Prototip kao specifikacijsko sredstvo

Škaro, Dominik

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Škaro

PROTOTIP KAO SPECIFIKACIJSKO SREDSTVO

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Škaro

Matični broj: 35224037–R

Studij: Informacijski sustavi

PROTOTIP KAO SPECIFIKACIJSKO SREDSTVO

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Vjeran Strahonja

Varaždin, rujan 2022.

Dominik Škaro

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Razvoj računalnih aplikacija i programske podrške dolazi uz mnoštvo problema kao što je specificiranje korisničkih zahtjeva takvih da specifikacija zahtjeva bude kompletna i nedvosmislena, a istovremeno, zbog svoje cijene, unutar budžeta. Rješavanje i korekcija problema i dvosmislenosti u fazi specifikacije znatno je jeftinija nego ispravljanje problema računalne aplikacije nakon što je ušla u proizvodnju. Tema ovog završnog rada je prikazati mjesto i ulogu prototipa kao specifikacijskog sredstva koje upotpunjuje modele analize i specifikacije zahtjeva te konceptualnog i logičkog modeliranja. Detaljno će se analizirati i specificirati zahtjevi prema budućem sustavu, a koji se odnose na funkcionalnost rada te korisničko sučelje, odziv, performanse te ostale zahtjeve. Cilj je prikazati važnost prototipnog razvoja u svrhu razumijevanja potreba krajnjih korisnika informatičkih usluga. U praktičnom će dijelu rada detaljnije biti prikazan model prototipa gdje će se kroz studije slučaja razraditi specifikacija zahtjeva te konceptualni model budućeg sustava i pripadajući prototip.

Ključne riječi: prototip; specifikacija; sredstvo; analiza; zahtjev; modeliranje; sustav;

Sadržaj

1. Uvod

Posljednjih godina važnost specifikacije zahtjeva sve je veća. Glavni je cilj pružiti korisniku ono što očekuje te definirati ono što će mu pružiti sustav. Nitko ne želi započeti projekt nesiguran u točno ono što klijent traži. Pažljiva i dobro napisana specifikacija korisničkih zahtjeva štedi vrijeme i novac, a što je specifikacija korisničkih zahtjeva bolja, to je bolji rezultat. Veliku pomoć procesu specifikacije zahtjeva te samom razvoju aplikacije pruža prototipiranje. Međutim, prototipovi nisu učestali zbog svoje cijene, ali ispravljanje grešaka i rješavanje problema i dvosmislenosti u fazi specifikacije znatno je jeftinije nego ispravljanje istih nakon što je sustav pušten u proizvodnju. Prototip je izvediv model softverskog sustava koji omogućuje procjenu određenih funkcija kroz interakciju korisnika i programera s operativnim scenarijima. Prototipiranje je značajno za testiranje ideja i razumijevanja potreba krajnjeg korisnika, a omogućava odličan način za demonstraciju ideja korisnicima uz mogućnost brzog ponavljanja.

Jedan od primarnih izvora kontinuiranih poteškoća je nedostatak rane provjere valjanosti zahtjeva. Validacija zahtjeva je problematična budući da se zahtjevi često ne razumiju dobro prije samog razvoja, često se mijenjaju tijekom razvoja te se umnožavaju kao rezultat razvoja. (Wood i Kang, 1992, str. 3).

Prototipiranjem se izrađuje jednostavan eksperimentalni model predloženog proizvoda kako bi se provjerilo koliko dobro odgovara onome što korisnici žele putem povratnih informacija koje daju. Postoji više načina izrade prototipa, a prednosti izrade i najobičnijeg su sljedeće:

- Dobivanje čvrste osnove s koje se može kretati prema poboljšanjima dajući zainteresiranim strankama jasnu sliku o potencijalnim koristima, troškovima, a i rizicima povezanima s time kamo bi prototip mogao dovesti.
- Pogodnije rano prilagođavanje promjenama te na taj način izbjeći predanost lažnim idealnim verzijama krajnjeg proizvoda.
- Dobivanje povratnih informacija korisnika pomoću kojih se lakše određuju elementi koji funkcioniraju dobro te oni koji funkcioniraju loše u svrhu moguće revizije.
- Poboljšana i povećana uključenost korisnika koja pomaže u uklanjanju nesporazuma i pogrešne komunikacije tijekom procesa razvoja, a time se pruža osjećaj vlasništva svim zainteresiranim dionicima. Brže se dolazi do potrebnih izmjena na projektu te promjena specifikacije modela.

- Smanjeno vrijeme i troškovi izrade krajnjeg proizvoda. Prorotipiranjem se poboljšava kvaliteta zahtjeva i specifikacija koje se daju programerima. Ranijim utvrđivanjem onoga što korisnik stvarno želi rezultira jeftinijim softverom.

Ovaj je završni rad podijeljen na dva dijela. U prvom, teorijskom dijelu, će se objasniti prototipni razvoj softvera pozicioniran unutar metodologije sistemskog i programskog inženjerstva. Pokrit će se prednosti i nedostaci prototipnog razvoja kao specifikacijskog sredstva te će u drugom, praktičnom dijelu, kroz studije slučaja biti razrađena specifikacija zahtjeva te konceptualni model budućeg sustava i njemu pripadajući prototip.

2. Općenito o prototipiranju

Prototipiranje softvera uključuje izradu prototipa softverskih aplikacija koje predstavljaju funkcije proizvoda prije razvoja, a to znači da postoji vjerojatnost da prototip možda neće sadržavati krajnju logiku izvornog softvera. Glavni razlog za izradu prototipa je provjera valjanosti ideje, što znači da je prototipiranje korak u pretvaranju ideje u pravi proizvod. Put do pravog proizvoda je korištenje radnih modela koji se stalno usavršavaju na temelju povratnih informacija krajnjeg korisnika, a sama izrada prototipa se najčešće koristi za razvoj sustava sa značajnom interakcijom krajnjeg korisnika i složenim korisničkim sučeljima.

Prototipovi mogu biti izrađeni koristeći određene programske jezike, a rješenje kodiranog prototipa prilično je blizu verziji proizvoda spremnog za puštanje u „promet“. Ova se vrsta izrade prototipova preporučuje onima koji su sigurni u svoje vještine kodiranja. Prednost kodiranog prototipa je prepoznavanje ograničenja platforme, a to omogućuje dizajnerima da razumiju stvarne mogućnosti i ograničenja platforme za koju dizajniraju. Učinkoviti su, jer kodirani prototip može biti dobar temelj za potpuno funkcionalnu aplikaciju, ali cilj je implementirati dizajn na način s kojim ljudi mogu komunicirati što je brže moguće kako bi dali povratnu informaciju o samom prototipu, a za proizvodni tim to je važno kako bi zadovoljili korisničke zahtjeve.

2.1. Svrha prototipiranja

Izrada prototipova omogućuje razvojnom timu prikupljanje informacija o korisnikovim zahtjevima dopuštajući korisniku interakciju s prototipom. Stoga služi kao preliminarna verzija sustava iz kojeg se izdvajaju zahtjevi i na kojima se temelje sljedeće verzije. (Ogedebe i Jacob, 2012, str. 219).

„Uočeno je da se prototipiranje, u bilo kojem obliku, treba koristiti kako bi se što bolje zadovoljili zahtjevi korisnika. Međutim, izrada prototipa najkorisnija je prilikom izrade sustava koji će moći imati interakciju s korisnicima. Također, utvrđeno je da je izrada prototipa vrlo učinkovita u analizi i dizajnu sustava, posebno sustava u kojima je prisutnija uporaba ekranskih dijaloga. Što je veća interakcija između korisnika i prototipa, veća je korist koja se može dobiti izgradnjom sustava.“ (Crinnion, 1991, str. 18).

Također se vjeruje da sustavi s malo interakcije korisnika, kao što je skupna obrada ili sustavi koji uglavnom rade izračune, imaju malo koristi od izrade prototipova (Crinnion, 1991, str. 18). Ponekad, kodiranje potrebno za izvođenje funkcija sustava može biti preintenzivno, a potencijalni dobici koje bi izrada prototipova mogla pružiti su premali (Crinnion, 1991, str. 18).

Prednosti prototipa su mnoge, može se koristiti kao komunikacijski alat između razvojnog tima i korisnika kako bi se prevladali određeni problemi vezani uz analizu korisničkih zahtjeva te njihovu realizaciju. Izrada prototipova može pomoći u prevladavanju problema nedostatka korisničke interakcije s projektom, nepotpunih korisničkih zahtjeva te mijenjanja zahtjeva (Ogedebe i Jacob, 2012, str. 220). Prototipovi, odnosno povratne informacije dobivene od korisnika pomažu i olakšavaju razvojnom timu vršiti ispravke i izmjene na proizvodu koji se razvija. Prema mnogima, prototipiranje pruža i druge prednosti kao što su:

- Pruža proces za usavršavanje definicije korisničkih zahtjeva.
- Pruža formalnu specifikaciju utjelovljenu na radnoj replici.
- Više entuzijastičnog i konstruktivnog sudjelovanja krajnjih korisnika i kupaca.
- Prototip se lako može promijeniti, pa čak i odbaciti.
- Pokazuje napredak u ranoj fazi razvoja.
- Bolja priprema za kasnije faze razvoja zbog poznavanja prototipa (Ogedebe i Jacob, 2012, str. 220).

2.2. Tipovi prototipiranja

Izrada prototipova softvera ima mnogo različitih varijanti. Međutim, postoje dva tipa koji se koriste češće, a koji imaju različit utjecaj na životni ciklus razvoja proizvoda. To su jednokratno prototipiranje i evolucijsko prototipiranje.

2.2.1. Jednokratno prototipiranje

Jednokratni prototipovi razvijaju se iz početnih zahtjeva, ali se ne koriste za konačni proizvod i nisu alternativa za pisanu specifikaciju zahtjeva. Omogućuje brzu izradu prototipa i obavezu odbacivanja istog. Ako korisnici mogu dobiti brzu povratnu informaciju o svojim zahtjevima, mogli bi poboljšati zahtjeve rano u razvoju softvera (Martinez, 2020). Tada se promjene mogu napraviti rano u životnom ciklusu razvoja. Jednokratni prototip ima kratak vremenski rok unutar projekta te se koristi za brže i lakše razvijanje sučelja, a ovu vrstu izrade prototipova u bilo kojem trenutku može koristiti bilo tko (Jayasinghe, 2020). Jednokratni prototipovi su zapravo prezentacija za ograničenu svrhu.

Jednokratni prototip nije integriran u razvojni ciklus. Umjesto toga, uzimaju se njegovi rezultati, a prototip se odbacuje. Sama priroda jednokratnog prototipa je razlog zašto je dobio ovo ime. Model prototipa za jednokratnu upotrebu tu je samo da potvrdi funkcionalnosti i zahtjeve sustava. Jednom kada ih se shvati i na njima se radi, prototipovi za jednokratnu

upotrebu se odbacuju jer više ne dodaju nikakvu vrijednost (Martinez, 2020). Smanjenje rizika prilikom razvoja proizvoda ključni je cilj kada je u pitanju izrada prototipova. Ova stalna konstrukcija i odbacivanje više prototipova korisni su za projekte jer testiraju različite aspekte projekta (Martinez, 2020).

S različitim povratnim informacijama koje krajnji korisnici daju razvojnim timovima, ovaj tip prototipiranja omogućuje programerima da naprave različite promjene ili implementacije unutar projekta. Ovom posebnom vrstom izrade prototipa postiže se povratna informacija o potrebama kupca. Razvojni ciklus prototipa se produljuje, ali on postaje eksponencijalno učinkovit, a kvaliteta krajnjeg proizvoda ispada bolja. Najočitiji razlog za korištenje jednokratnih prototipova je taj što se mogu brzo napraviti. Ukoliko dođe do promjene na projektu nakon što je obavljena značajna količina posla, male promjene mogu zahtijevati velike napore za implementaciju jer softverski sustavi imaju mnogo ovisnosti (Crinnion, 1991, str. 18). Brzina je presudna u implementaciji prototipa za jednokratnu upotrebu, budući da se s ograničenim vremenom i novcem malo može potrošiti na prototip koji će biti odbačen (Crinnion, 1991, str. 18).

Jednokratna izrada prototipova vrlo je korisna za programere kada uzimaju u obzir povratne informacije korisnika. Obično programeri imaju problem implementirati cijeli projekt za kupce ili natjerati krajnje korisnike da im daju korisne povratne informacije. Jednokratni prototipovi imaju određene prednosti, kao što su:

- Isplativost – s uvođenjem većeg broja prototipova sigurno će biti prisutan i velik broj testiranja, a jeftinije je ukloniti i pronaći greške testiranjem prije razvoja finalnog proizvoda.
- Svestranost – moguća je primjena programskog jezika drugačijeg od onoga koji se koristi na glavnom projektu, a to ubrzava obradu razvoja pod pretpostavkom da se testira značajka koja se tiče glavnog projekta. (Martinez, 2020).

2.2.2. Evolucijsko prototipiranje

Evolucijsko prototipiranje jedno je od najpopularnijih tipova izrade prototipa. Ovaj tip prototipiranja uključuje niz poboljšanja samog prototipa. Prvi je zadatak dizajnirati i podijeliti sustav u nekoliko neovisnih modula. Svakom modulu daju se potrebne funkcionalnosti i zatim se prezentiraju kupcima (Martinez, 2020). Prikupljaju se povratne informacije kupaca i model se ponovno dorađuje. Ovaj proces se nastavlja sve dok se ne stvori konačni zadovoljavajući prototip.

Na temelju povratnih informacija kupaca, doći će vrijeme kada će kupci biti zadovoljni prototipom i njegovim funkcionalnostima, a to je vrijeme kada se zaustavlja ciklus podešavanja

projekta prema povratnim informacijama kupca. Prototip se razvija u svakoj fazi, a zato se naziva evolucijski prototip. Najčešće će se niz uključenih faza više usredotočiti na funkcionalnost nego na dizajn.

