetno važno kako bi se provjerilo radi li sustav u neobičnim uvjetima te prikazuje li se odgovarajuća poruka o pogrešci kada je sustav pod stresom. Također, kvar sustava u ekstremnim uvjetima može rezultirati ogromnim gubitkom prihoda te je zbog toga bolje biti pripremljen za iste provođenjem stress testa.

Glavna svrha testiranja otpornosti na stres je osigurati da se sustav oporavi nakon kvara, što se naziva obnovljivošću.[27]

4.2.3. Test sigurnosti

Test sigurnosti je jedan od najbitnijih testova jer osigurava sigurnost i zaštitu svih podataka te sprječava nepoželjne hakerske napade na softver. On primjerice istražuje funkcije (npr. vatrozid) koje se odnose na otkrivanje prijetnji, poput virusa, od zlonamjernih strana. Stoga testeri moraju biti vrlo kreativni u pronalaženju načina da istestiraju sigurnost željenog softvera.[28]

Značajke ili karakteristike alata za ispitivanje sigurnosti uključuju podršku za: prepoznavanje virusa, otkrivanje upada poput napada uskraćivanja usluge, simuliranje raznih vrsta vanjskih napada, ispitivanje otvorenih luka ili drugih vanjski vidljivih mjesta napada, prepoznavanje slabosti u datotekama lozinki i lozinkama te sigurnosne provjere tijekom rada, npr. za provjeru cjelovitosti datoteka i otkrivanje upada, npr. provjera rezultata test napada.[29]

Na primjeru Facebook mrežnog mjesta može se testirati sigurnost. Upisujući vlastitu registriranu email adresu prilikom prijave na Facebook te tri puta zaredom upisujući netočnu lozinku, Facebook na ekranu ispisuje obavijest ”Incorrect Password. The password you entered was not valid.” te nudi opciju za ponovno postavljanje lozinke putem email adrese. Nadalje, na Facebook-u se test sigurnosti očituje u slanju obavijesti korisniku putem email adrese ako je njegov račun prijavljen s druge IP adrese te ga pita “Jeste li to bili vi?”. Ovo su samo neki od primjeri kako su testeri Facebook-a osigurali svoje korisnike.

[27] Guru99. Nav. dj. (2021-05-17)
[28] Graham, Dorothy…[et al.]. Nav. dj., str. 50. (2021-5-17)
[29] Isto. Str.182. (2021-5-17)

Svrha testiranja sigurnosti je identifikacija i uklanjanje programskih pukotina koje potencijalno mogu voditi k povredi sigurnosti, kao i vrednovanje efektivnosti sigurnosnih mjera.[30]

Nažalost, primjeri loše osiguranih softvera postoje, iako se programeri i testeri trude svim snagama da oni budu maksimalno sigurni. Primjerice, neka osoba želi pristupiti tuđem bankovnom računu. Dobar softver mora biti napisan da spriječi sve moguće načine za otkrivanje i zlouporabu tuđih osobnih podataka.

4.3. Razine testiranja

Nakon metoda i najvažnijih vrsta testiranja obratit će se pozornost i na razine testiranja. Generičke vrste testiranja su: funkcionalno testiranje, nefunkcionalno testiranje, testiranje softverske strukture i testiranje povezano s promjenama i regresijsko testiranje.[31] Funkcionalno testiranje pokriva funkcionalnost sustava i što sustav radi, a ono će se oprimjeriti u primjerima podjele razina testiranja. Razine testiranja su korisne kako bi se razumjelo koliki je dio softvera obuhvaćen konkretnim testom. Postoje brojne razine testiranja softvera, no za potrebe ovoga rada, objasnit će se četiri najvažnije razine funkcionalnog testiranja, a to su: jedinično testiranje, integracijsko testiranje, sustavno testiranje te testiranje prihvatljivosti.

Jedinično testiranje, kako mu samo ime govori, usmjereno je na najmanji dio softvera – jednu njegovu jedinicu, tj. dio koda koji izvršava pojedinačnu funkciju, odnosno naredbu.[32] Iako se jediničnim testiranjem provjerava ispravnost jedinice sustava, ne postoji jedinstvena definicija jedinice. Ovisno o načinu implementiranja programskog koda jedinica može predstavljati funkciju, postupak, metodu, klasu, itd. Jednostavno rečeno, programska jedinica je komad koda koji se poziva izvan jedinice i koji može pozvati druge programske jedinice.[33] Glavni cilj ove razine testiranja jest otkrivanje pogreški na razini funkcija koje se koriste u softveru.

Kako bi se što bolje prepoznala važnost ove razine testiranja, koristit će se primjer prijave na korisnički račun Facebook-a. U kodu koji pokreće prijavljivanje na korisnički račun postoji

[30] Martinović, Dragan. Nav. dj., str. 291. (2021-05-18)
[31] Jukić, Ivana. Automatsko testiranje programa. Str.14. URL: https://zir.nsk.hr/islandora/object/foi:3888/preview (2021-5-17)
[32] L. Bradford. Everything You Need to Know About Software Testing Methods. URL: https://www.thebalancecareers.com/all-you-need-to-knowabout-software-testing-methods-4019921 (2021-5-18)
[33] S. Naik, P. Tripathy, Software testing and quality assurance: Theory and practice, New Jersey: John Wiley & Sons, Inc., 2008., str. 83.

funkcija koja provjerava ispravnost lozinke pojedinog korisnika. Ta funkcija uzima lozinku koju je korisnik unio i uspoređuje je s lozinkom iz baze povezanom s računom korisnika. U slučaju da ova funkcija ne funkcionira ispravno, to znači da postoji rizik da će netko s pogrešnom lozinkom moći pristupiti računu korisnika o kome se govori. Iz navedenog primjera zaključuje se da je ova razina testiranja vrlo bitna.

Govorilo se o razini testiranja koje je usmjereno na jednu jedinicu softvera, no nakon takvog testiranja, odnosno nakon testiranja više takvih jedinica, potrebno je testirati rade li takve jedinice zajedno. U tom slučaju govori se o integracijskom testiranju. Ono je slično jedinstvenom testiranju, ali sadrži višu razinu integracije. Ako se jediničnim testiranjem provjere sve jedinice i utvrdi se da one rade ispravno, ne znači da će to biti i slučaj kada se koriste zajedno. Primjerice, može se dogoditi da se jedna funkcija izvrši prije druge, iako ne bi trebalo.

Moduli softvera ispituju nedostatke kako bi se provjerila njihova funkcija. Testiranje integracije identificira pogreške prilikom integracije modula. Različite metode za integracijske testove uključuju "odozdo prema gore", "odozgo prema dolje" i "funkcionalni inkrementalni".[34]

U spomenutom primjeru prijave na korisnički račun Facebook-a može doći do takve situacije. Primjerice, postavlja se pitanje što bi se dogodilo kada bi korisnik unio email adresu, lozinku i kliknuo na gumb za prijavu, a da se prijavljivanje na račun pokrene prije provjere unesenih podataka, odnosno funkcija prijavljivanja se izvrši prije funkcije provjere u bazi podataka. To bi značilo da bi korisnik mogao unijeti i pogrešne ili tuđe podatke, a da mu bude omogućeno prijavljivanje, što je nedopustivo ponašanje softvera.

Nakon provjere svih podsustava jednog softvera, vrši se sustavno testiranje. To je posljednja provjera prije nego li se softver proslijedi na korisničko odobrenje. Kroz ovo testiranje potvrđuje se funkcionira li cjelokupan sustav na ispravan način i surađuje li dobro s okolinom. Testiranje sustava provode testeri, testirajući sustav iz perspektive korisnika. Zadatak testera je provjeriti da li sustav odgovara svim zahtjevima te da li se ponaša na željeni način. Osim toga, potrebno je testirati cijeli sustav jer neke funkcionalnosti sustava mogu ispravno raditi jedino ako su u potpunosti integrirane s ostatkom sustava.[35]

[34] L. Bradford. Nav.dj. (2021-5-18)
[35] Jukić, Ivana. Nav. dj., str.14. (2021-5-18)

Na primjer, tester završi testiranje završenog softvera, odnosno Facebook-a. Tada provjerava kako on surađuje s operacijskim sustavom. Može se primjerice dogoditi da on usporava rad drugih aplikacija u operacijskom sustavu. Također, ispituje se komunicira li dobro s hardverom i mrežnim mjestima na koja se korisnici mogu prijaviti ili registrirati putem Facebook-a, te prenosi li očitane podatke ispravno u bazu podataka.

Nakon provedenih testova na sve tri razine ostaje i potvrđivanje ispravnosti na posljednjoj, odnosno četvrtoj razini. Ta razina podrazumijeva testiranje prihvatljivosti od strane korisnika. Korisnik softvera može biti poznat i tada se govori o klijentu koji je naručio softver i unaprijed dao specifikacije koje od softvera očekuje. U tom slučaju testiranje prihvatljivosti obavljaju klijent i test tim zajedno, a cilj je provjeriti odgovara li softver zahtjevima klijenta.

Međutim, korisnik softvera može biti i nepoznat. To se događa u situaciji kada je softver napravljen na inicijativu IT tvrtke i ponuđen na korištenje putem internet preuzimanja. Općenito postoje dvije vrste ispitivanja prihvatljivosti. Tijekom alfa testiranja, softver se interno izvršava na web mjestu programera u simuliranom ili stvarnom okruženju. Softver radi kao da ga koriste krajnji korisnici. Programeri bilježe sve probleme i počinju ispravljati programske pogreške i druge probleme. Beta testiranje ili terensko testiranje omogućuje klijentima da testiraju proizvod na svojim web mjestima u stvarnim uvjetima. Klijenti mogu ponuditi grupi krajnjih korisnika priliku da testiraju softver putem izdavanja ili beta verzije. Beta testiranje želi dobiti stvarne povratne informacije korisnika koje se šalju programeru.[36]

U nastavku će se prikazati popis uobičajenih razlika između alfa testiranja i beta testiranja.[37]

[36] L. Bradford. Nav. dj. (2021-5-18)
[37] ReQtest: Alpha vs Beta Testing: How They are Different? URL: https://reqtest.com/testing-blog/alpha-vs-beta-testing/ (2021-5-18)

ALFA TESTIRANJE	BETA TESTIRANJE
Ovo testiranje uključuje i bijelu i crnu kutiju.	Ovo testiranje uključuje samo testiranje crne kutije.
Zaposlenici tvrtke za razvoj softvera u kojoj se softver izrađuje, provode alfa testiranje.	Vanjski korisnici koji nisu dio tvrtke za razvoj softvera provode beta testiranje.
Alfa testiranje odvija se na kraju programera.	Beta testiranje odvija se na kraju korisnika i s platformom po njihovom izboru.
Pouzdanost i sigurnost softvera nisu obuhvaćeni alfa testiranjem.	U beta testiranju provjeravaju se pouzdanost, sigurnost i robusnost softvera.
Alfa testiranje pomaže u testiranju kvalitete softvera prije nego što se prođe softver za beta testiranje.	Beta testiranje pomaže u procjeni softvera je li spreman za puštanje na tržište ili ne.
Za alfa testiranje potreban je ispitni laboratorij s postavljanjem testnog okruženja.	U beta testiranju ne postoji takav zahtjev za postavljanje testnog okruženja ili laboratorija.
Izvršni ciklus alfa testiranja može se proširiti na duže trajanje.	Beta testiranje usporedno zahtijeva manje vremena za izvršenje.
Programeri mogu brzo riješiti probleme ili pogreške pronađene tijekom alfa testiranja.	Problemi pronađeni u beta testiranju riješeni su u budućoj verziji softvera.

Alfa testiranje, kao i beta testiranje, važno je kako bi se osiguralo da konačna izdana verzija softvera ne ispunjava samo očekivanja tvrtke za razvoj softvera, već i očekivanja korisnika.

5. Korisnička priča

Korisnička priča kratki je opis funkcionalnosti ispričan iz perspektive korisnika koja je vrijedna bilo korisniku ili korisničkom timu. Priče se tradicionalno zapisuju na indeksne kartice. Kartica obično sadrži opis značajke u jednom retku. Na primjer, "Kao korisnik Facebook-a, mogu komentirati fotografije". Kartice se mogu koristiti samo u kombinaciji s naknadnim razgovorima između korisničkog tima i razvojnog tima i određenom provjerom da je priča implementirana kroz pisanje i pokretanje testova.[38]

Korisničke priče potiču odgađanje detalja dok se ne postigne najbolje razumijevanje o onome što stvarno treba. Budući da korisničke priče naglasak preusmjeravaju na razgovor i dalje od pisanja, važne odluke nisu zabilježene u dokumentima koji vjerojatno neće biti pročitani. Umjesto toga, važni aspekti o pričama bilježe se u automatiziranom testu prihvaćanja i često se pokreću. Pored toga, izbjegavaju se ne razumljivi pisani dokumenti s izjavama poput: Sustav mora pohraniti adresni i poslovni telefonski broj ili broj mobitela. Što to znači? To bi moglo značiti da sustav mora pohraniti jedan od ovih.

1. (Adresa i poslovni telefon) ili mobitel
2. Adresa i (poslovni ili mobitel)

Budući da su korisničke priče bez tehničkog žargona (korisnički tim ih piše), razumljivi su i programerima i korisničkom timu. Svaka korisnička priča predstavlja diskretni dio funkcionalnosti, tj., nešto što bi korisnik vjerojatno mogao učiniti u pojedinačnim postavkama. Takve korisničke priče se čine prikladnim kao alat za planiranje.[39]

Product Owner zajedno s timom obično piše korisničke priče i objašnjava korisničku priču razvojnom timu te razjašnjava sva pitanja koja tim može imati.

[38] Gregory, Janet; Crispin, Lisa. Nav. dj., str. 497. (2021-6-1)
[39] Cohn, Mike. User Stories Applied: For Agile Software Development. Str. 14. URL: https://books.google.hr/books?id=SvIwuX4SVigC&printsec=frontcover&dq=user+story&hl=hr&sa=X&redir_esc=y#v=onepage&q=user%20story&f=false (2021-6-1)

5.1. Product Manager

Product Manager je odgovoran za maksimiziranje vrijednosti proizvoda i rada razvojnog tima tvrtke. Način na koji se to radi može se uvelike razlikovati među organizacijama, Scrum timovima i pojedincima. Product Manager je jedina osoba odgovorna za upravljanje zaostatkom proizvoda. Upravljanje zaostatkom proizvoda[40] uključuje: jasno izražavanje stavki zaostalih proizvoda, naručivanje predmeta u zaostatku proizvoda za najbolje postizanje ciljeva i misija, optimiziranje vrijednosti posla koji razvojni tim obavlja, osiguravanje da je zaostatak proizvoda vidljiv, transparentan i jasan svima te pokazuje što će tim će raditi na sljedećem i osiguravanje da razvojni tim na razini razumije stavke u zaostatku proizvoda.

Product Manager može obaviti gore navedeni posao ili to treba učiniti razvojni tim. Međutim, Product Manager je taj koji ostaje odgovoran. Važno je naglasiti da je Product Manager jedna osoba, a ne radna skupina Da bi on uspio, cijela organizacija mora poštivati njegove odluke. One su vidljive u sadržaju i redoslijedu zaostalih proizvoda. Nedopušteno je reći razvojnom timu da radi iz različitih skupina zahtjeva, i razvojni tim ne smije djelovati prema onome što bilo tko drugi kaže.

Product Owner često se naziva "glasom kupca" na agilnim projektima ili "Product Manager" na tradicionalnim projektima, a osoba je odgovorna za priopćavanje zahtjeva kupaca. Iako se smatra dijelom korisničke jedinice, uloga vlasnika proizvoda presudna je za uspjeh napora agilnog razvoja proizvoda. Uz priopćavanje zahtjeva, vlasnik proizvoda odgovoran je za definiranje korisničkih kriterija za prihvaćanje zahtjeva, određivanje prioriteta i upravljanje zaostatkom proizvoda, kao i suradnju s razvojnom jedinicom tijekom iterativnog razvojnog ciklusa.[41]

[40] Zaostatak proizvoda jedan je od bitnih dijelova lanca razvoja proizvoda, prioritetni popis značajki proizvoda koji vodi od vizije tvrtke i proizvoda do izvršenja do punog izdanja. Moćan je alat jer pretvara viziju na visokoj razini u radne detalje stvaranja proizvoda.
[41] ISTQB Agile Tester. Str. 102.

5.2. QA tester

Osiguranje kvalitete (QA) i ispitivanje obično ne spadaju u radne odgovornosti Product Managera. Međutim, budući da je to tako presudan dio procesa razvoja softvera (a Product Manageri odgovorni su za svoj proizvod), nekoliko je bitnih stvari koje treba znati o ovom dijelu procesa.

Osiguranje kvalitete osigurava da softver ispunjava tehničke zahtjeve utvrđene u fazi planiranja sprinta. Općenito govoreći, to je trajna preventivna mjera koja se događa tijekom procesa razvoja softvera. Testiranje osigurava da nema pogrešaka koje bi mogle poremetiti upotrebljivost softvera. Općenito govoreći, ovo je korektivna mjera koja se događa na kraju svakog sprinta.

6. Alati i metodologije za provođenje agilne metode

U ovome će se poglavlju praktično predstaviti agilno testiranje. Postoji nekoliko agilnih metoda u praksi, a u sljedećih nekoliko odlomaka objasnit će se kako se primjenjuje ista Scrum i Kanban metodologijom u JIRA alatu za provođenje agilne metode.

6.1. JIRA

JIRA je obitelj proizvoda koja je stvorena kako bi pomogla svim vrstama timova da upravljaju svojim radom. Alat JIRA je kreiran od strane tvrtke Atlassian. JIRA nudi nekoliko proizvoda i mogućnosti implementacije koji su posebno namijenjeni softveru, IT-u, poslovanju, operativnim timovima i mnogim drugim.

Proizvodi i aplikacije izgrađeni na vrhu platforme JIRA pomažu timovima da planiraju, dodjeljuju, prate, izvještavaju i upravljaju radom. Platforma JIRA okuplja timove za sve, od agilnog razvoja softvera i korisničke podrške do novoosnovanih poduzeća.

Proizvodi izgrađeni na platformi JIRA su JIRA Software, JIRA Service Management i JIRA Work Management. JIRA Align je agilna platforma za poslovno planiranje koja povezuje rad u velikoj mjeri.

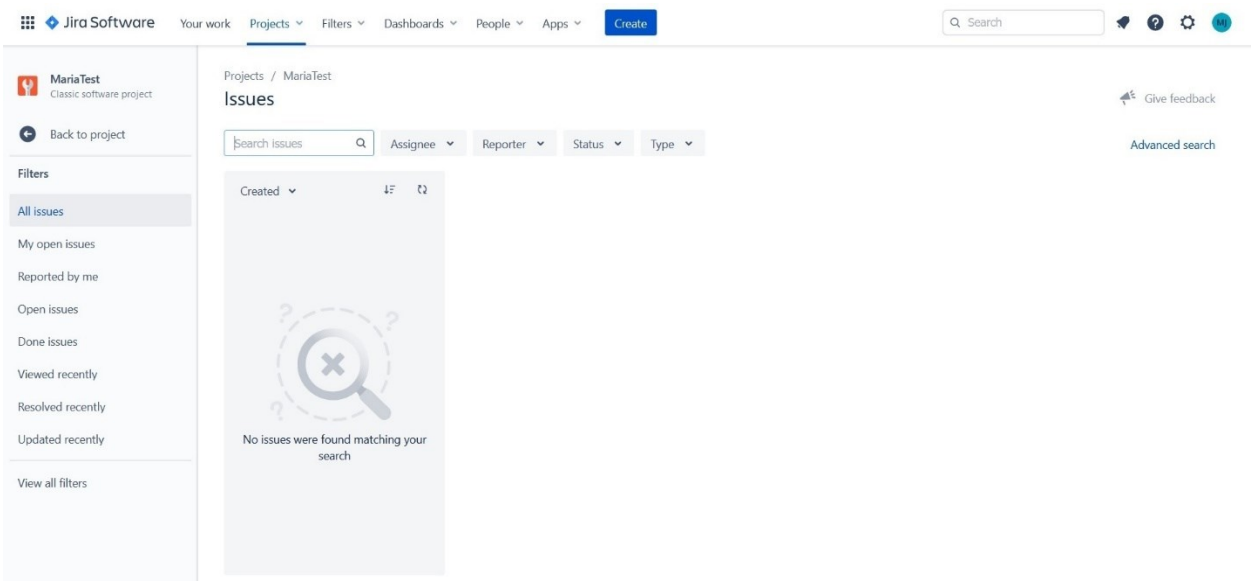
JIRA Software, JIRA Service Management i JIRA Work Management dolaze s ugrađenim predlošcima projekata za različite slučajeve upotrebe i besprijekorno se integriraju, tako da timovi iz različitih organizacija mogu zajedno raditi bolje i pametnije.[42]

Da bi se ovaj alat mogao koristiti potrebno je napraviti račun, a kroz to vodi sučelje na stranici Atlassian firme, gdje se odabire besplatna ili plaćena verziju softvera. Nakon odabira pojavljuje se prozor za registraciju. Registracija se provodi tako što se u polje “Claim your site” unosi naziv mrežnog mjesta koji se želi testirati. Ako za određeno mrežno mjesto račun već postoji, polje se oboja crvenom bojom te se mora odabrati neki drugi naziv.