Evolucijski model prikladan je ako se isprobava novi proizvod ili tehnologija koja trenutno nije jasno shvaćena. U takvom scenariju, trenutne povratne informacije možda neće biti moguće sve dok korisnici to ne pokušaju koristiti. Nakon što ga sami isprobaju, njihovo razumijevanje proizvoda se poboljšava, a razvojni tim može prikupiti informacije kako bi ga poboljšali. Također, veliki projekti sa složenom funkcionalnošću bi odgovarali ovom modelu. To je zato što tako sofisticirani projekti imaju nekoliko pojedinačnih funkcionalnosti koje bi trebalo jednom provjeriti. (Martinez, 2020). Velike projekte lakše je podijeliti u nekoliko modula koji se mogu predstaviti korisnicima za neprocjenjive povratne informacije. Ukoliko zahtjevi projekta nisu jasno razumljivi ili još nisu u potpunosti određeni, evolucijska izrada prototipa mogla bi biti dobar model za prikaz proizvoda ili sustava. Kao i u svakom segmentu izrade prototipa, povratne informacije su od velike važnosti jer pomažu u usmjeravanju zahtjeva prema potrebama kupaca i korisnika. (Jayasinghe, 2020).

Wood i Kang (1992, str. 8) navode da gledano iz drugačije perspektive, cijeli životni ciklus proizvoda može se vidjeti kao niz sve detaljnijih evolucijskih prototipa te da je tradicionalno životni ciklus podijeljen u dvije različite faze: razvoj i održavanje. Prema iskustvima mnogih ovo je pokazalo da je ova razlika donekle proizvoljna i odaje da su zapravo veliki dio ili većina troškova životnog ciklusa softverskog proizvoda nastali nakon što je proizvod isporučen. Zato postoje različita gledišta na pojam kvalitete. Wood i Kang (1992, str. 8) navode da postoje nekoliko gledišta na pojam kvalitete, kao što su:

- Za programera, kvalitetan proizvod je onaj koji ispravno funkcionira u skladu sa specifikacijama.
- Za održavatelja, kvalitetan proizvod može se smatrati onim koji je lako podložan izmjenama i poboljšanjima.
- Za korisnika je kvalitetan onaj proizvod koji ima „pravi“ izgled i dojam te performanse i ponašanje.
- Kupcu je svaki od gore navedenih pogleda važan aspekt kvalitete.

Zbog ove temeljne razlike u percepcijama, nije neuobičajeno da organizacija za održavanje potroši dosta vremena i truda na razumijevanje dizajna i implementacije proizvoda koja je pri ruci, a čak i kada rezultirajući proizvod ne ispunjava očekivanja korisnika. Tendencija da se održavanje smatra odvojenom aktivnošću od razvoja često rezultira ozbiljnim podcjenjivanjem troškova životnog ciklusa (Wood i Kang, 1992, str. 8). Kao što je gore

navedeno, evolucijska izrada prototipova oslanja se na nekoliko ciklusa za poboljšanje prototipa projekta, a postoje određene prednosti korištenja ovog modela izrade, kao što su:

- Pogodnost korištenja u velikim projektima. Veliki se projekti mogu lako rastaviti na modele koji se mogu zasebno implementirati za kupce te prikazati korisnicima. Jedna od najvećih prepreka evolucijskoj izradi prototipa je teškoća dijeljenja projekta u nekoliko segmenata koji su prihvatljivi kupcu i lako se mogu povećati od strane programera. (Martinez, 2020).
- Smanjenje pogrešaka. Činjenica da je projekt segmentiran u module znači da se zasebni moduli mogu temeljito testirati. Temeljitim testiranjem moguće je znatno minimizirati pogreške u različitim modelima temeljnog projekta. (Martinez, 2020).
- Minimiziranje resursa. Segmentiranjem modela te nizom obnova osigurava se da se nekoliko modela neovisno smatra prikladnim. (Martinez, 2020).
- Zadovoljava zahtjeve korisnika. Uvijek je ključno implementirati proizvod koji odgovara potrebama potrošača. Stavljanjem prototipova u cikluse povratnih informacija i dorada prema zahtjevima korisnika, moguće je integrirati potrebe korisnika u sustav (Martinez, 2020). Nakon toga mogućnost za pritužbe i nezadovoljstvo od strane korisnika i kupca sve je manja jer sudjeluju u dizajnu.

2.3. Razine vjernosti prikaza prototipa

Vjernost prikaza odnosi se na razinu detalja i funkcionalnosti koje se uključuju u prototip, a različite se razine razlikuju prema metodi izrade. U konačnici, vjernost prototipa odnosi se na to kako prenosi izgled i dojam konačnog proizvoda. Metoda izrade razlikuje prototipove na papiru do računalno izrađenih prototipova. Razina vjernosti prikaza najčešće ovisi o tome u kojoj se fazi razvoja nalazi proizvod (Babich, 2017). Prototipovi ne izgledaju nužno kao konačni proizvodi, a to znači da mogu imati različitu vjernost prikaza.

Vjernost prikaza može se razlikovati u ovim područjima:

- Vizualni dizajn
- Sadržaj
- Interaktivnost

Postoji više vrsta prototipova koji se nalaze između ova dva krajnja tipa:

- Prototipovi niske razine vjernosti

- Prototipovi visoke razine vjernosti

Tipovi prototipova, odnosno razine njihovih vjernosti u procesu razvoja proizvoda ovise o fazi razvoja te se biraju na temelju ciljeva izrade, potpunosti dizajna i dostupnih resursa.

2.3.1. Prototip niske razine vjernosti

Izrada prototipova niske vjernosti brz je i jednostavan način za prevođenje koncepata dizajna visoke razine u opipljive artefakte koje je moguće testirati. Jedna od najvažnijih uloga prototipova niske vjernosti je provjera i testiranje funkcionalnosti, a ne vizualnog izgleda proizvoda (Babich, 2017). Prototipovi niske vjernosti obično služe kao početna točka dizajna te su prilično grubi i stvoreni bez mreže i točnosti, odnosno razmjera.

Postoji više prototipnih tehnika izrade prototipa niske razine vjernosti. Najčešće su takvi prototipovi skice na papiru ili žičani okviri koji su korak dalje od papirnatih skica, odnosno 2D skeletni obrisi neke softverske aplikacije.

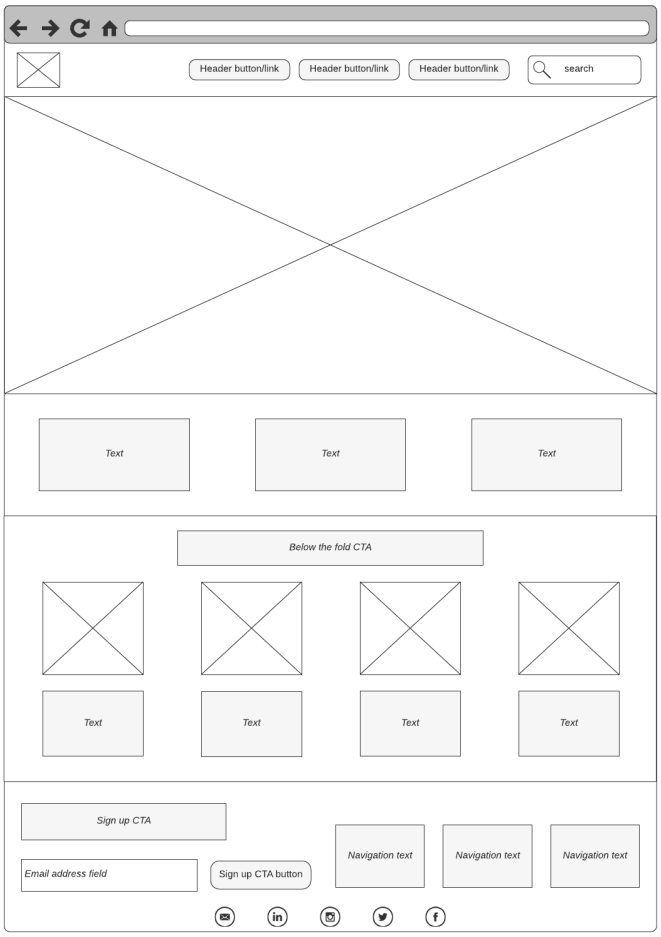
Izrada prototipa na papiru omogućuje izradu prototipa digitalnog sučelja bez korištenja digitalnog softvera. Tehnika se temelji na izradi ručnih crteža različitih zaslona koji predstavljaju korisnička sučelja. Iako je ovo relativno jednostavna tehnika, može biti korisna kada proizvodni tim treba istražiti različite ideje, a osobito je bitna u ranoj fazi razvoja kada tim iskušava različite načine pristupa. Prednosti korištenja ove tehnike uključuju dopuštanje ranog testiranja koje proizvodnim timovima omogućuje pronalazak općeg problema, poput nejasne informacijske arhitekture te podržavanje brzog eksperimentiranja (Babich, 2017). Različiti elementi korisničkog sučelja mogu se nacrtati, izrezati, a zatim sastaviti na novom komadu papira, a s papirnatim je prototipovima moguće oponašati složene interakcije, poput pomicanja. (Babich, 2017).

S druge strane, žičani okvir je vizualni prikaz stranice proizvoda koju dizajner može koristiti za raspored elemenata stranice. Žičani okviri koji se mogu kliknuti najjednostavniji su oblik interaktivnog prototipa stvorenog povezivanjem statičnih okvira. Baš kao i papirnati prototipovi, žičani okviri koji se mogu kliknuti često ne izgledaju kao gotov proizvod. Prednosti korištenja ove tehnike uključuju postojanje rezultata dizajna koji se mogu ponovno upotrijebiti (Esposito, 2018). Tijekom određene faze procesa dizajna postoje okviri koji predstavljaju dizajn korisničkog sučelja proizvoda. U većini slučajeva moguće ih je koristiti za stvaranje tijeka koji se može kliknuti. Također, lako je promijeniti izgled i raspored okvira. Dizajneri lako mogu prilagoditi žičane okvire na temelju povratnih informacija korisnika i ponoviti postupak testiranja. Elementi su crno-bijeli, ali je moguće koristiti nijanse sive kako bi se pojedini elementi vizualno istaknuli. (Babich, 2017).

Obje su tehnike usmjerene na pružanje najbržeg mogućeg načina za ponavljanje dizajnerskih ideja sve dok i projektni tim i dionici nisu zadovoljni osnovama. One su osnovna smjernica za izradu neke aplikacije koje dizajneri i programeri trebaju slijediti. Međutim, dizajneri i programeri drugačije gledaju na prototipove niske vjernosti. Razvojni ih programeri obično koriste za bolje razumijevanje temeljne funkcije web-stranice ili aplikacije, dok ih dizajneri mogu koristiti za prikaz toka navigacije između raznih zaslona. (Esposito, 2018).

Prednosti prototipova niske razine vjernosti su niska cijena, brzina izrade te jasnoća izrade. Mogućnost izrade prototipa iznimno niske razine vjernosti u kratkom vremenu omogućuje proizvodnim timovima istraživanje različitih ideja bez previše truda. Također, budući da izrada prototipova niske razine vjernosti ne zahtjeva posebne vještine, više ljudi može biti uključeno u proces dizajna. (Esposito, 2018).

Postoje i određeni nedostaci, a među njima je nesigurnost tijekom ispitivanja. S prototipovima niske vjernosti moglo bi biti nejasno što bi trebalo funkcionirati, a što ne, odnosno od korisnika se zahtijeva mašta, ograničavajući ishod korisničkog testiranja. Također, postoji i ograničena interaktivnost. Nemoguće je prenijeti složene animacije ili prijelaze pomoću ove vrste prototipa.



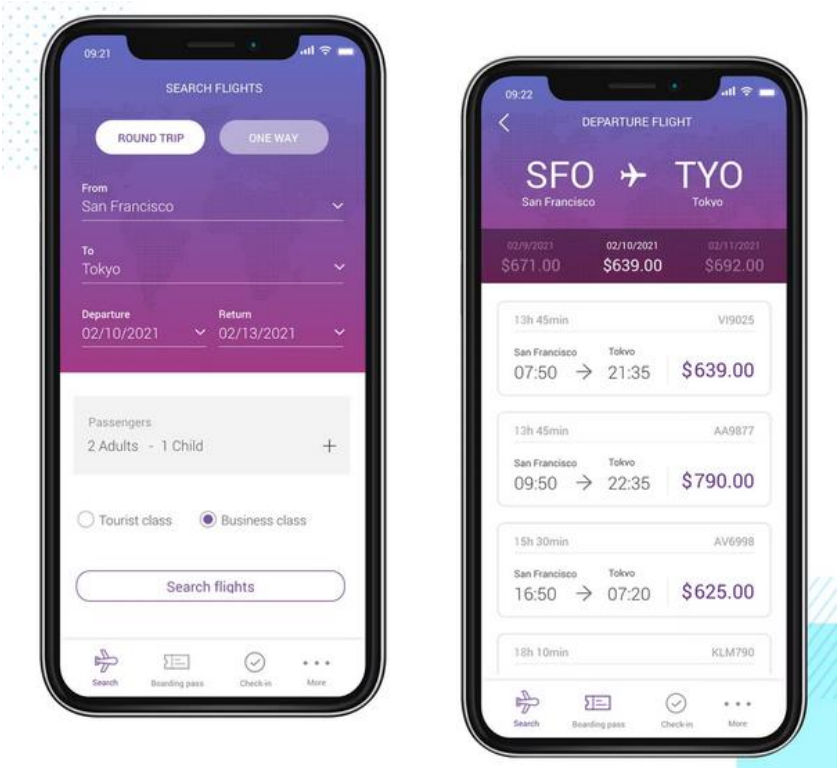
Slika 1. Prikaz prototipa niske vjernosti prikaza (*Low-fidelity mockup*, bez dat.)