Nakon postavki računa korisnika se upućuje na prozor za odabir template-a za rad. U tom dijelu izabire template za praćenje pogreški pod nazivom “Bug tracking”. Nakon odabira templatea

[42] A brief overview of Jira. URL: https://www.atlassian.com/software/jira/guides/getting-started/overview (2021-5-31)

“Bug tracking” otvara se prozor s obavijesti da se treba unijeti naziv projekta. Potom se dolazi do glavne kontrolne ploče JIRA-e, gdje se vidi da trenutno nema niti jednog pokrenutog “Issue” kako prikazuje slika 7.

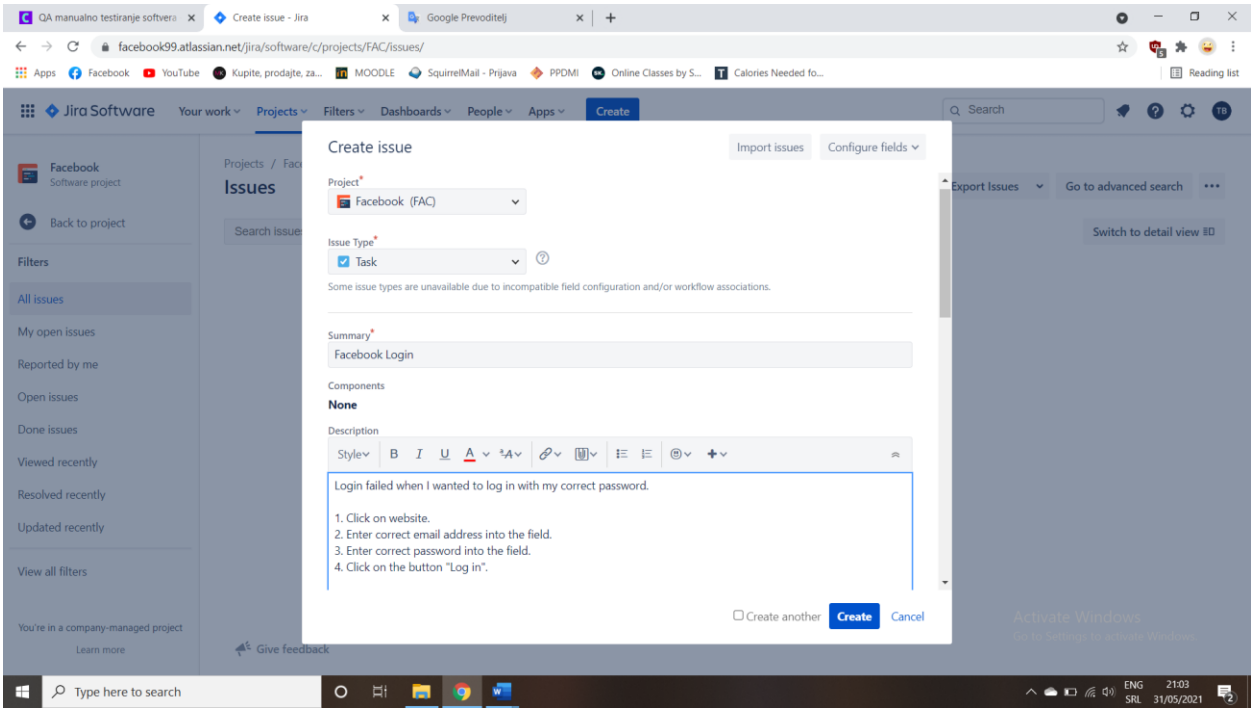


Slika 7. Kreiranje “Issue” u JIRA alatu

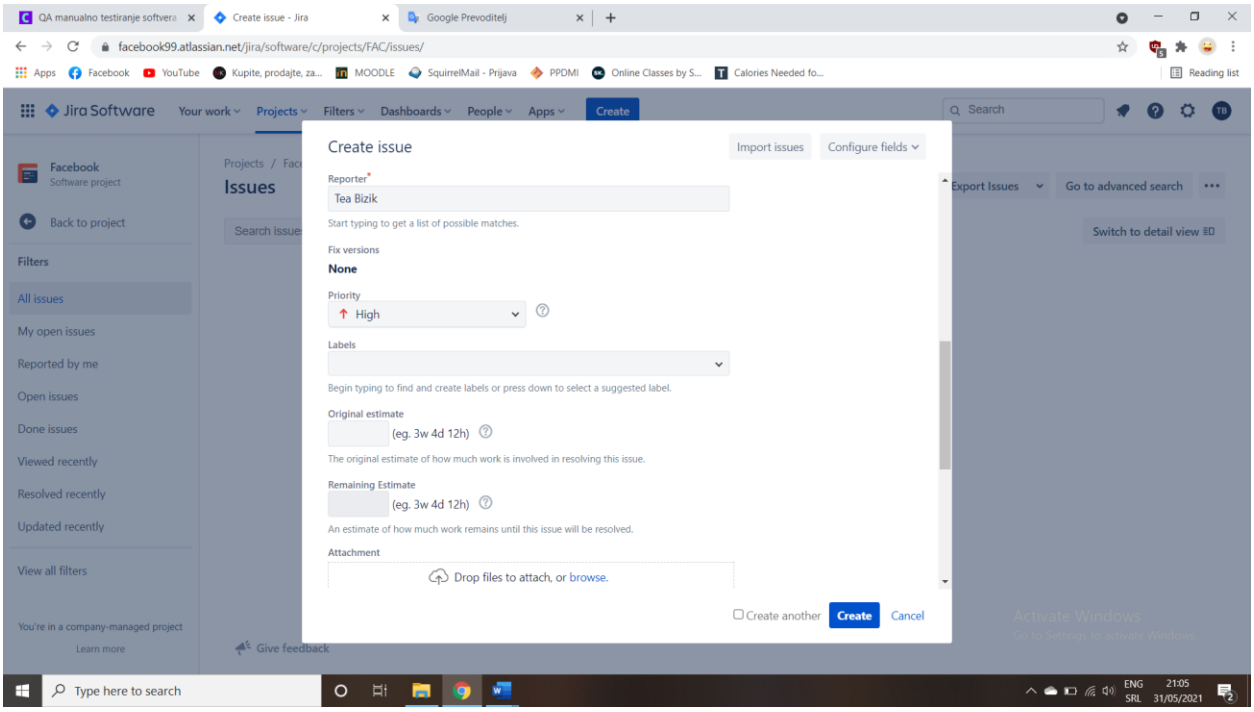
Kako bi se kreirao prvi “Issue” bira se ikona “+” s lijeve strane izbornika nakon čega se pojavljuje prozor za kreiranje novih, gdje se popunjavaju sljedeća polja koja su vrlo bitna:

1. Issue Type – odabire se “Bug”
2. Summary – dodaje se naslov pogreške
3. Description – opisuje se pogreška koja je pronađena
4. Priority – bira se prioritet rješavanja pogreške
5. Environment – opisuje se okolina u kojoj je pronađena pogreška (npr. operacijski sustav i preglednik)

Osim ovih mogu se popuniti i dodatna polja koja alat nudi. Nakon popunjavanja odabire se gumb “Create”. Na figurativnom primjeru pojasnit će se kako izgleda prijava pogreške u alatu JIRA na projektu Facebook. (Slika 8. i 9.)



Slika 8. Prijavljivanje bug-a u JIRA alatu



Slika 9. Prijavljivanje bug-a u JIRA alatu

Kada se kreira "Issue" kroz koji je prijavljena pogrešku u testiranom softveru, može se pratiti popis svih pogreški koje su prijavljene kroz opciju izbornika "Reported by me". Nakon ispravke može se ažurirati status pogreške i zatvoriti je kroz odabir statusa "Done".

Za timove koji se bave agilnim metodama, JIRA Software nudi Scrum i Kanban ploče. Ploče su čvorišta za upravljanje zadacima, gdje se zadaci preslikavaju u prilagodljive tijekove rada. Ploče pružaju transparentnost timskog rada i vidljivost statusa svakog radnog predmeta. Mogućnosti praćenja vremena i izvješća o izvedbi u stvarnom vremenu (grafikoni sagorijevanja / smanjenja, izvještaji o sprintu, tablice brzine) omogućuju timovima da pomno prate svoju produktivnost tijekom vremena.

6.2. Scrum

Scrum je prva metoda koja će se obraditi u okviru agilnih metoda. Iako je to samo jedna od metoda, mnogi Scrum poistovjećuju s agilnom metodom.

Scrum je inkrementalni[43] i iterativni[44] okvir za razvoj softvera i vjerojatno je jedna od najčešće korištenih agilnih metoda. Za razliku od ostalih agilnih metoda, Scrum nije kratica. Metodu su kao takvu osmislili Hirotaka Takeuchi i Ikujiro Nonaka te predstavili u članku Harvard Business Review-a, općeg menadžerskog časopisa koji izdaje Harvard Business Publishing, pod naslovom *Igra razvoja novih proizvoda* napisanom 1986. Riječ "Scrum" potječe iz sporta. U pitanju je ragbi, gdje se ova riječ koristi za situaciju kada jedan tim svoju snagu koristi da odgura protivnički tim od sebe. Svaki član jednog tima ima definiranu ulogu sa zadacima i može je koristiti kako pri napadu tako i u obrani. Uključeni postupak razvili su Ken Schwaber i dr. Jeff Sutherland.[45]

Zbog svojih brzih ponavljanja, Scrum je najprikladniji za timove čiji kupci i dionici žele biti aktivno uključeni redovitim viđanjem radnih proizvoda na izložbenim sastancima. Ovakva suradnja omogućava timu da napravi promjene za predstojeće vitrine. U usporedbi s ragbi timom, po Scrum metodi , u jednoj IT tvrtki također postoje definirane ključne uloge, a to su: vlasnik proizvoda (Product Owner), vođa tima (Scrum Master) te razvojni tim (programeri i testeri).

[43] Objašnjenje: uvećan, povišen.
[44] Objašnjenje: ponavljajući.
[45] ISTQB Agile Tester. Nav. dj., str. 107.

Cijeli proces provođenja Scrum metode sastoji se od više aktivnosti. Najprije se vrši kreiranje liste zadataka koje Scrum tim treba izvršiti. Na temelju nje definiraju se prioriteti koje treba odraditi u definiranom roku, tj. sprintu.[46]

Sprint je unaprijed definirano vremensko razdoblje i ponavljajući vremenski period u kojem se stvara radni softver, a često se naziva i ponavljanjem ili iteracijom. Najčešće korištena trajanja sprinta su razdoblja od dva, četiri i šest tjedana.[47]

Kako bi se bolje razumio prethodni opis, naglasit će se tri elementa Scruma koji se pojavljuju:

1) Lista zadataka koje treba obaviti kako bi proizvod bio isporučen klijentu (Product Backlog)[48]
2) Lista zadataka koju treba riješiti do kraja sprinta (Sprint Backlog)[49]
3) Rezultat koji treba nastati na kraju jednog sprinta (Increment)[50]

U tijeku svakog sprinta postoje bitni trenuci (event-i) koje je dobro znati, a to su: ažuriranje liste zadataka, sastanak za planiranje sprinta, dnevni Stand-Up sastanak, sastanak za osvrt na sprint.

Na kraju Sprinta se radi pregled Sprinta (Sprint review) s ciljem da se utvrdi napredak, demonstriraju svojstva kupcu, menadžeru, korisniku i vlasniku projekta te da se izvrši pregled projekta iz tehničke perspektive. Na ovom sastanku, kojeg vodi Scrum majstor, a vlasnik softvera i drugi zainteresirani mogu prisustvovati, demonstrira se posljednja verzija softvera. Dakle, prezentacija u smislu slajdova nije dozvoljena. Ciklus se ponavlja sa sastankom planiranja sprinta (Sprint Planning meeting) na kojem se opet odabiru svojstva koja će se implementirati u sljedećem sprintu.[51]

[46] Hundermark, Peter. Do better Scrum: An unofficial set of tips and insights into how to implement Scrum well.. Str. 15-16.
[47] ISTQB Agile Tester. Nav. dj., str. 109.
[48] Hundermark, Peter. Nav. dj., str. 15.
[49] Isto. Str. 18.
[50] Isto. Str. 26.
[51] Stefanović, Kristina; Tošić, Goran. Nav. dj., str. 13. (2021-05-31)

Scrum je besplatan alat za provođenje agilne metode. Scrum-ove uloge, artefakti, događaji i pravila su nepromjenjivi i iako je moguća implementacija samo dijelova Scruma, rezultat nije Scrum. Scrum postoji samo u cjelini i funkcionira dobro kao spremnik za druge tehnike, metode i prakse.[52]

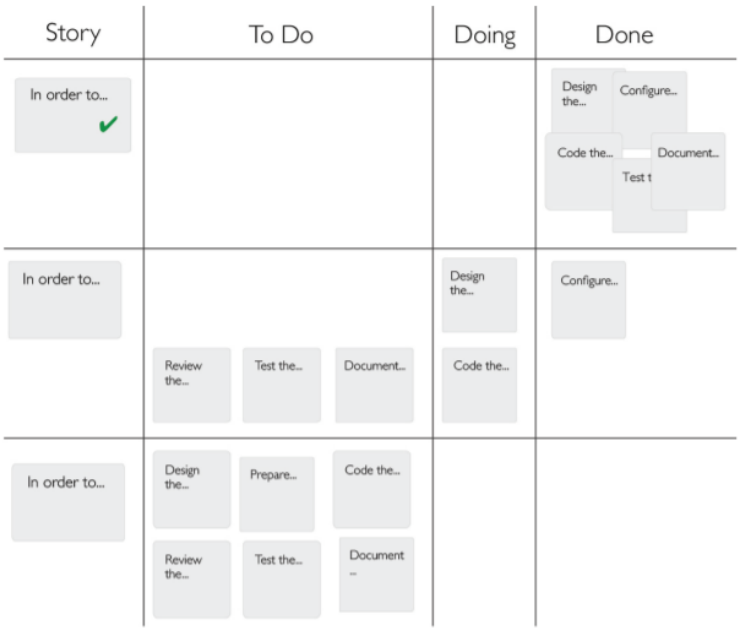
6.2. Kanban

Većina razvojnih timova prikazat će svoj Sprint Backlog na ploči zadataka, što je fizički prikaz popisa posla koji je dogovoren tijekom trenutnog Sprinta. Ploča zadataka primjer je iz Kanban-a, vrlo popularnog alata za primjenu agilne tehnologije. Najprije će se saznati što on predstavlja, a potom će se kroz primjer prikazati kako se on doista koristi u praksi.

Riječ "Kanban" potječe iz Japana i znači ploča s obavijestima. Termin se počeo primjenjivati najprije u auto industriji u tvrtki Toyota kako bi opisao ideju za upravljanje kretanjem materijala u proizvodnji. Kasnije se on počeo koristiti i u svijetu programiranja gdje ima nešto drugačiju ulogu. Može se reći da korištenje Kanban-a utječe na produktivnost uz minimalan napor jer je usmjeren na kretanje te se u svakom trenutku može kontrolirati i optimizirati kako bi se projekt na kojemu se radi u danom trenutku, nastavio kretati u pozitivnom smjeru.

Kao dio agilne metode, on se oslanja na komunikaciju u realnom vremenu na temu kapaciteta i preostalih zadataka. Zadaci se predstavljaju na kanban ploči putem kanban kartica i omogućavaju da članovi tima vide trenutni status zadataka prema redoslijedu i popisu koji prikazuje Slika 10.

[52] The Scrum Guide™ The Definitive Guide to Scrum: The Rules of the Game. Str. 16.



Slika 10. Kanban

Prema priloženome, uočava se da kod Kanban-a postoje tri stanja na ploči, a to su zadatci koji se trebaju riješiti (*To do*), zadatci u tijeku (*Doing*) i završeni zadatci (*Done*). Kako bi pratili zadatke od početka do završetka, tim definira navedena stanja koje svaki zadatak može imati. Ujedno, to su i nazivi tri elementa kanban ploče.

Nakon grupiranja na navedene dijelove kako je prikazano, zadatak se postavlja na Kanban ploču u vidu Kanban kartice, koja na sebi ima nekoliko informacija. Ovisno o potrebama i organizaciji u timu, ona može prikazivati više ili manje podataka, a najčešće sadrži: opis zadatka koji treba riješiti, razinu prioriteta zadatka, rok za završetak zadatka te ime i prezime odgovorne osobe koja treba zadatak završiti. Kanban kartica sadrži informacije o opisu zadatka (Task), roku za završetak zadatka (Duration) i ime odgovorne osobe (Tester).

Kanban kartica #1
Zadatak: Testirati prijavu na Facebook **Trajanje:** 1 sat **Tester:** Tea Bizik

Kada popune sve kartice, testeri ih postavljaju na Kanban ploču koja može izgledati slično ovoj s kreiranog vizualnog prikaza u nastavku. Uočava se da se sve do sada kreirane kartice nalaze u dijelu "To-do" – napraviti, što znači da sprint još nije započeo.

TO DO			DOING			DONE		
Kanban kartica #1 Zadatak: **Testirati prijavu na Facebook** Trajanje: **1 sat** Tester: **Tea Bizik**	Kanban kartica #2 Zadatak: **Testirati registraciju na Facebook** Trajanje: **1 sat** Tester: **Pero Perić**	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9

Kako bi se bolje razumjelo kako funkcionira princip korištenja Kanban kartica i Kanban ploče učinit će se to kroz primjer. Na prethodnoj slici vidimo da su sve kartice u dijelu "To-do"- „napraviti". Ima ih dvije, što znači da je vođa tima dodijelio dva zadatka svojim članovima. U timu su dvije osobe, odnosno Tea Bizik i Pero Perić. Tea ima jedan zadatak i on je na crvenoj kartici, dok Pero ima isto jedan zadatak, ali na zelenoj kartici. Zadaci su vremenski podjednako raspodijeljeni između njih dvoje, pa tako Tea i Pero imaju po jedan zadatak za koji je potreban 1 sat kako bi se završio.

Njihov tim lider dao im je ove zadatke i obavijestio ih da će imati sastanak za sata vremena kada će popričati o tome kakav je status njihovih zadataka.

Tea i Pero su počeli sa svojim zadacima. Kartice za zadatke koje trenutno obavljaju prebacili su u dio "Doing" – „izvođenje", a kada završe zadatak, prebacuju bi ga u dio ploče pod nazivom "Done" – „završeno". Nakon sat vremena, sastanak je počeo, a njihova kanban ploča je izgledala ovako:

TO DO			DOING			DONE		
1	2	3	1	Kanban kartica #2 Zadatak: **Testirati registraciju na Facebook** Trajanje: **1 sat** Tester: **Pero Perić**	3	Kanban kartica #1 Zadatak: **Testirati prijavu na Facebook** Trajanje: **1 sat** Tester: **Tea Bizik**	2	3
4	5	6	4	5	6	4	5	6

7	8	9	7	8	9	7	8	9

Sa slike Kanban ploče nakon sat vremena rada možemo vidjeti da je Ana završila jedan zadatak za koji je bio potreban jedan sat. Kartica za taj zadatak nalazi se u dijelu "Done" – „završeno" . Pero ima drugačiju situaciju. Njemu nije bilo dovoljno sat vremena za njegov zadatak pa je njegov zadatak u dijelu "Doing" – „izvođenje". Na ovom primjeru vidimo kako se Kanban ploča i kartice uspješno koriste kako bi ispratile napredak rada tima.

Promjena u Kanbanovoj metodi je evolucijska. To Kanban može učiniti boljim odabirom za organizacije čija kultura otežava uvođenje brzih promjena. Scrum, za razliku od Kanban-a, napada složenost frontalno i zahtijeva radikalniju promjenu stava. Scrum izaziva temeljne promjene koje su teške za organizacije i ljude, ali rezultati mogu biti zapanjujući.[53]

[53] Hundermark, Peter. Nav. dj.,str. 60.

7. Primjer primjene tehnika testiranja

7.1. Cilj i svrha istraživanja

Cilj praktičnog dijela rada je prikazati primjenu tehnika testiranja na primjeru Facebook aplikacije. Testira se usklađenost s pravilima Facebooka. Postoji više tehnika, a za potrebe ovog testiranja koristit će se sljedeće tehnike: testiranje klasa ekvivalencije, testiranje graničnih vrijednosti, testiranje na temelju tablice odlučivanja te testiranje promjene stanja.