Prototipovi niske razine vjernosti koriste se kada želimo brzo prenijeti ideje i dobiti povratne informacije. U ranim fazama planiranja, prototipovi niske razine vjernosti mogu pripomoći identificirat potencijalne probleme dizajna prije nego što se potroši previše vremena na izradu prototipa boljeg dizajna (Esposito, 2018). Također, dobro funkcioniraju u bilo kojoj fazi razvoja kada se razmišlja o idejama ili prikupljanja podataka od dionika bez dizajnerske pozadine. S takvih su prototipova izostavljeni detalji koji bi mogli odvratiti pozornost, poput slika ili tipografije. Stvaranje savršenog dizajna u ranim fazama razvoja nema smisla. Prototip prolazi kroz mnogo iteracija prije nego dođe do konačnog dizajna.

2.3.2. Prototip visoke razine vjernosti

Prototipovi visoke vjernosti izgledaju i funkcioniraju što je moguće sličnije stvarnom proizvodu koji će biti isporučen. Timovi stvaraju prototipove visoke vjernosti kada dobro razumiju što će graditi i moraju to testirati sa stvarnim korisnicima ili dobiti odobrenje konačnog dizajna zainteresiranih dionika (Babich, 2017). Prototipovi visoke vjernosti prikaza bolje su za

dokumentaciju zbog povećane razine detalja, vrlo su funkcionalni i interaktivni. Izgledom su vrlo slični konačnom proizvodu, s većinom razvijenih i integriranih dizajnerskih sredstava i elemenata. Takvi se prototipovi često koriste u kasnijim fazama za testiranje upotrebljivosti i identificiranje problema u tijeku rada (Esposito, 2018).



Slika 2. Prikaz prototipa visoke vjernosti prikaza (*High-fidelity mockup*, bez dat.)

Na slici 2. prikazan je prototip visoke vjernosti prikaza mobilne aplikacije. Neke od osnovnih karakteristika prototipova visoke vjernosti uključuju vizualan dizajn koji je realističan i detaljan, a svi elementi sučelja, razmaci i grafika su ukomponirani u cjelinu i izgledaju kao prava aplikacija ili web-stranica (Ibragimova, 2016). Dizajneri koriste sadržaj sličan stvarnom, a prototip uključuje većinu ili sav sadržaj koji će se pojaviti u konačnom dizajnu. Za razliku od prototipova niske vjernosti koje sadrže pseudolatinske odlomke teksta i sive okvire ispunjene znakom „X“ koji označavaju sliku, prototipovi visoke vjernosti mogu sadržavati stvarne istaknute slike i odgovarajući pisani sadržaj. Riječ „visoka“ odnosi se na razinu sveobuhvatnosti koja omogućuje detaljan pregled upotrebljivosti i donošenje zaključaka o ponašanju korisnika. Prototipovi visoke razine vjernosti vrlo su realistični prilikom interakcija. (Esposito, 2018).

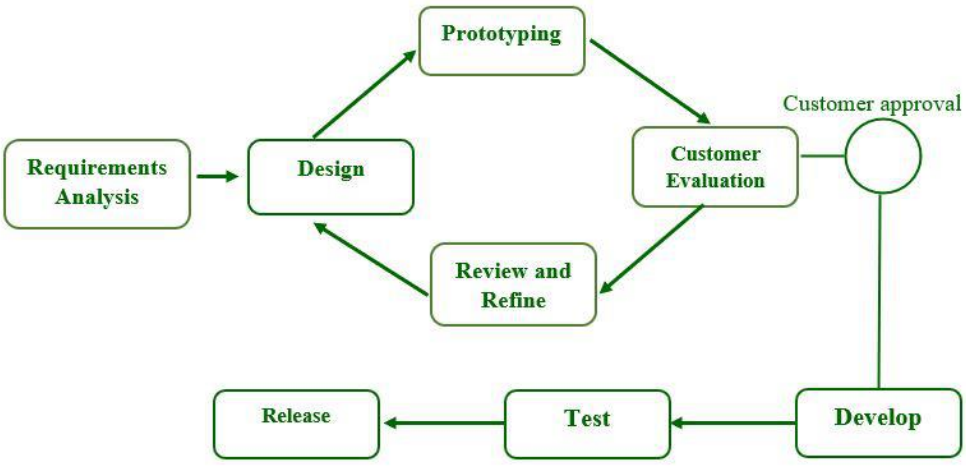
Kada se koriste prototipovi visoke vjernosti? Koriste se kada postoji vizualan dizajn proizvoda i ideja o interaktivnim elementima, kao što su navigacijske sheme, animacije i interakcije. Također, koriste se kada se žele testirati detalji proizvoda u smislu elemenata korisničkog sučelja ili shema boja, ali najbitnije kada se želi znati što korisnici misle o proizvodu i mišljenje o dizajnu i funkcionalnostima. (Ibragimova, 2016).

Prototipovi visoke vjernosti daju značajne povratne informacije tijekom studija upotrebljivosti jer sudionici mogu komunicirati s prototipom kao što bi mogli s konačnim proizvodom. To znači da će tijekom sesija testiranja upotrebljivosti sudionici testiranja vjerojatnije ponašati prirodno. Prednost je i mogućnost testiranja specifičnih elemenata korisničkog sučelja ili interakcija. Ibragimova (2016) navodi da je uz interaktivnost visoke vjernosti moguće testirati grafičke elemente kao što je dostupnost ili specifične interakcije, kao što su animirani prijelazi i mikrointerakcije.

Izrada prototipova s krivom vjernošću jedna je od najvećih pogrešaka u izradi prototipa. Rješenje treba procijeniti na temelju pojedinačnog scenarija. Za brzinu i fleksibilnost, prototipovi niske vjernosti imaju prednost, ali ako se proizvod već razvija na temelju povratnih informacija krajnjeg korisnika, onda prednost imaju prototipovi visoke vjernosti.

2.4. Faze prototipiranja

Izrada prototipa je metodologija razvoja softvera koja se fokusira na korištenje radnih modela koji se stalno usavršavaju na temelju povratnih informacija krajnjeg korisnika. Potrebno je staviti naglasak na zadnji dio prethodne rečenice. Cilj prototipa je biti prihvaćen od strane krajnjeg korisnika kako bi se što bolje i jednostavnije prešlo na fazu razvoja pravog proizvoda koji će uz pomoć napravljenog prototipa zadovoljiti specifikacije korisničkih zahtjeva.



Slika 3. Faze prototipiranja (Shsahu, 2021)

2.4.1. Prikupljanje i analiza zahtjeva

Analiza zahtjeva prvi je korak u razvoju modela prototipa. Tijekom ove faze precizno su definirani elementi i dijelovi željenog sustava. Analiza zahtjeva proces je određivanja očekivanja korisnika za novi ili modificirani proizvod. Zahtjevi u pravilu moraju biti mjerljivi, relevantni i detaljni. U programskom inženjerstvu takvi se zahtjevi nazivaju funkcionalne specifikacije. Analiza zahtjeva važan je dio upravljanja projektom (Sahu, 2021).

Faza analize zahtjeva uključuje komunikaciju s klijentom kako bi se odredila određena očekivanja specifičnih značajki, rješavanje sukoba ili dvosmislenosti u zahtjevima prema zahtjevima različitih korisnika ili grupa korisnika, izbjegavanje mijenjanja značajki i dokumentiranje svih aspekata procesa razvoja proizvoda od početka do kraja. Cilj je da konačni sustav ili proizvod bude u skladu s potrebama klijenta, a ne oblikovanje očekivanja korisnika kako bi odgovarale zahtjevima. (Sahu, 2021).

Analiza zahtjeva timski je rad koji zahtijeva kombinaciju hardverske, softverske i inženjerske ekspertize ljudskih faktora, kao i vještina u ophođenju s ljudima, jer je to početna i temeljna faza za razvoj sustava ili proizvoda, a izradom prototipa proizvodni timovi mogu što prije dobiti povratne informacije korisnika. Promatranje korisnika u interakciji s prototipom pomaže procijeniti kvalitetu specifikacije zahtjeva.

2.4.2. Početni dizajn

Druga faza može se sastojati od idejnog projekta, odnosno brzog dizajna. Tijekom ove faze formira se osnovni dizajn sustava. Međutim, to nije potpuni dizajn, nego korisniku pruža brzi pregled sustava dajući kratku ideju o sustavu i pomaže u razvoju prototipa. Moguće je brzo dobiti povratnu informaciju korisnika koja svakako pomaže usmjeriti razvoj proizvoda te poboljšanje nekih osnovnih elemenata sustava. (Sahu, 2021).

Prednost ove faze je brza vizualizacija željenog sustava ili proizvoda te lakša procjena izvedivosti prije implementacije samog dizajna.

2.4.3. Izrada prototipa

Tijekom ove faze izrađuje se prototip kojemu je namjena podržati informacije i znanje stečenih brzim dizajnom. Izradom prototipa omogućit će se procjena određenih funkcija kroz interakciju korisnika i programera s operativnim scenarijima. (Sahu, 2021).

2.4.4. Procjena korisnika

U ovoj se fazi predloženi sustav prezentira klijentu na preliminarno testiranje. Potrebno je prikupiti informacije o snagama i slabostima danog prototipa te ih istražiti i po potrebi promijeniti i popraviti. Povratne informacije i prijedlozi korisnika se prikupljaju i prosljeđuju proizvodnom timu. (Sahu, 2021).

Početna procjena korisnika usredotočuje se na to koliko dobro korisnici mogu naučiti i koristiti proizvod za postizanje svojih ciljeva. Također se odnosi na učinkovitost te sveukupno zadovoljstvo korisnika.

Testira se upotrebljivost sustava ili proizvoda, a upotrebljivost nije jednodimenzionalno svojstvo proizvoda, nego je kombinacija čimbenika poput intuitivnosti dizajna, jednostavnosti učenja te pamtljivosti. Intuitivnost dizajna pomaže pri razumijevanju arhitekture i lakše navigacije kroz proizvod. Jednostavnost učenja određuje koliko brzo korisnik koji nikada prije nije vidio korisničko sučelje može izvršiti osnovne zadatke, a učinkovitost korištenja je faktor koji pokazuje koliko brzo iskusan korisnik može izvršiti određene zadatke. Faktor pamtljivosti pokazuje koliko se brzo ili može li se uopće korisnik, nakon posjeta stranici, sjetiti učinkovito ju koristiti. U obzir ulazi i učestalost pogrešaka prilikom testiranja, a pokazuje koliko često korisnici griješe prilikom korištenja sustava te koliko su te pogreške ozbiljne. (Sahu, 2021).

2.4.5. Prerada prototipa

Ako korisnik nije zadovoljan trenutnim modelom, potrebno je pregledati povratne informacije i prijedloge korisnika i bolje im se prilagoditi. Do nezadovoljstva korisnika može

doći zbog problema u analizi zahtjeva, ali i zbog propusta proizvodnog tima. Kada je korisnik zadovoljan nadograđenim modelom, kreće se u izgradnju konačnog sustava ili proizvoda temeljenog na odobrenju konačnog prototipa.

2.4.6. Implementiranje proizvoda i održavanje

Jednom kada se konačan proizvod ili sustav razvije na temelju konačnog prototipa, on se temeljito testira te se podvrgava rutinskom održavanju kako bi se moguće greške svele na minimum i spriječili neočekivani kvarovi.

3. Specifikacija zahtjeva

Kako bi se moglo razumjeti što je to specifikacija zahtjeva, potrebno je znati razlikovati pojmove zahtjeva i specifikacije zahtjeva. Zahtjevi su nešto što je potrebno, odnosno zahtjevi opisuju što proizvod ili sustav trebaju raditi, kvalitete koje trebaju posjedovati i ograničenja u njegovom radu. Specifikacija zahtjeva skup je svih zahtjeva koji se trebaju nametnuti dizajnu i verifikaciji proizvoda. Specifikacija može sadržavati i druge relevantne podatke potrebne za izradu projekta te provjeru i održavanje proizvoda. Uglavnom se rade u obliku dokumenata. (Duncan, 1999).

Prototipovi omogućuju da se zahtjevi potvrde, poništе, modificiraju ili zamijene rano u procesu razvoja proizvoda, čime pomažu izbjeći rizik i troškove velikih izmjena dizajna nakon saznanja da su koncept dizajna i upotrebljivost neispravni tijekom faze razvoja proizvoda. Izrada prototipova može pratiti softverske zahtjeve zbog kontinuiranog testiranja dokumentiranih zahtjeva sve dok se svi zahtjevi ne potvrde. U jednoj iteraciji prototipa neki se zahtjevi potvrđuju dok se pojavljuju nova pitanja koja se trebaju riješiti u sljedećoj iteraciji. Pregledom zahtjeva i povratnih informacija daje se osnova za zadržavanje onoga što je dobro i odbacivanje onoga što je loše.

3.1. Poticaj za ranu i kontinuiranu validaciju

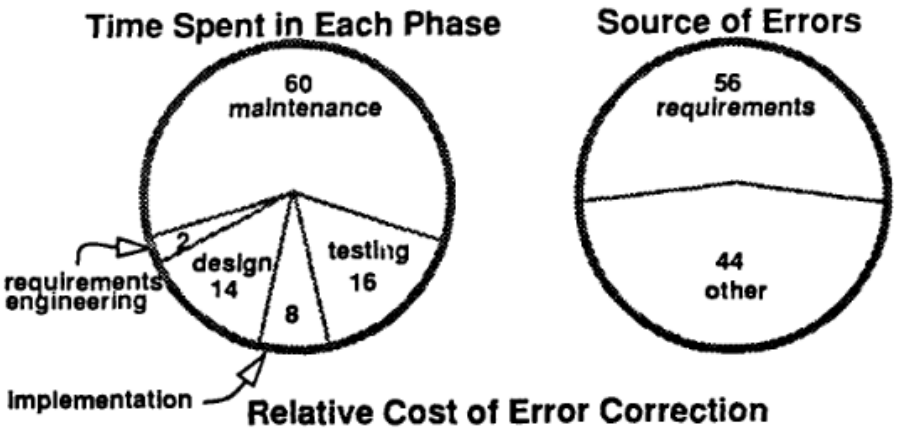
Wood i Kang (1992, str. 3) navode da iako je došlo do značajnog napretka u metodama softverskog inženjeringa i alata koji se koriste pri izradi istih tijekom proteklih godina, inženjering zahtjeva i dalje ostaje jedno od ključnih problematičnih područja u razvoju složenih softverskih sustava. Jedan od primarnih izvora kontinuiranih poteškoća prilikom razvoja softverskog sustava je nedostatak rane provjere valjanosti zahtjeva. Validacija zahtjeva može biti prilično problematična budući da se zahtjevi često ne razumiju dobro prije samog razvoja te se često mijenjaju i umnožavaju kao rezultat samog razvoja.