Pravilo 1: Facebook ne dopušta otvaranje profila osobama mlađim od 13 godina.

Pravilo 2: Facebook dopušta otvaranje profila osobama s 14 godina.

Pravilo 3: Prijava korisnika na Facebook nije moguća ako se u obrazac za prijavu ne unesu oba parametra - email adresa i zaporka.

Pravilo 4: Facebook nakon trećeg neuspješnog pokušaja prijave na korisnički račun zabranjuje pristup iz sigurnosnih razloga.

Hipoteza je da se predloženim metodama može testirati usklađenost s pravilima.

7.2. Metode

7.2.1. Testiranje klasa ekvivalencije

Testiranje klasa ekvivalencije tehnika je koja pomaže testerima tako što omogućuje da se uvjeti koji određuju testiranje rasporede u skupine koje se nazivaju klase ekvivalencije. Tako testeri ne moraju testirati svaku pojedinačnu ulaznu vrijednost, već mogu odabrati po jednu vrijednost iz svake klase ekvivalencije, a da pokrivenost testa i dalje bude potpuna.

Facebook nakon unosa datuma i godine rođenja prilikom registracije vraća informaciju je li osobi dopušteno kreirati korisnički račun ili nije, generirajući tu informaciju na osnovu definiranih kriterija. U ovom slučaju postoje dvije klase ekvivalencije:

1. Nedopuštena dob za kreiranje Facebook računa: 1-12 godina
2. Dopuštena dob za kreiranje Facebook računa: 13- 99+ godina

7.2.2. Testiranje graničnih vrijednosti

Tehnika testiranja graničnih vrijednosti ne samo što je vrlo slična tehnici testiranja klasa ekvivalencije, već se i dosta oslanja na nju. Ona također koristi podjelu na klase, s tim što definira koje vrijednosti u okviru klasa treba testirati.

Služeći se primjerom iz prethodne metode, definirat će se granične vrijednosti. U prethodnoj metodi definirane su dvije skupine podataka: grupu podataka cijelih brojeva u rasponu 1-12 i grupu podataka cijelih brojeva od 13-99+.

Navedene vrijednosti 1, 12, 13 i 99 smatraju se graničnim vrijednostima, jer se nalaze na granicama ove dvije klase ekvivalencije. U ovom primjeru vrijednosti koje su ujedno i predmet testiranja su: 0, 1, 2, 11, 12, 13, 14, 98, 99 i 100.

7.2.3. Testiranje na temelju tablice odlučivanja

Tablica odlučivanja koristi se u slučajevima kada postoji puno kombinacija ulaznih parametara koje se trebaju istestirati. Prilikom prijave na Facebook račun korisnici ispred sebe vide sučelje za unos email adrese i zaporke. Prijava je uspješna jedino ako su oba parametra unesena, a u ostalim slučajevima prijavljivanje će biti neuspješno i ispraćeno obavijestima da je obavezno popuniti oba polja. U takvom okruženju s dva ulazna parametra moguće je kreirati četiri kombinacije.

Ulazi	Kombinacija 1	Kombinacija 2	Kombinacija 3	Kombinacija 4
Email adresa	Da	Da	Ne	Ne
Zaporka	Da	Ne	Da	Ne

7.2.4. Testiranje promjene stanja

Tehnika testiranja promjene stanja podrazumijeva ispitivanje reakcije softvera na prosljeđivanje točnih i netočnih ulaznih vrijednosti. Ako Facebook zaključa svoj nalog zbog prevelikog broja neuspješnih pokušaja prijave, znači da radi baš kako je isplanirano zbog sigurnosnih razloga.

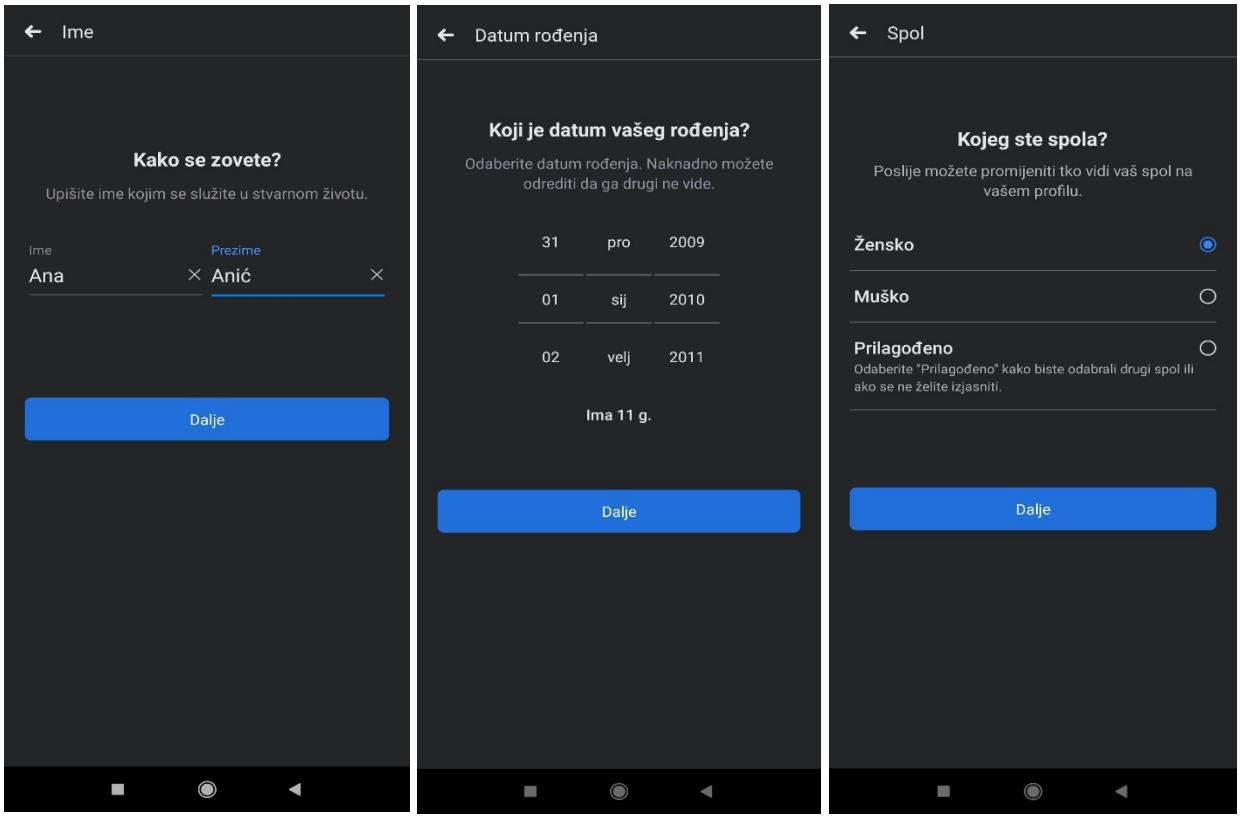
7.3. Rezultati

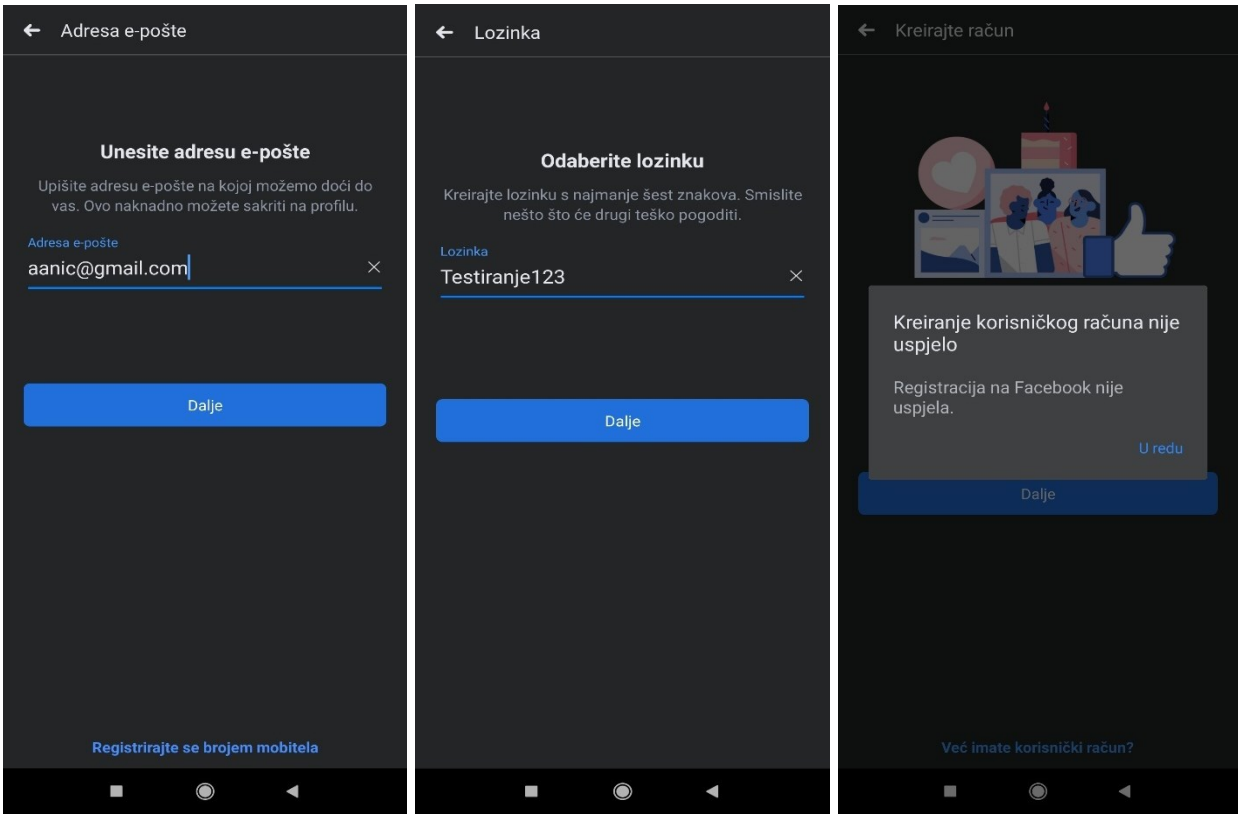
Za testiranje klasa ekvivalencije kreirane su dvije klase:

1) Nedopuštena dob za kreiranje Facebook računa: 1-12 godina
2) Dopuštena dob za kreiranje Facebook računa: 13- 99+ godina

Prvo se testira klasa nedopuštene dobi. U obrazac za registraciju Facebook računa najprije se unosi ime i prezime.Tekuća godina prilikom istraživanja je 2021. stoga se za unos datuma rođenja unosi 1. sij 2010. što označava da osoba koja pokušava registrirati Facebook račun trenutno ima 11 godina. Nakon unosa datuma rođenja, odabire se spol te unosi broj mobitela ili email adresa. (Slika 11.-16.) Podatci korišteni za ovu metodu su:

Ime i prezime: Ana Anić
Datum rođenja: 1. sij. 2010.
Spol: Žensko
Adresa e-pošte: aanic@gmail.com
Lozinka: Testiranje123



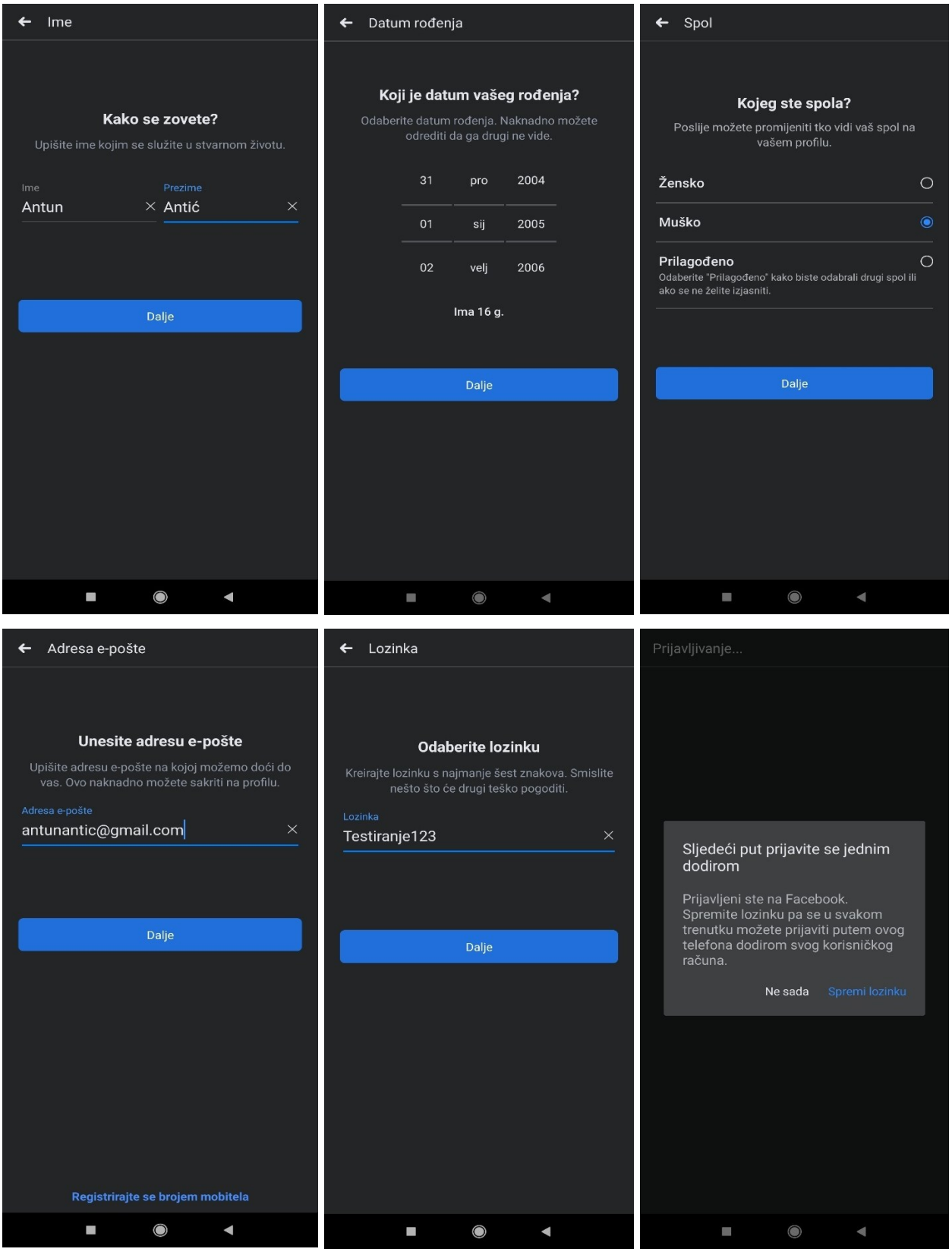


Slika 11.-16. Testiranje prve klase ekvivalencije na Facebook mobilnoj aplikaciji

Nakon ispunjenih podataka za registraciju Facebook računa od strane osobe nedopuštene dobi za kreiranje istoga, odnosno mlađe od 13 godina, registracija je neuspješna.

Zatim se testira dopuštena dob za kreiranje Facebook računa, odnosno registracija za osobe s napunjenih 13 i više godina. Obrazac za popunjavanje je isti kao na prethodnom primjeru, no prilikom testiranja, mijenja se samo parametar datum rođenja. Za ovo testiranje, uzima se primjer osobe s navršenih 16 godina. (Slika 17.-22.) Podatci korišteni za ovu metodu su:

Ime i prezime: Antun Antić
Datum rođenja: 1. sij. 2005.
Spol: Muško
Adresa e-pošte: antunantic@gmail.com
Lozinka: Testiranje123

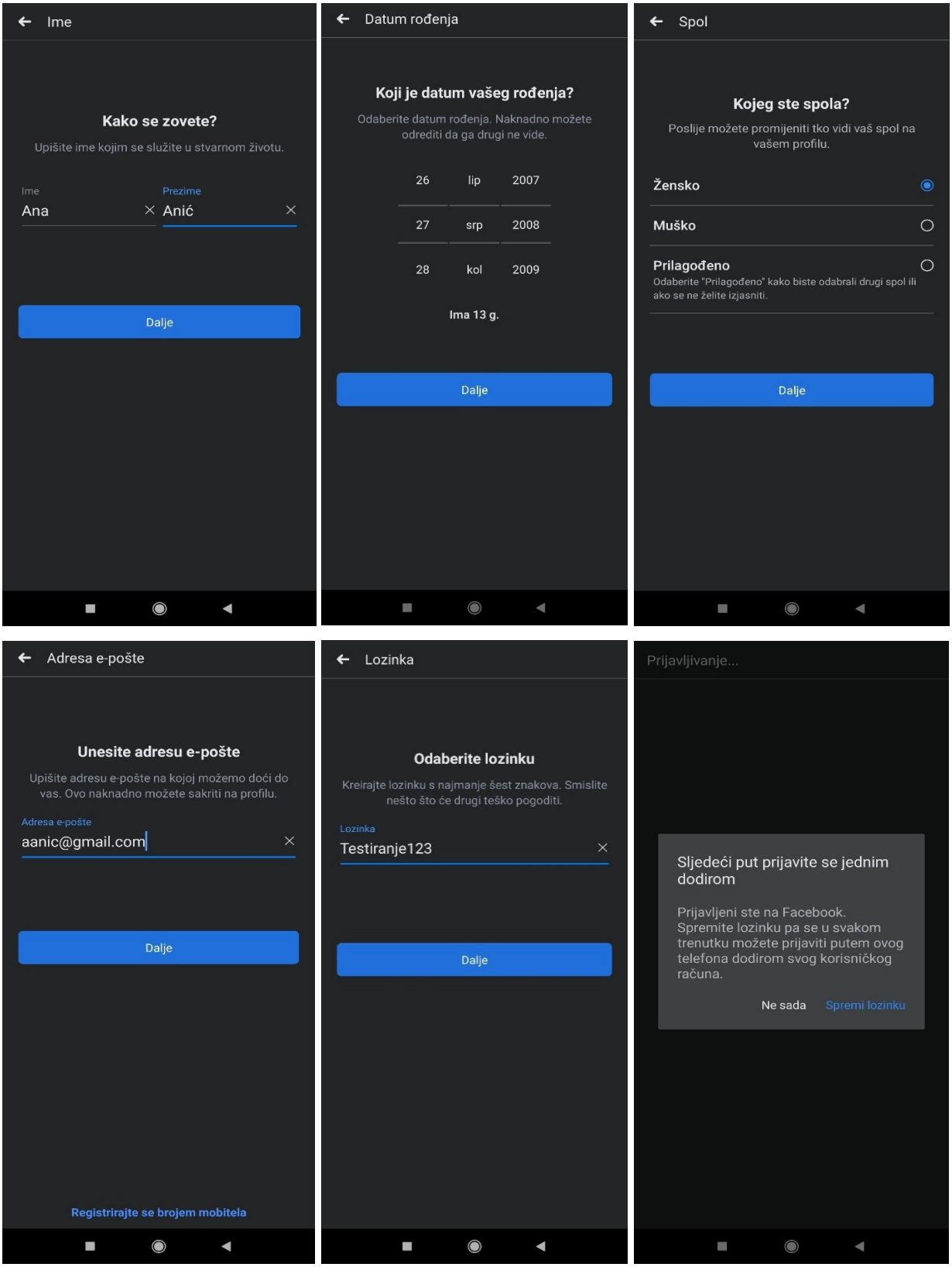


Slika 17.-22. Testiranje druge klase ekvivalencije na Facebook mobilnoj aplikaciji

Nakon ispunjenih podataka za registraciju Facebook računa od strane osobe dopuštene dobi za kreiranje istoga, odnosno starije od 13 godina, registracija je uspješna.