Wood i Kang (1992, str. 3) smatraju kako je moguće statistički dokazati važnost rane validacije zahtjeva te da je u njihovim istraživanjima čak 54% svih pogrešaka ikada otkrivenih u softverskim projektima proučavanim na TRW-u otkriveno nakon faza kodiranja i testiranja te da se većina tih pogrešaka (83%) može pripisati zahtjevima i fazama dizajna, a ne fazi kodiranja (17%). Mnoge pogreške u zahtjevima prosljeđuju se neotkrivene u kasnije faze životnog ciklusa, a ispravljanje tih pogrešaka tijekom ili nakon implementacije pokazalo se izuzetno skupim. Također, njihova istraživanja pokazuju da su rani popravci nedostataka

obično dvostruko jeftiniji od kasnih popravaka i nedostataka, a rani problemi u zahtjevima i nedostaci dizajna obično ostavljaju ozbiljne operativne posljedice.

Basili i Weiss (1981, str. 8) navode da su najčešće pogreške koje se čine tijekom inženjeringa zahtjeva tehničke te da je 77% svih pogrešaka zahtjeva pronađeno u operativnom programu leta mornaričkog zrakoplova A-7E bilo greška samog pisanja, od kojih su 49% bile netočne činjenice, a 31% propusti. Nedosljednost i dvosmislenost čine oko 18% svih nepismenih pogrešaka.

Slika 4 prikazuje važnost rane validacije zahtjeva, a iz slike je vidljivo vrijeme provedeno u svakoj fazi. Vrijeme održavanja te sam trošak održavanja moguće je smanjiti pravilnom analizom i specifikacijom zahtjeva.



Relative Cost of Error Correction

Stage	Relative Cost of Repair
Requirements	1
Design	5
Implementation	10
Unit Test	20
Acceptance Test	50
Maintenance	200

Slika 4. Važnost rane validacije zahtjeva (Wood i Kang, 1992, str. 3)

3.2. Vrste zahtjeva prema razini detalja

Arnowitz, Arent i Berger (2006, str. 34) navode da prilikom stvaranja softvera, zahtjevi mogu proizaći iz više različitih izvora, kao što su tržišta, krajnji korisnici, kupci i tehničke mogućnosti. Iz perspektive izrade prototipa zahtjeve je moguće razvrstati u više kategorija koje se razlikuju prema razini detalja i prema funkcionalnostima.

Jović, Frid i Ivošević (2019, str. 22) tvrde da prema razini detalja razlikujemo:

- Korisničke zahtjeve (engl. *user requirements*)
- Zahtjeve sustava (engl. *system requirements*)

Korisnički zahtjevi su izjave na prirodnom jeziku zajedno s odgovarajućim obrascima ili dijagramima koji detaljno opisuju usluge koje pruža sustav i operativna ograničenja s kojima mora biti usklađen. Zahtjevi korisnika prvenstveno su usmjereni na potrebe korisnika, što znači da korisnički zahtjevi zadovoljavaju kupca. U suštini, zahtjevi su takvi da korisnik želi ili sposobnost da izvede neku funkciju ili radnju sa sustavom. Korisnički zahtjevi podrazumijevaju razumljivost i korisnicima bez tehničkog znanja. („GuidingCode“, 2022).

S druge strane, zahtjevi sustava sastoje se od strukturiranog dokumenta koji detaljno opisuje funkcije sustava, njegove usluge te određena ograničenja. Ovaj je dokument uglavnom pisanog oblika i služi kao alat za programere sustava za izgradnju i implementaciju dizajna sustava. Sadrži funkcionalnost koja je potrebna sustavu kako bi mogao ispuniti zahtjeve korisnika. Sastoji se od svega što sustav mora sadržavati da bi ga se izgradilo ili razvilo. Često se opisuju kao proširena verzija korisničkih zahtjeva koje inženjeri koriste na početku dizajna sustava. Stoga zahtjevi sustava služe kao nacrt koji treba slijediti definirajući dijelove sustava koji će se implementirati, tako da djeluju kao ugovor između klijenta i izvođača. („GuidingCode“, 2022).

Korisnički zahtjevi:

1. Programski sustav za ministarstvo zdravlja generirat će mjesečna izvješća za upravu u kojima će predočiti cijenu lijekova koji se propisuju za svaku kliniku tijekom tog mjeseca.

Zahtjevi sustava:

1.1 Na zadnji radni dan u mjesecu, generirat će se sažetak o propisanim lijekovima, njihovoj cijeni i klinikama koje su ih propisale.

1.2 Sustav će automatski generirati izvješće spremno za ispis nakon 17:30 na zadnji radni dan u mjesecu.

1.3 Izvješće će biti napravljeno i za svaku kliniku posebno i popisat će pojedinačne nazive lijekova, ukupan broj recepata, broj propisanih doza i ukupnu cijenu propisanih lijekova.

1.4 Ako su lijekovi dostupni u različitim dozama (npr. 10 mg, 20 mg), zasebna izvješća će se napraviti za svaku postojeću dozu.

1.5 Pristup svim izvješćima o cijenama bit će ograničen na sve ovjerene korisnike koji se nalaze na kontrolnoj listi s razinom pristupa: "uprava".

Slika 5. Primjeri korisničkog zahtjeva i zahtjeva sustava (Jović i ostali, 2019, str. 23**)**

Izradom prototipa nekog sustava omogućuje se pretpregled na to kako će se neki zahtjevi moći izvršiti i je li ih moguće realizirati na stvarnom proizvodu. Korištenjem programskog alata kao što je Figma jednostavnije je vizualizirati korisničke zahtjeve te time odrediti na koji način i kojim redoslijedom treba implementirati određene elemente sustava. Izradom prototipova i dobrom suradnjom klijenta, izvođača i korisnika sustava moguće je dobiti korisne povratne informacije koje proizvodni tim analizira i oblikuje prema zahtjevima korisnika te koje utječu na poboljšanje elemenata krajnjeg proizvoda.

3.3. Vrste zahtjeva prema funkcionalnostima

Prema funkcionalnostima, zahtjeve se može podijeliti na funkcionalne i nefunkcionalne zahtjeve te ih je važno razlikovati. Funkcionalni zahtjevi opisuju što bi proizvod trebao raditi, dok nefunkcionalni zahtjevi postavljaju ograničenja na to kako bi proizvod to trebao činiti.

Funkcionalni se zahtjevi odnose na specifične funkcije proizvoda, a njihovo definiranje, mjerenje i testiranje jednostavnije je u odnosu na nefunkcionalne zahtjeve koji su apstraktniji. Nefunkcionalni zahtjevi nameću ograničenja na implementaciju funkcionalnih zahtjeva u smislu izvedbe, sigurnosti, pouzdanosti, skalabilnosti i prenosivosti.

Često se dogodi da funkcionalni zahtjev ima skup povezanih nefunkcionalnih zahtjeva, na primjer:

- Funkcionalni zahtjev: „Sustav mora omogućiti korisniku slanje e-pošte putem obrasca za kontakt.“
- Nefunkcionalni zahtjev: „Kada se pritisne gumb za slanje e-pošte, poruka potvrde mora se učitati unutar dvije sekunde.“

Jović i ostali (2019, str. 25) navode da su funkcionalni zahtjevi izjave o uslugama koje programski proizvod mora pružati, kako će sustav reagirati na određeni ulazni poticaj te kako bi se trebao ponašati u određenim situacijama. U nekim slučajevima, funkcionalni zahtjevi trebaju eksplicitno definirati i što sustav ne treba raditi. Funkcionalni zahtjevi su kompletni ako sadrže opise svih zahtijevanih mogućnosti, dok su konzistentni ako ne sadržavaju konflikte ili proturječne tvrdnje. U praksi je nemoguće postići kompletan i konzistentan dokument o funkcionalnim zahtjevima. (Jović i ostali, 2019, str. 25).

Jović i ostali (2019, str. 25) navode da su nefunkcionalni zahtjevi ograničenja u uslugama i funkcijama programske potpore te da je najgrublja podjela na tri tipa:

- Zahtjevi programske potpore – specificiraju na koji se osobit način treba ponašati isporučeni proizvod kao npr. Vrijeme odziva, podržani sustav, komunikacijski protokol i sl. (Jović i ostali, 2019, str. 25).
- Organizacijski zahtjevi – rezultat organizacijskih pravila i procedura kao npr. Uporaba propisanog normiranog procesa razvoja, korištenje uvijek točno određenog programskog jezika pri razvoju. (Jović i ostali, 2019, str. 25).
- Vanjski zahtjevi – izvan sustava i razvojnog procesa, npr. Postizanje međusobne operabilnosti s drugim sustavima, zakonski zahtjevi i drugo. (Jović i ostali, 2019, str. 25).

Ovisno o tipu prototipa koji je u izradi te prilikom prototipiranja, postoji mogućnost da neće svi zahtjevi koji su dostupni biti od pomoći za izradu prototipa. Stoga je važno podijeliti prototip s ključnim dionicima kako bi se osigurala potpuna pokrivenost. Za poslovne zahtjeve bilo bi najbolje potražiti pomoć dionika marketinškog tima. Za funkcionalne zahtjeve može pripomoći onaj tko je proveo istraživanja korisnika ili tržišta. Za tehničke zahtjeve, programeri mogu pružiti kritične informacije. Zahtjevi za upotrebljivost mogu proizaći iz istraživanja korisnika, analize određenih zadataka te provjere upotrebljivosti trenutnog prototipa. Zahtjevi mogu imati neželjene nuspojave koje, prije izrade konačnog proizvoda, može izložiti samo prototip.

3.4. Problemi u specificiranju zahtjeva

Kako bi zadovoljili dane zahtjeve, potrebno ih je analizirati, a zatim specificirati. To se odvija u fazama analize i specifikacije zahtjeva softvera. Nepisano je pravilo da se prilikom specifikacije novog računalnog sustava radi i specifikacija zahtjeva te se ista predaje na odobravanje. Međutim, postoji mogućnost dolaska do problema prilikom samog čitanja specifikacija. Glavni je problem stvoriti imaginarnu sliku rada sustava u određenom okruženju čitajući specifikacije s papira. U obzir ulazi i kvaliteta specifikacija, a loše napisana specifikacija može biti:

- Dvosmislena – ima nekoliko značenja ili interpretacija.
- Nekohezivna – ne obuhvaća potrebnu funkcionalnost iz perspektive poslovanja, korisnika ili podataka.
- Nepotpuna – ne pokriva sve što se treba riješiti s proizvodom ili sustavom.
- Nedosljedna – dijelovi zahtjeva negiraju informacije drugih zahtjeva.

- Zastarjela – specifikacije nisu ažurirane jer su identificirane nove potrebe ili je uključena nova funkcionalnost.
- Neizvediva – nalaže funkcionalnost koja se ne može implementirati.
- Neorganizirana – otežava određivanje međusobnog odnosa zahtjeva.
- Napisana nemajući na umu dionike – zahtjevi su napisani bez jasnog razumijevanja onoga što treba korisniku. („Solving the Most Common Problems with Requirements Specifications“, bez dat.).

Prototip pomaže stvoriti imaginarnu sliku rada sustava u određenom okruženju te već prilikom prvog pregleda prototipa od strane korisnika moguće je riješiti neke od gore navedenih problema specifikacije zahtjeva. Jasnim prikazom prototipa postavlja se naglasak na prvobitnu valjanu interpretaciju proizvoda.

Gomaa i Scott (1981, str. 334) navode da korištenje formalnog jezika specifikacije sustava kao što je PSL (engl. *Problem Statement Language*) i RSL (engl. *Requirements Specification Language*) može pomoći osigurati da specifikacija bude jednoznačna i dosljedna. Međutim, ne može osigurati da je specifikacija točna, to jest problem je možda točno naveden, ali to možda nije problem koji korisnik želi riješiti. Također, navode da se ne može osigurati da je specifikacija potpuna te da su neka funkcija ili skup funkcija možda u potpunosti izostavljeni. Prema njihovim istraživanjima pokazalo se da je 20% pogrešaka u specifikaciji zahtjeva nekoliko velikih softverskih projekata bilo zbog nedostajućih informacija, a daljnjih 30% zbog netočnih zahtjeva.

Tipičnom korisniku koji u nekim slučajevima mora odobriti specifikaciju zahtjeva može biti teško utvrditi je li specifikacija napisana formalnim jezikom doista zadovoljava njegove zahtjeve. Iz tog razloga neki se specifikacijski alati temelje na grafičkom pristupu specificiranju korisničkih zahtjeva, a pristup se često temelji na dijagramima protoka informacija. Jedna od prednosti grafičkog pristupa je to što su ti dijagrami razumljivi korisnicima koji tada mogu pružiti vrijedne povratne informacije razvojnim timovima. Međutim, s korisnikove točke gledišta najbolji način da se utvrdi hoće li sustav zadovoljiti njegove zahtjeve je praktična uporaba sustava. (Gomaa i Scott, 1981, str. 334).

3.5. Prototipski pristup specificiranju zahtjeva

Komunikacijski jaz između razvojnog tima sustava i korisnika može se premostiti razvojem prototipa predloženog sustava. S prototipom, korisnik može koristiti sustav baš kao da već radi u njegovom vlastitom okruženju, i na taj način dati vitalne povratne informacije razvojnom timu o prikladnosti specifikacije.

Gomaa i Scott (1981, str. 334) navode da druge grane inženjeringa često koriste prototipove i tehnike modeliranja prije stavljanja složenih sustava u proizvodnju. Izrada prototipova također može pomoći procesu softverskog inženjeringa stvaranjem aplikacijskih sustava koji su više orijentirani na korisnika.

Gomaa i Scott (1981, str. 334) navode da je dobro vrijeme za razvoj prototipa nakon što je razvijena preliminarna verzija specifikacije zahtjeva. Također, navode da na toj razini programer vjeruje da ima dostatno razumijevanje korisnikovih problema i završio je svoj prvi ozbiljni pokušaj da predloži sustav koji bi zadovoljio zahtjeve korisnika. Trošak unošenja promjena u sustav značajno raste kako razvoj softvera napreduje. Stoga je poželjno dobiti povratnu informaciju od korisnika što je ranije moguće u životnom ciklusu softvera i svakako prije finaliziranja specifikacije zahtjeva. Kako je cilj prototipa maksimizirati povratne informacije korisnika, prototip bi trebao naglasiti korisničko sučelje na trošak softvera niže razine koji nije vidljiv korisniku. Nakon što je prototip razvijen, korisnicima treba dati dovoljno prilika da ga prakticiraju i da daju svoje povratne informacije programerima. Na temelju povratnih informacija korisnika, moguće je napraviti daljnje izmjene na prototipu. Obično su te promjene relativno manje prirode.

Gomaa i Scott (1981, str. 334) navode da u određenoj fazi, koju korisnik i programer moraju pažljivo pratiti i procijeniti, prednosti daljnjih poboljšanja prototipa nadmašeni su od strane vremena i cijene potrebne za takve modifikacije. Također, navode da se u takvoj fazi potrebno oduprijeti daljnjem razvoju prototipa nakon što je nadživio svoju korisnost, a osobito se ne preporučuje dopuštanje prototipu da se razvije u željeni sustav za isporuku, jer se pristup softverskog inženjeringa potreban za razvoj proizvodnog sustava uvelike razlikuje od pristupa potrebnog za razvoj prototipa. Stoga je održavanje mnogo važnija stvar u razvoju proizvodnog sustava nego održavanje prototipa.

Na temelju povratnih informacija korisnika, specifikacija zahtjeva je revidirana. Ova revidirana specifikacija čini osnovu od koje polazi dizajn i implementacija sustava. Prototip se također može koristiti za obuku korisnika o tome kako koristiti sustav. Međutim, to zahtijeva ažuriranje prototipa. Prednosti ovog pristupa moraju se odvagnuti u odnosu na dodatne troškove održavanja prototipa. („Interaction Design Foundation [IDF]“, bez dat.).

Izrada prototipa smatra se komplementarnom drugim alatima za analizu i određivanje zahtjeva korisnika. Stoga se prototip može koristiti kako bi se utvrdilo da predloženi sustav zaista zadovoljava zahtjeve korisnika, tj. kako bi se osiguralo da je specifikacija zahtjeva potpuna i točna. Jezik specifikacije zahtjeva tada se može koristiti za specificiranje korisničkih zahtjeva, osiguravajući da je specifikacija nedvosmislena i dosljedna (Gomaa i Scott, 1981, str. 335).

4. Praktični primjer korištenja prototipa kao specifikacijskog sredstva

4.1. O projektu i programskom alatu Figma

Praktični dio ovog rada sastoji se od razvoja prototipa aplikacije pod nazivom „factoryPC“. Korištenjem navedenog prototipa i prikazom korisničkog sučelja kroz dvije studije slučaja te dane probleme, biti će prikazani odgovori, odnosno rješenja za dane probleme.

„factoryPC“ je mobilna aplikacija namijenjena za korištenje od strane kupaca računalnih komponenti, kao što su procesorske jedinice, matične ploče, grafičke kartice, kućišta, hladnjaci za komponente, napajanja, radna memorija i dijelovi za pohranu podataka. „factoryPC“ služi kao posrednik između kupaca i trgovine „factoryPC“ čiji administratori na aplikaciju mogu objavljivati asortiman koji nude. Drugim riječima, ova aplikacija zapravo prikazuje artikle prodavača koji svojom ponudom žele privući kupce na jedno mjesto, a u isto vrijeme kupcima olakšava pretraživanje računalnih komponenti. Kupci računalnih komponenti imat će bolji pregled cijena i raznolikog asortimana koji je u ponudi na jednom mjestu. Aplikacija „factoryPC“ potpuno je novo softversko rješenje osmišljeno i usmjereno na pružanje potpore potrošačima, a pridodalo bi mnogobrojne funkcionalnosti te bi korisnicima olakšalo pregled i pretraživanje računalnih komponenti. Dugoročni je cilj razvitak kvalitetnog izgleda aplikacije gdje će prodavači moći objavljivati asortiman koji nude, a koji će kupci htjeti kupiti.

Tehnika koja će se koristiti pri izradi prototipa je tehnika izrade prototipa visoke razine vjernosti prikaza pomoću programskog alata Figma koji je besplatan i javno dostupan putem interneta. Skup značajki navedenog alata usredotočen je na dizajn korisničkog sučelja i korisničkog iskustva koristeći razne alate za uređivanje vektorske grafike i izradu prototipova. Napravljeni prototip nastoji biti što sličniji krajnjem proizvodu koji će biti isporučen te sadrži veću razinu detalja, funkcionalan je i interaktivan. Korištenjem programskog alata Figma omogućen je realističan i detaljan vizualni dizajn, a svi elementi sučelja, razmaci i grafika ukomponirani su u cjelinu.

Korištenjem programskog alata Figma te pregledom prototipa izgrađenog u tom alatu nije moguće ostvariti pregled nad određenim nefunkcionalnim zahtjevima, poput performansi koje su zapravo određene optimalnim korištenjem programskog koda pri izradi pravoga proizvoda, a cilj koji se želi postići dobrim performansama sustava je kraći vremenski interval procesuiranja podataka te što kraće vrijeme odziva na radnje korisnika. Također, postoji i sigurnost, a za korištenje aplikacije potrebno se validirati unosom ispravne kombinacije

korisničkog imena i lozinke u svrhu zaštite korisničkih podataka i informacija koje je korisnik samom registracijom predao sustavu od neovlaštenog pristupa.

Međutim, programski alat Figma odličan je za prikaz dizajna sustava te za prikaz dostupnosti i izgleda određenih funkcionalnih zahtjeva. Korisničko sučelje mora biti prilagođeno potrebama korisnika, neovisno o statusu i ulozi, a dizajn mora biti intuitivan i prilagođen korisniku kako bi se krajnji korisnik mogao što jednostavnije snaći. Korisnik u svakom trenutku mora znati što će neka tipka ili pritisak na ikonu napraviti. Interaktivni prototip dostupan je putem sljedeće poveznice: Praktični rad.

4.2. Problem aplikacije iz perspektive kupca

Podrazumijeva se da je glavni cilj rada i korištenja aplikacije od bilo koje strane korisnika uvjetovan danim zahtjevima za izradu samog softvera i njihovom analizom. Svaka bi aplikacija trebala biti kvalitetno izrađena, prateći neke suvremene trendove za izradu dizajna i funkcionalnosti koji su zadani zahtjevima. Kvalitetnom analizom i specifikacijom zahtjeva, razvojni tim ima mogućnost dati odgovor i rješenja na dane zahtjeve. Daljnjim interakcijama korisnika aplikacije i razvojnog tima uz pomoć nekog modela aplikacije, kao što je prototip, poboljšava se faza razvoja pravog proizvoda.

Objekt ove studije je izrada prototipa koji odgovara danim specifikacijama zahtjeva te njegove karakteristike i značaj za izradu glavnog proizvoda.

Svrha ove studije je istražiti važnost korištenja prototipa prilikom specifikacije zahtjeva te koliko je prototip dobar kao model koji se može koristiti kao specifikacijsko sredstvo.

Ova studija usmjerena je na jednu perspektivu, onu iz gledišta korisnika aplikacije koji bi ju koristio kao kupac računalnih dijelova.

Korisnik kao kupac je uloga u aplikaciji koja je određena statusom, ovisno o tome je li kupac registriran na aplikaciji ili nije. Korisnik ima mogućnost prijave u aplikaciju korištenjem određenih vrsta autentikacije, mogućnost uređivanja vlastitog profila, pregleda trenutnih stavki košarice i mogućnost pregleda prošlih kupovina. Korisnik kao kupac također ima mogućnost pretraživanja artikala, pregleda njihovih specifikacija te kompatibilnosti s ostalim komponentama i dodavanja odabranih artikala u košaricu i kupnje istih.

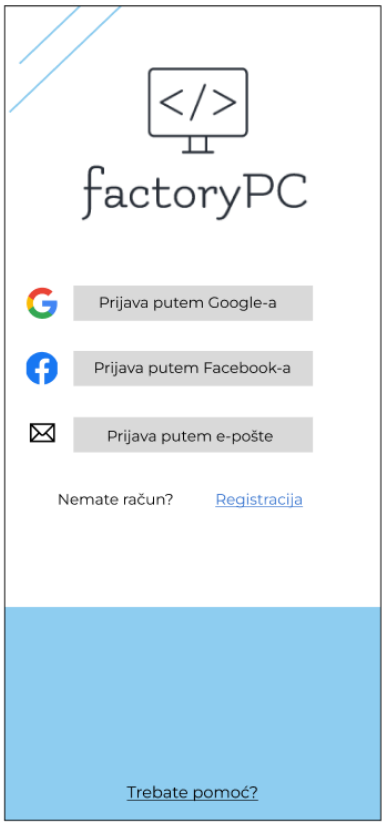
Korisničko sučelje važan je dio svakog softvera, a opći zahtjevi za korisničko sučelje su da bude jednostavan za rukovanje, brz u odgovoru, učinkovit u rješavanju operativnih pogrešaka te pružanje jednostavnog, ali dosljednog korisničkog sučelja. Prihvaćanje korisnika uvelike ovisi o tome kako korisnik može koristiti sustav pa je tako jedini način da korisnik percipira sustav. Aplikacija „factoryPC“ mora biti opremljena atraktivnim, jasnim, dosljednim i

brzim korisničkim sučeljem te mora pružiti sredstva za njegovo učinkovito korištenje. U određivanju funkcionalnih zahtjeva, korisničko sučelje mora jasno prezentirati određeni sadržaj, jednostavnu navigaciju te mora sadržavati dosljedne elemente i kvalitetnu uporabu boje i teksture. Najbitnije od svega je da bude usmjeren na korisnika.

Odgovori na funkcionalne zahtjeve biti će prikazani u obliku visokokvalitetnih skica napravljenog prototipa.

Prvi funkcionalni zahtjev je napraviti i dizajnirati mogućnost registracije i prijave korisnika u aplikaciju. Ukoliko korisnik ima namjeru izvršiti glavnu svrhu aplikacije, a to je kupovina proizvoda, morat će se registrirati u aplikaciju. Jednom kada je korisnik registriran i u sustavu aplikacije, bit će mu omogućena kupovina proizvoda. Prilikom registracije korisnik mora unijeti podatke o imenu, prezimenu, adresi e-pošte te odrediti vlastitu lozinku, koja će služiti pri budućim korištenjima prilikom prijave u aplikaciju. Nakon registracije korisnika, potrebno se prijaviti u aplikaciju. Prilikom prijave korisnika koji je registriran, korisnik mora unijeti adresu e-pošte i lozinku koju je postavio prilikom registracije. Ukoliko se korisnik ne želi prijaviti putem e-pošte, potrebno mu je omogućiti prijavu putem aplikacija treće strane, poput Google-a i Facebook-a. Prilikom prijave potrebno je omogućiti postavljanje nove lozinke ukoliko je stara lozinka zaboravljena. Nakon uspješno odrađenih aktivnosti registracije, prijave ili promjene lozinke potrebno je obavijestiti korisnika o uspješnosti iste.

Slika 6 prikazuje zaslon u kojemu korisnik ima mogućnost odabira želi li se prijaviti u aplikaciju, ili ukoliko nema račun, želi li se registrirati. Pritiskom na tipku „Registracija“ otvara se zaslon u kojemu je potrebno unijeti podatke, poput: imena, prezimena, adrese e-pošte, lozinke i potvrdne lozinke. Slika 7 prikazuje zaslon za popunjavanje podataka te kako taj zaslon izgleda jednom kada su podaci uneseni u formu za registraciju. Ukoliko je registracija uspješna, pojavljuje se obavijest u obliku pop-up prozora koja korisnika obavještava o uspješnosti registracije. Pozadina te obavijesti je zamućena kako bi se stvorio fokus na ono bitno, a to je potvrda uspješnosti registracije.