Za potrebe testiranja graničnih vrijednosti, testirat će se primjer za registraciju računa za osobu koja je na dan registracije Facebook računa napunila 13 godina. (Slika 23.-28.) Podatci korišteni za ovu metodu su:

Ime i prezime: Ana Anić
Datum rođenja: 27. srp. 2008.
Spol: Žensko
Adresa e-pošte: aanic@gmail.com
Lozinka: Testiranje123

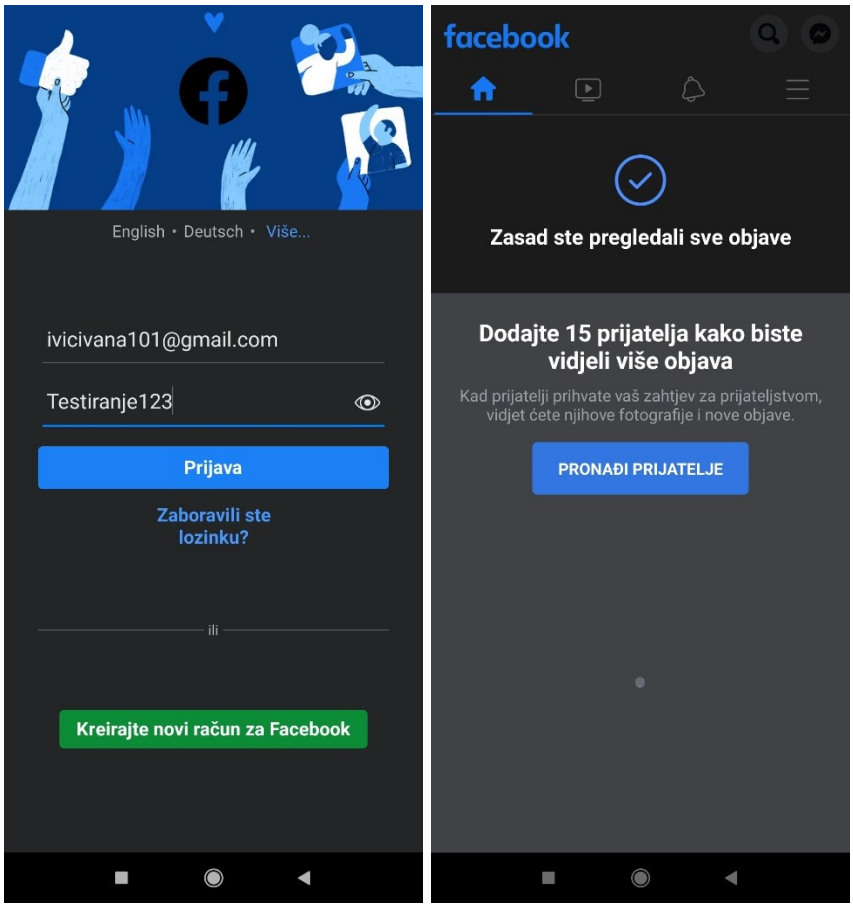


Slika 23.-28. Testiranje graničnih vrijednosti na Facebook mobilnoj aplikaciji

Nakon ispunjenih podataka za registraciju Facebook računa od strane osobe dopuštene dobi za kreiranje istoga, odnosno s navršenih točno 13 godina, registracija je uspješna.

Za testiranje na temelju tablice odlučivanja koriste se sve četiri moguće kombinacije iz tablice odlučivanja, a u fokusu je testiranje prijave na Facebook račun. Podatci korišteni za prvu kombinaciju (Slika 29. i 30.), odnosno točnu email adresu i točnu lozinku su:

Email adresa: ivicivana101@gmail.com
Lozinka: Testiranje123



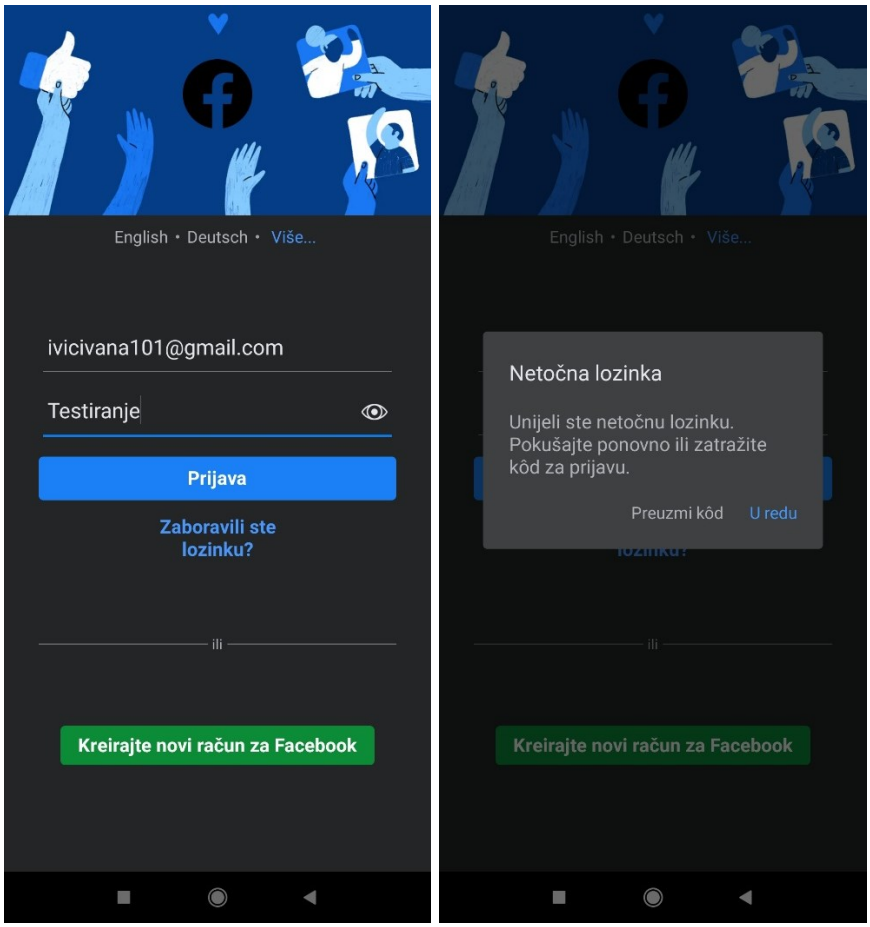
Slika 29. i 30. Testiranje na temelju tablice odlučivanja na Facebook mobilnoj aplikaciji

Korištenjem navedenih podataka prve kombinacije, prijava na Facebook račun je uspješna.

Podatci korišteni za drugu kombinaciju (Slika 31. i 32.), odnosno točnu email adresu i netočnu lozinku su:

Email adresa: ivicivana101@gmail.com

Lozinka: Testiranje



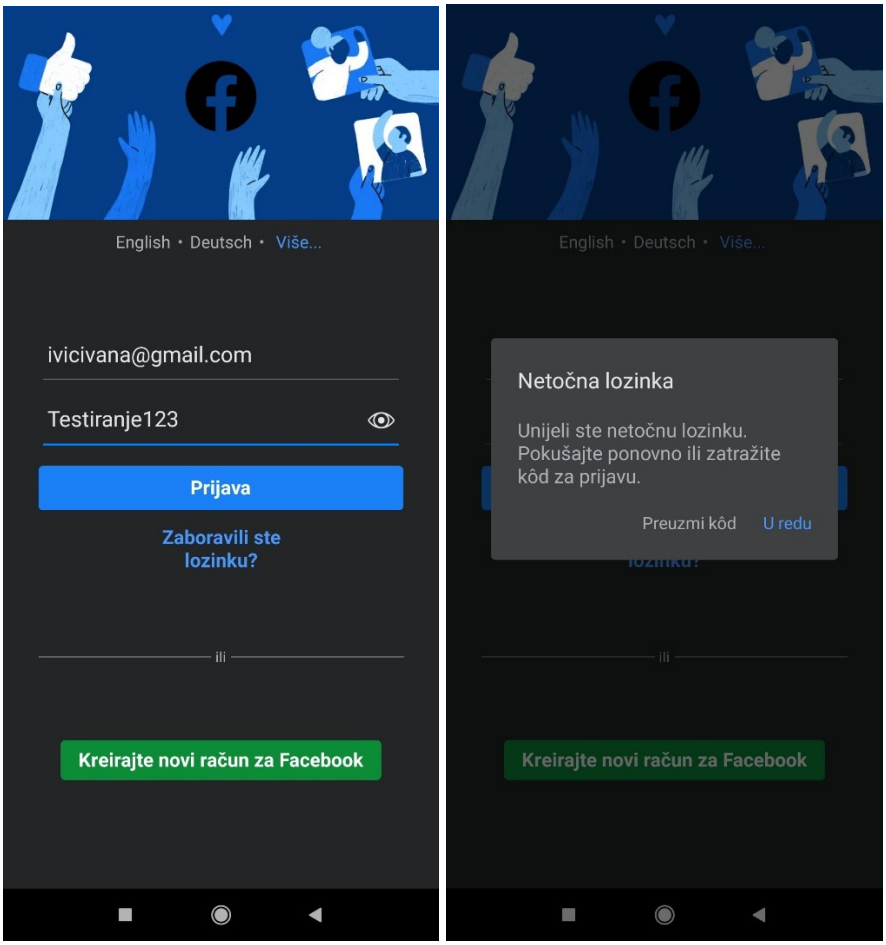
Slika 31. i 32. Testiranje na temelju tablice odlučivanja na Facebook mobilnoj aplikaciji

Korištenjem navedenih podataka druge kombinacije, prijava na Facebook račun je neuspješna.

Podatci korišteni za treću kombinaciju (Slika 33. i 34.), odnosno netočnu email adresu i točnu lozinku su:

Email adresa: ivicivana101@gmail.com

Lozinka: Testiranje123

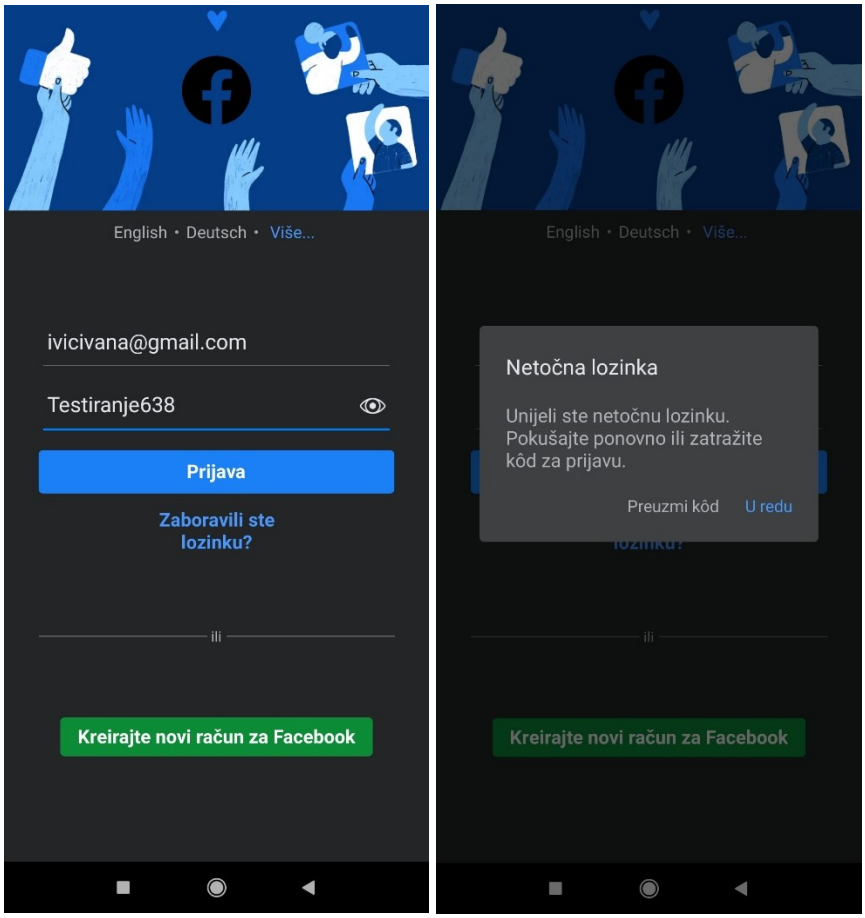


Slika 33. i 34. Testiranje na temelju tablice odlučivanja na Facebook mobilnoj aplikaciji

Korištenjem navedenih podataka treće kombinacije, prijava na Facebook račun je neuspješna.