Slika 6. Zaslon za odabir prijave ili registracije (autorski rad)

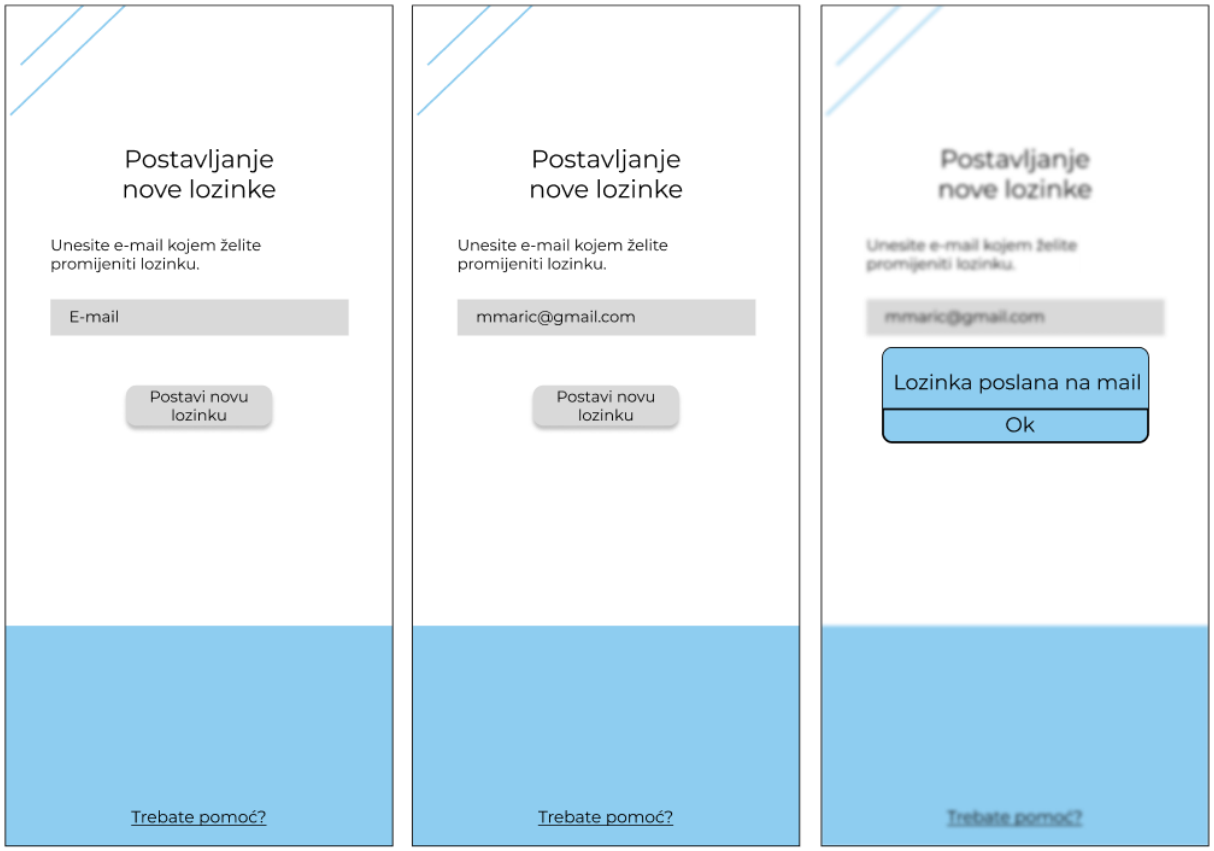
Slika 7. Zasloni za popunjavanje podataka prilikom registracije (autorski rad)

Slika 8 prikazuje zaslon koji se pojavljuje prilikom procesa prijave u aplikaciju. Na ovom su zaslonu prikazane mogućnosti unosa podataka za prijavu ili, ukoliko je korisnik zaboravio postavljenu lozinku, mogućnost pritiska tipke koja vodi na zaslon za promjenu lozinke.

Slika 8. Zaslon za prijavu u aplikaciju (autorski rad)

Slika 9 prikazuje zaslon na kojemu je moguće promijeniti postavljenu lozinku. Da bi se ova aktivnost izvršila, korisnik prvo mora unijeti adresu e-pošte kojom se registrirao na

aplikaciju. Nova lozinka je poslana na korisnikovu adresu e-pošte. Jednom kada je korisnik upisao adresu e-pošte i pritisnuo tipku „Postavi novu lozinku“ otvorena mu je mogućnost ponovne prijave unosom adrese e-pošte i lozinke koje se nalaze u sustavu aplikacije. Također, pritiskom na tipku „Postavi novu lozinku“, uz uvjet da se adresa e-pošte zaista nalazi u sustavu aplikacije, pokazuje se obavijest o uspješnosti postavljanja nove lozinke.



Slika 9. Zaslon za postavljanje nove lozinke (autorski rad)

Slika 10 prikazuje zaslone unosa podataka u zaslonu za prijavu korisnika te zaslon na koji je moguće doći pritiskom na tipku „Prijavi se“. Ukoliko je korisnik upisao točnu lozinku i adresu e-pošte pritiskom na tipku „Prijavi se“ otvorit će se novi prozor početne stranice u kojemu je zamućena pozadina, a fokusu zaslona prikazana je obavijest o uspješnosti prijave u aplikaciju.

Slika 10. Zaslon uspješne prijave u aplikaciju (autorski rad)

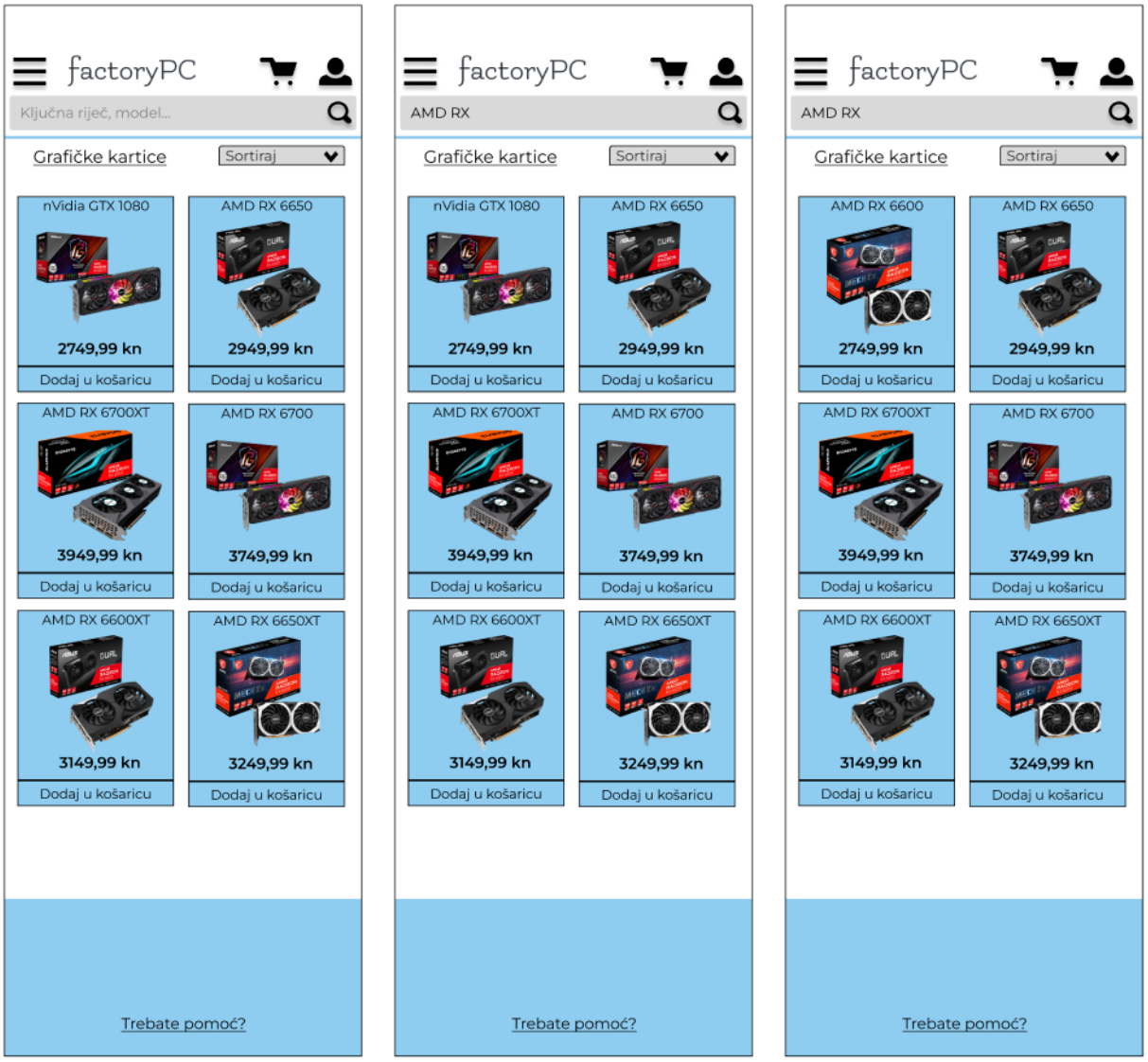
Drugi funkcionalni zahtjev je napraviti i dizajnirati mogućnost pretraživanja artikala po ključnoj riječi ili imenu modela, sortiranja po imenu i cijeni silazno ili uzlazno. Potrebno je postaviti intuitivan raspored elemenata koji služe za izvršavanje zadanih zahtjeva. Korisnik mora imati mogućnost pregleda tehničkih informacija o dostupnom artiklu te dodavanja istog u košaricu. Korisnik mora imati pravo na transparentan prikaz cijena proizvoda te imati mogućnost na prikaz intuitivne navigacijske trake. Zahtjev je potreban kako bi krajnji korisnik imao mogućnost pregleda dostupnih proizvoda i informacija o njima. Zahtjev je ispunjen ukoliko korisnik može iskoristiti funkcionalnosti pretraživanja, sortiranja, pregleda informacija o proizvodu te dodavanja proizvoda u košaricu.

Slika 11 prikazuje početni zaslon aplikacije nakon što se korisnik u nju prijavi. Početni zaslon prikazuje proizvode preporučene od strane administratora aplikacije, a korisnik ima mogućnost pregleda navigacijske trake, košarice, svog profila i određenog odabranog artikla. Drugi zaslon prikazuje ekran nakon što korisnik pritisne tipku za meni u gornjem lijevom kutu, a u fokusu je navigacijska traka kojoj je pozadina zamućena. Korisnik ima mogućnost odabrati određenu računalnu komponentu kako bi filtrirao i smanjio vrijeme traženja određenog proizvoda.

Slika 11. Početni zaslon i navigacijska traka (autorski rad)

Slika 12 prikazuje na koji je način odrađen proces pretraživanja proizvoda po ključnoj riječi. Aplikacija treba biti u mogućnosti pretražiti dostupne proizvode na temelju upisane

ključne riječi te takve proizvode izlistati na zaslonu. Upisivanjem ključne riječi u traku za pretraživanje te pritiskom na tipku za pretraživanje u obliku povećala aplikacija izvršava upit te na zaslon izlistava proizvode koji u svom nazivu sadrže upisanu jednu ili više riječi.



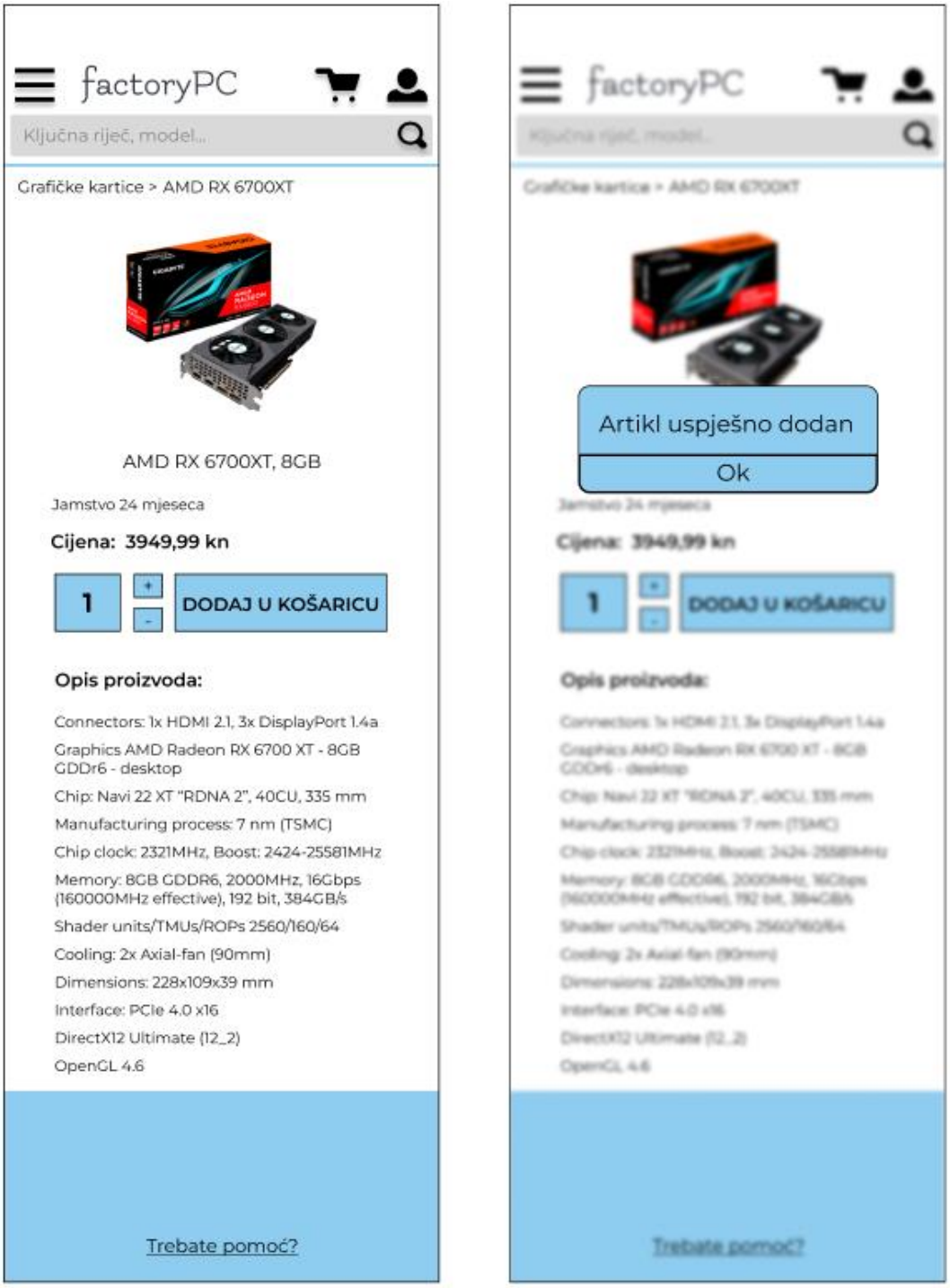
Slika 12. Funkcionalnost pretraživanja proizvoda po ključnoj riječi (autorski rad)

Slika 13 prikazuje na koji je način odrađen proces sortiranja proizvoda po njihovoj cijeni, uzlazno ili silazno. Pritiskom na tipku za sortiranje, korisnik ima opciju sortiranja proizvoda po cijeni uzlazno ili silazno, a aplikacija će upit obraditi prikazom proizvoda na način koji je odabrao korisnik.

Slika 13. Funkcionalnost sortiranja proizvoda (autorski rad)

Slika 14 prikazuje na koji je način odrađen proces pregleda informacija o odabranom proizvodu te kako je dizajniran zaslon dodavanja proizvoda u košaricu. Prilikom pregleda informacija o proizvodu, korisniku je prikazana navigacija za povratak na prošli zaslon, slika proizvoda, ime proizvoda, duljina jamstva te cijena proizvoda. Korisnik ima mogućnost pregleda i mijenjanja količine proizvoda koju želi staviti u košaricu jednostavnim pritiskom na tipku „+“ ili „-“. Donji dio zaslona prikazuje neke osnovne informacije o proizvodu, poput

dostupnih konektora ili radne memorije ukoliko je riječ o grafičkoj kartici. Pritiskom na tipku „Dodaj u košaricu“ u košaricu se dodaje odabrana količina prikazanog proizvoda uz obavijest o uspješnosti da je artikl uspješno dodan u košaricu.



Slika 14. Dodavanje proizvoda u košaricu (autorski rad)

Posljednji funkcionalni zahtjev je napraviti i dizajnirati mogućnost kupovine proizvoda koji se nalaze u košarici. Potrebno je napraviti intuitivan prikaz košarice kako bi se korisnik mogao što lakše kretati prilikom kupnje proizvoda. Korisnik mora imati mogućnost pregleda proizvoda u košarici, uklanjanja proizvoda iz košarice te ponovnog ispravljanja željene količine određenog proizvoda. Također, na zaslonu moraju biti prikazane cijene proizvoda, kao i sveukupna cijena košarice. Korisnik mora imati mogućnost unosa podataka za dostavu željenih proizvoda te odabira načina plaćanja i pregled cijene dostave. Zahtjev je ispunjen ukoliko korisnik može naručiti odabrane proizvode koji se nalaze u njegovoj košarici, upisati podatke za dostavu te odabrati način plaćanja i dostavu.

Slika 15 prikazuje izgled košarice korisnika u kojoj su prikazane stavke koje je korisnik sam dodao u košaricu. Na tom zaslonu korisnik ima mogućnost ažuriranja količine željenog proizvoda, uklanjanja proizvoda iz košarice i dovršavanja kupovine pritiskom na tipku „Dovrši kupovinu“. Na zaslonu su prikazane cijene pojedinačnih proizvoda ovisno o njihovoj količini te sveukupna cijena košarice.

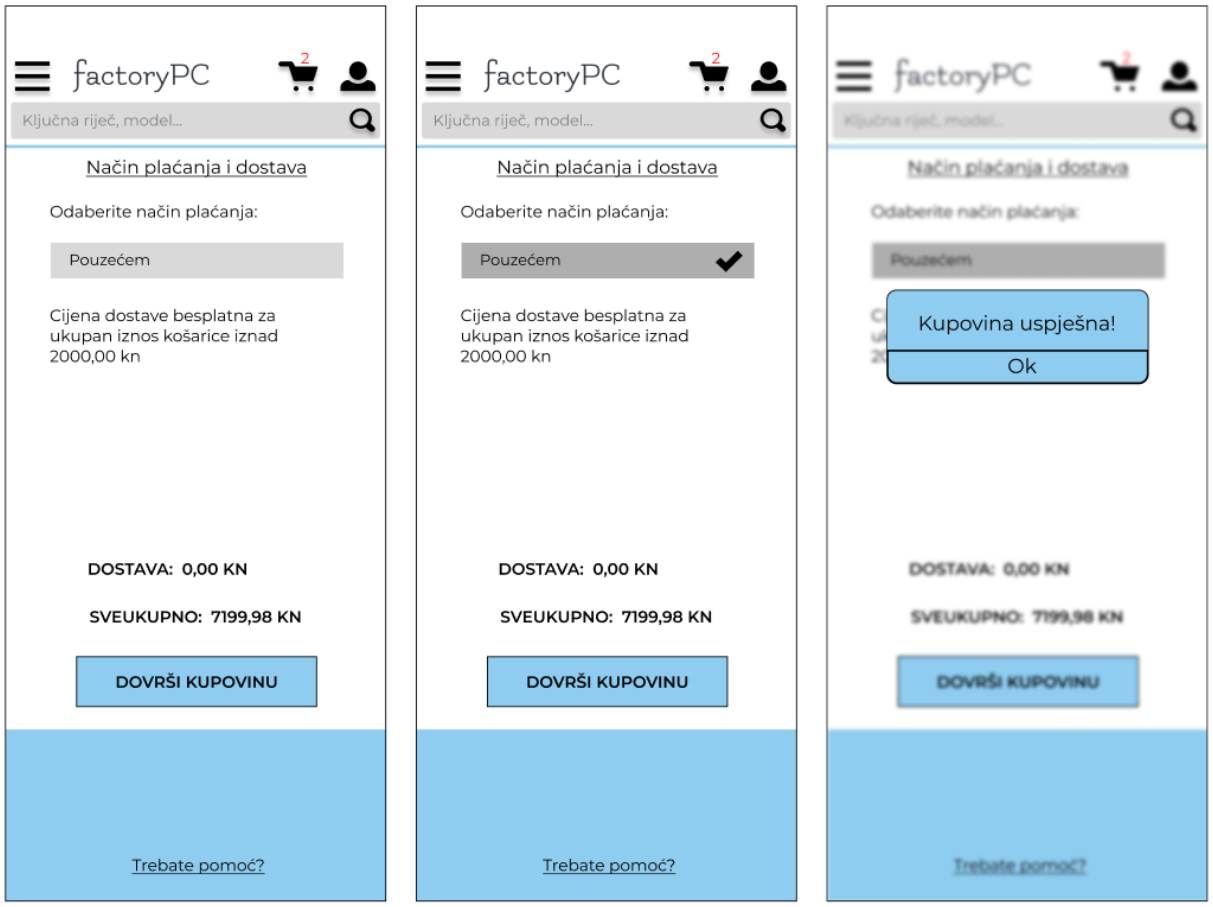
Slika 15. Košarica (autorski rad)

Slika 16 prikazuje zaslon na kojemu korisnik ima mogućnost unosa podataka za dostavu nakon što je na prethodnom zaslonu pritisnuo tipku „Dovrši kupovinu“. Podaci koji se moraju unijeti su mjesto, poštanski broj, adresa i država, a napomena je opcionalna. Na zaslonu je prikazana i sveukupna cijena košarice.

Slika 16. Unos podataka za dostavu (autorski rad)

Slika 17 prikazuje zaslone u kojima korisnik ima mogućnost odabira željenog načina plaćanja i dostave proizvoda. Na zaslonima je vidljiva cijena dostave i sveukupna cijena kupovine računajući cijenu dostave. Pritiskom na neki od ponuđenih načina plaćanja i dostave,

prikazani se odabir označuje kvačicom te je korisnik sada u mogućnosti pritisnuti tipku „Dovrši kupovinu“. Ukoliko je korisnik odabrao način plaćanja i dostave, pritiskom na tipku „Dovrši kupovinu“ prikazuje se obavijest o uspješnosti kupovine.



Slika 17. Način plaćanja i dovršetak kupovine (autorski rad)

Prikazanim zaslonima i izradom interaktivnog prototipa visoke razine vjernosti prikaza dana su rješenja na zadane zahtjeve. Korištenjem interaktivnog prototipa moguće je utvrditi tijek izvršavanja određenih zadataka te dobiti povratne informacije korisnika. Interakcijom korisnika za čiju je upotrebu izrađen ovaj prototip može se utvrditi jesu li zadovoljeni specificirani zahtjevi.

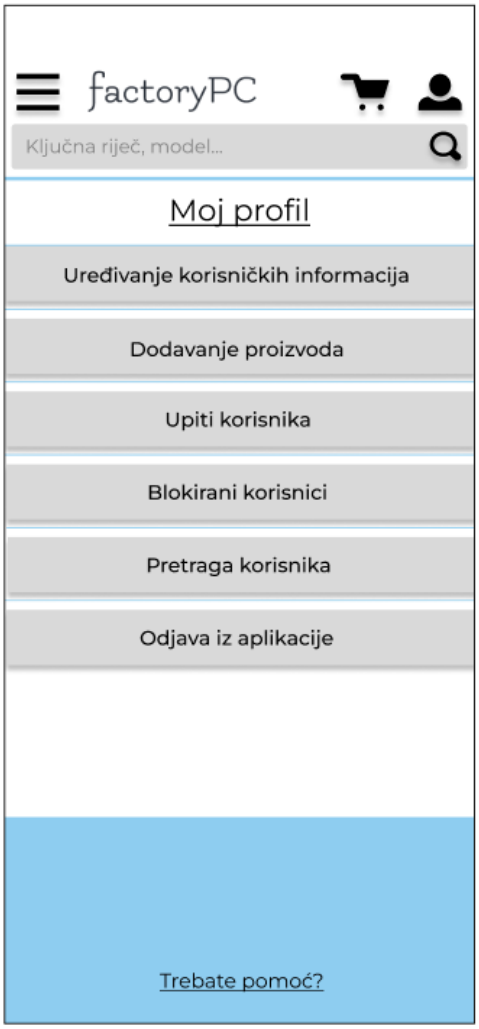
4.3. Problemi aplikacije iz perspektive moderatora

Objekt ove studije je izrada i prikaz prototipa koji odgovara danim specifikacijama zahtjeva te njegove karakteristike i značaj za izradu glavnog proizvoda.

Svrha ove studije je istražiti važnost korištenja prototipa prilikom specifikacije zahtjeva te koliko je prototip dobar kao model koji se može koristiti kao specifikacijsko sredstvo.

Ova studija usmjerena je na jednu perspektivu, onu iz gledišta moderatora aplikacije koji bi ju koristio u svrhu dodavanja ili brisanja proizvoda s aplikacije te kao tehnička podrška kupcima.

Prvi funkcionalni zahtjev za ulogu moderatora je napraviti mogućnost dodavanja novih računalnih komponenti kako bi ih kupci mogli kupiti. Prilikom kreiranja novog artikla, moderator izabire kategorije kojoj komponenta pripada, prilaže sliku ukoliko je dostupna te nakon toga unosi opis proizvoda u kojemu navodi osnovne informacije o proizvodu ovisno o njegovoj kategoriji, primjerice, naziv proizvoda, naziv proizvođača, dimenzije proizvoda i sve ostalo što je potrebno kako bi opis proizvoda bio upotpunjen i kako bi ga kupac mogao što jednostavnije razumjeti. Naravno, svaka kategorija komponenata mora imati različite atribute kojima se može opisati.



Slika 18. Profil moderatora (autorski rad)

Moderator ima isti pregled na početnu stranicu kao i običan korisnik. Međutim, ovlasti moderatora omogućuju mu neke dodatne mogućnosti. Pritiskom na tipku za otvaranje profila korisnika u gornjem desnom uglu na bilo kojem zaslonu, otvara se profil trenutnog korisnika. Slika 18 prikazuje zaslon profila trenutno prijavljenog korisnika, u našem slučaju moderatora. Moderator za razliku od običnog korisnika ima ovlasti dodavanja proizvoda u aplikaciju, pregleda upita poslanih od strane korisnika, pregled blokiranih korisnika te mogućnost pretraživanja korisnika.

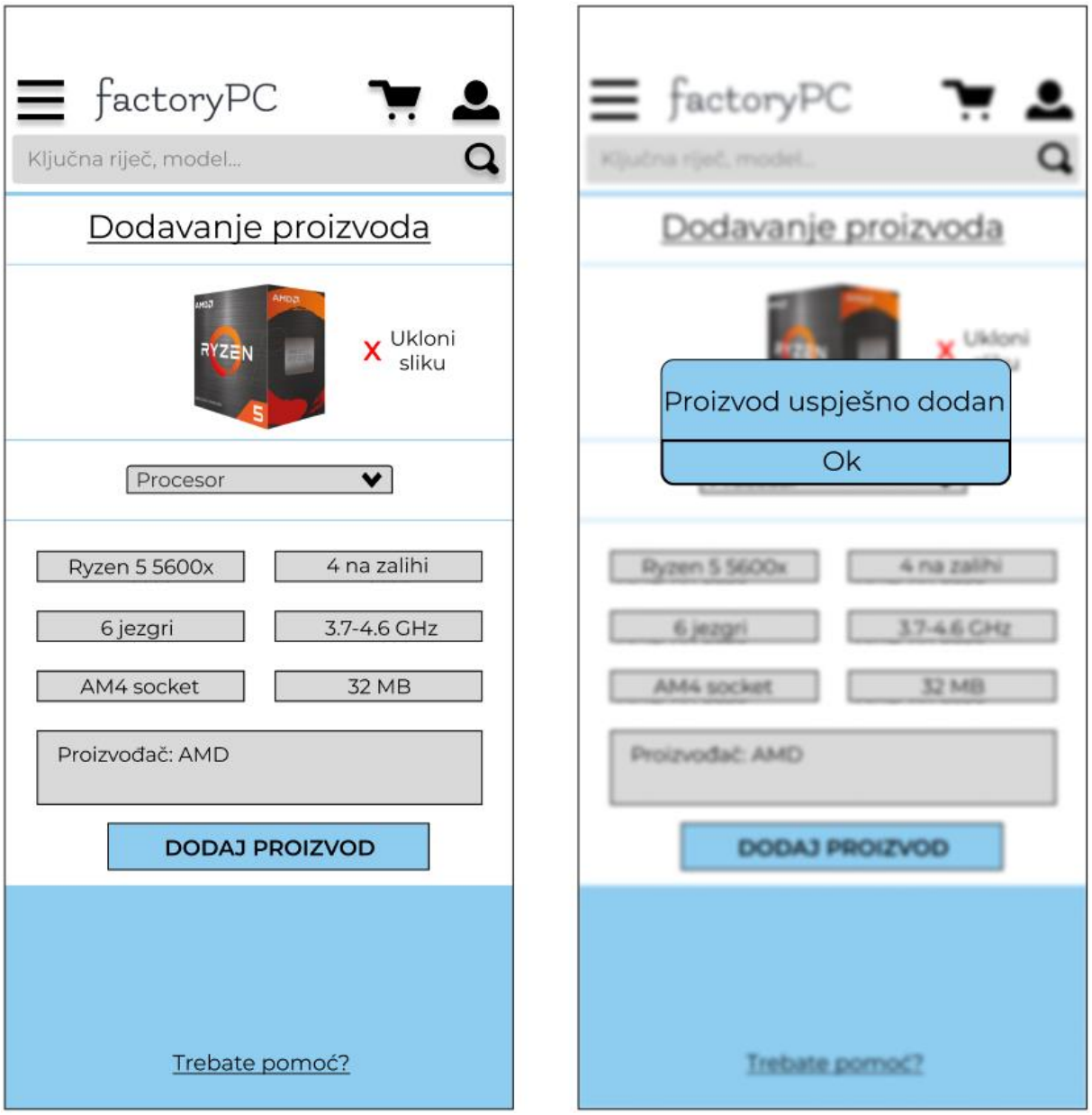
Slika 19 prikazuje zaslon koji se pojavi ukoliko moderator na vlastitom profilu pritisne tipku „Dodavanje proizvoda“. Moderator na početku ima mogućnost dodavanja fotografije proizvoda te odabir kategorije proizvoda. Pritiskom na padajući izbornik za odabir kategorije, nude se kategorije računalnih komponenti, poput procesora, grafičke kartice, pohrane podataka, radne memorije i napajanja.