Podatci korišteni za četvrtu kombinaciju (Slika 35. i 36.), odnosno netočnu email adresu i netočnu lozinku su:

Email adresa: ivicivana@gmail.com
Lozinka: Testiranje638



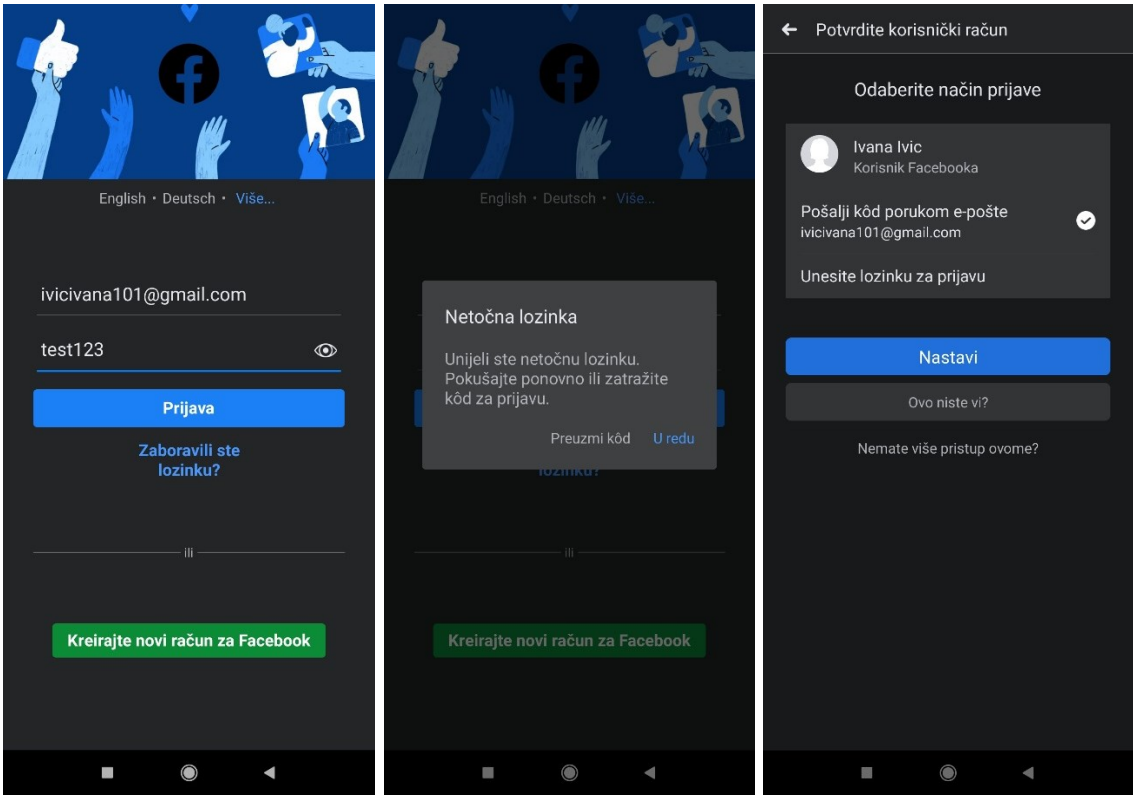
Slika 35. i 36. Testiranje na temelju tablice odlučivanja na Facebook mobilnoj aplikaciji

Korištenjem navedenih podataka četvrte kombinacije, prijava na Facebook račun je neuspješna.

Prilikom prijave na Facebook račun provodi se i testiranje promjene stanja. (Slika 37.-39.) Podatci korišteni za ovu metodu su:

Email adresa: ivicivana101@gmail.com
Lozinka: test123

Korištena je ispravna email adresa te netočna lozinka. Postupak prijave s navedenim parametrima ponavlja se tri puta.



Slika 37.-39. Testiranje promjene stanja na Facebook mobilnoj aplikaciji

Nakon trećeg pokušaja prijave na Facebook račun s netočnim podatcima za prijavu, Facebook zaključava svoj nalog zbog prevelikog broja neuspješnih pokušaja prijave što znači da radi baš kako je isplanirano zbog sigurnosnih razloga.

8. Zaključak

Testiranje i osiguranje kvalitete u svijetu informacijske tehnologije danas imaju veliku važnost. Svaki bi IT sektor trebao imao odjel za tehnički obrazovane testere i iskusne softverske profesionalce kako bi pravovremeno razgovarali o preliminarnim fazama razvoja bilo koje mobilne ili web aplikacije. Da bi testeri osigurali kvalitetu te temeljito testirali nove digitalne proizvode i osigurali da nemaju grešaka, uklonili loše performanse i riješili probleme sa sučeljem, oni izvode razna ispitivanja u različitim fazama životnog ciklusa softvera. Neki od primjeri takvih testova predstavljenih kroz rad su test performance, stres test i test sigurnosti. Budući da je testiranje softvera toliko kritično za kvalitetu i iskoristivost konačnog proizvoda, testeri se obično dovode u ranim fazama, poput planiranja i dizajna. Testiranje proizvoda nije brz i jednostavan, već dug i temeljit proces zbog čega se testeri koriste brojnim metodama, vrstama i razinama testiranja. Svaki tester ima svoj način pristupa rješavanju problema kao i osobni odabir programa u kojemu će komunicirati s ostatkom razvojnog tima.

Prilikom usporedbe Agilne i Waterfall metode testiranja, za potrebe ovog rada odabrana je agilna metoda zbog svoje fleksibilnosti koja dopušta promjene u zahtjevima za razvoj projekta. Agilnost uključuje vještine suočavanja s promjenjivim promjenama, brzog kretanja i ne posustajanja pred nepoznatom dinamikom tima. U praktičnom dijelu rada provedeno je testiranje usklađenosti pravila korištenja Facebook aplikacije. Koristeći se metodama poput testiranje klasa ekvivalencije, testiranje graničnih vrijednosti, testiranje na temelju tablice odlučivanja te testiranje promjene stanja zaključuje se da se primjenom korištenih metoda može adekvatno testirati funkcionalnost Facebook aplikacije.

9. Literatura

1. A brief overview of Jira. URL: https://www.atlassian.com/software/jira/guides/getting-started/overview
2. Cohn, Mike. User Stories Applied: For Agile Software Development. URL: https://books.google.hr/books?id=SvIwuX4SVigC&printsec=frontcover&dq=user+story&hl=hr&sa=X&redir_esc=y#v=onepage&q=user%20story&f=false
3. EDUCBA. URL: https://www.educba.com/test-cases-vs-test-scenario/
4. First Contentful Paint (FCP) URL: https://web.dev/fcp/
5. Graham, Dorothy…[et al.]. Foundations of Software Testing: ISTQB certification. URL: https://www.utcluj.ro/media/page_document/78/Foundations%20of%20software%20testing%20-%20ISTQB%20Certification.pdf
6. Gray Box Testing, 2020. URL: http://softwaretestingfundamentals.com/gray-box-testing/
7. Gregory, Janet; Crispin, Lisa. Agile testing: a practical guide for testers and agile teams. URL: http://index-of.co.uk/Software-Testing/AGILE_TESTING_-_A_PRACTICAL_GUIDE_FOR_TESTERS_AND_AGILE_TEAMS.pdf
8. Guru99. URL: https://www.guru99.com/
9. Hundermark, Peter. Do better Scrum: An unofficial set of tips and insights into how to implement Scrum well.
10. ISTQB Agile Tester.
11. Jukić, Ivana. Automatsko testiranje programa. URL: https://zir.nsk.hr/islandora/object/foi:3888/preview
12. Kannan, Vaishnavi; Jhajharia Smita; dr. Verma, Seema. Agile vs waterfall: A Comparative Analysis. // International Journal of Science, Engineering and Technology Research (IJSETR) 3, 8(2014). URL: http://ijsetr.org/wp-content/uploads/2014/10/IJSETR-VOL-3-ISSUE-10-2680-2686.pdf
13. Ključni pokazatelji web stranice. URL: https://web.dev/vitals/
14. L. Bradford, „Everything You Need to Know About Software Testing Methods“. URL: https://www.thebalancecareers.com/all-you-need-to-knowabout-software-testing-methods-4019921

15. Lighthouse performance scoring: How Lighthouse calculates your overall Performance score, 2019. URL: https://web.dev/performance-scoring/
16. Lučić, Mijo. Prakse testiranja programskih proizvoda, 2019.
17. Martinović, Dragan. Testiranje programske podrške. URL: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=6512
18. PageSpeed Insights. URL: https://developers.google.com/speed/pagespeed/insights/?hl=hr
19. ReQtest: Alpha vs Beta Testing: How They are Different? URL: https://reqtest.com/testing-blog/alpha-vs-beta-testing/
20. S. Naik, P. Tripathy, Software testing and quality assurance: Theory and practice, New Jersey: John Wiley & Sons, Inc., 2008.
21. Stefanović, Kristina; Tošić, Goran. Agilne metodologije razvoja softvera: osnovne agilne metodologije. URL: http://files.kristina-stefanovic.webnode.com/200000105-0059b016e2/Survey.pdf
22. The Scrum Guide™ The Definitive Guide to Scrum: The Rules of the Game.
23. Try QA. URL: http://tryqa.com/
24. Walton, Philip. Web Vitals, 2020. URL: https://web.dev/vitals/