Slika 19. Odabir kategorije proizvoda prilikom dodavanja (autorski rad)

Slika 20 prikazuje što se dogodi kada moderator odabere neku od ponuđenih kategorija računalnih komponenti. Nakon što je odabrana kategorija proizvoda kojeg moderator želi dodati u aplikaciju, pojavljuju se polja za unos podataka za proizvod kojeg moderator želi dodati. Na zaslonu je prikazano da je odabrana kategorija procesor pa se sukladno odabranoj kategoriji pojavljuju polja za unos koja sadrže neke osnovne podatke o procesoru, poput polja za unos imena procesora, količine na zalihi, broja jezgri, radnog takta procesora, procesorskog ležišta (engl. *CPU socket*), količine predmemorije i polje za unos dodatnih informacija.

Slika 20. Prikaz polja za unos podataka o proizvodu (autorski rad)

Slika 21 prikazuje zaslon na kojemu su uneseni podaci o proizvodu, a samim popunjavanjem podataka omogućen je pritisak na tipku „Dodaj proizvod“ čijim se pritiskom proizvod dodaje u aplikaciju i postaje vidljiv običnom korisniku. Dodavanjem proizvoda u aplikaciju moderatoru je prikazana obavijest o uspješnom dodavanju proizvoda u aplikaciju.

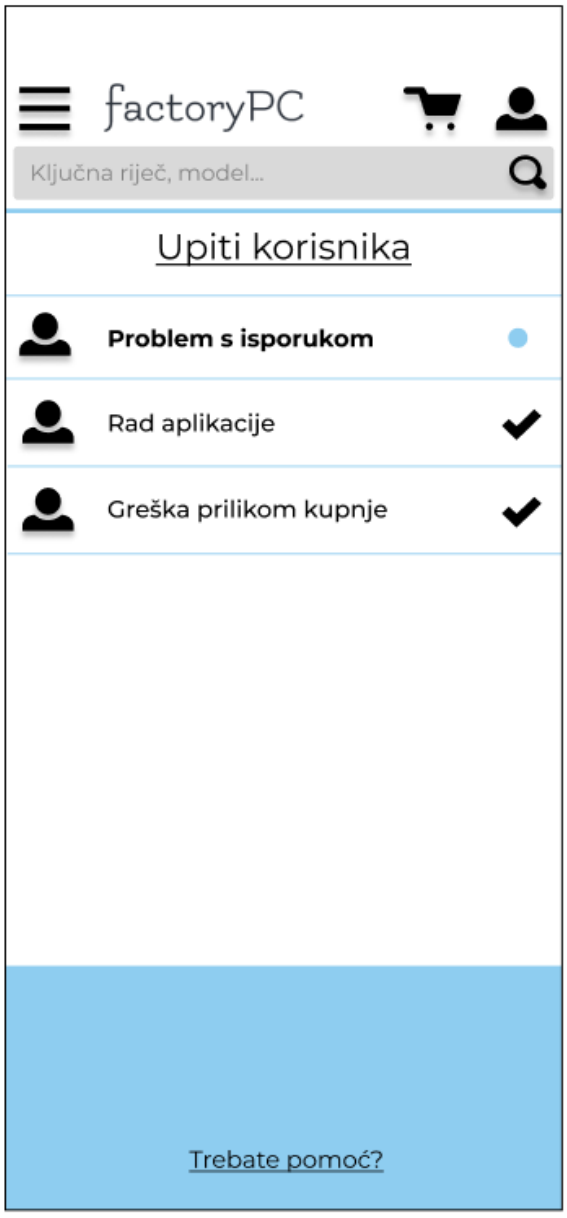


Slika 21. Dodavanje proizvoda u aplikaciju (autorski rad)

Moderator sustava pokušava kupcima olakšati sveukupno korištenje aplikacije. U bilo kojem trenutku pri kupovini proizvoda, kupac je u mogućnosti poslati upit pomoći moderatoru. Moderator ima uvid u popis svih upita koje mu je sustav dodijelio. Odabirom određenog upita,

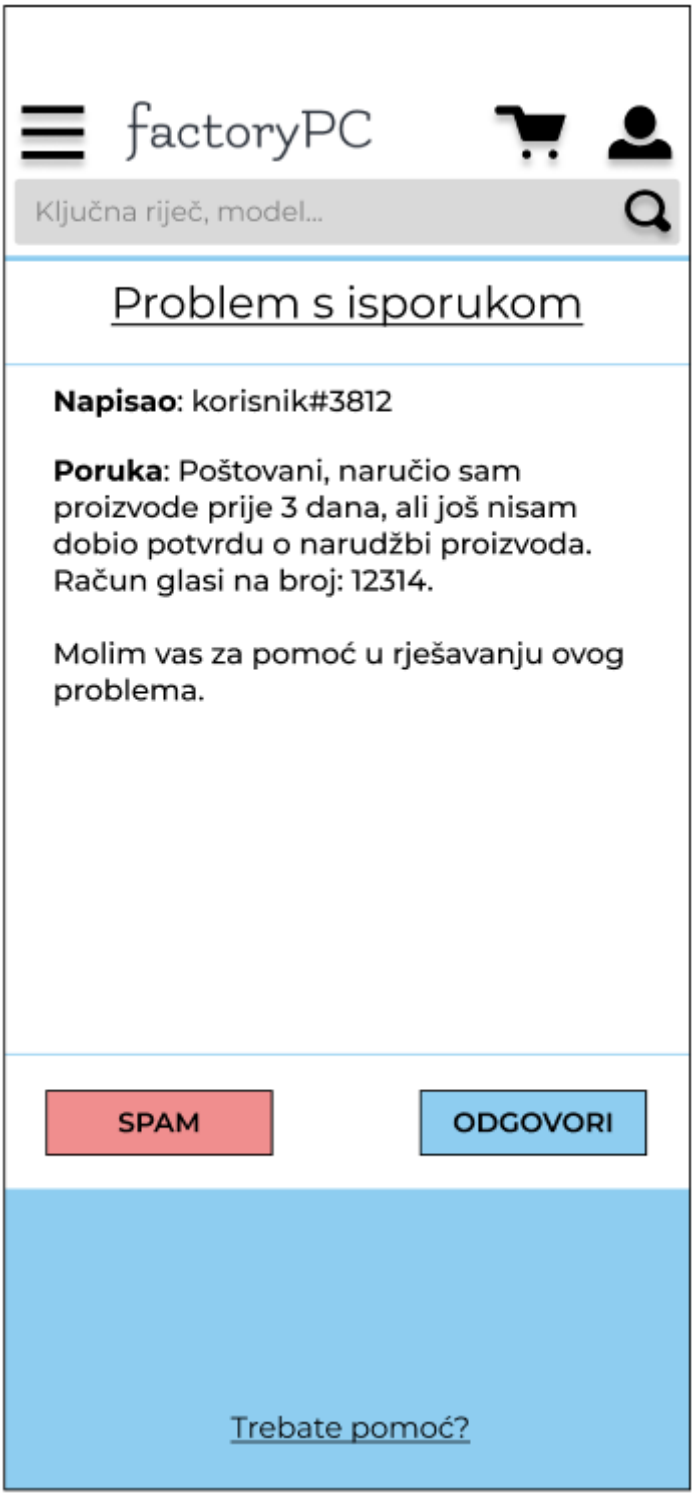
moderatoru se otvara mogućnost odgovaranja korisniku koji je postavio upit, a ponuđena je i mogućnost svrstavanja upita pod spam. Odgovor moderatora dolazi na adresu e-pošte kupca koji je poslao upit.

Slika 22 prikazuje zaslon s upitima korisnika, a tom zaslonu može se pristupiti sa zaslona vlastitog profila pritiskom na tipku „Upiti korisnika“. Moderatoru su vidljivi svi upiti korisnika, a njihov status o tome jesu li dani odgovori na upite vide se pomoću drugačijeg prikaza elemenata koji čine jedan upit. Naslov upita koji nije pročitan niti odgovoren podebljan je, a s desne strane vidljiva je ikona koja signalizira da je upit na čekanju. Upiti na koje je odgovoreno s desne strane imaju ikonu kvačice.



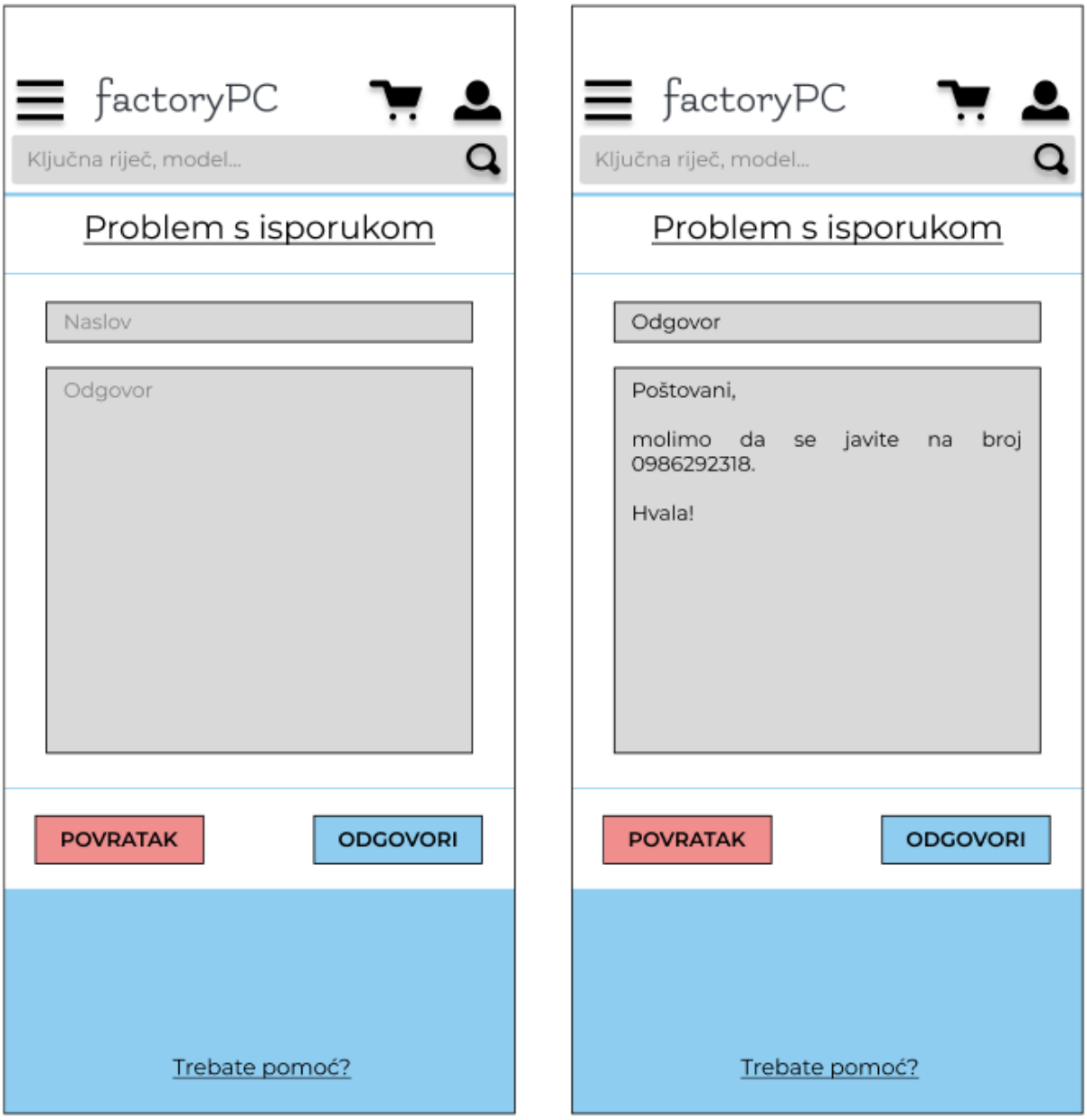
Slika 22. Upiti korisnika (autorski rad)

Slika 23 prikazuje zaslon za odgovaranje korisniku koji se prikaže ukoliko se na zaslonu s upitima korisnika pritisne na željeni upit. Moderator ima mogućnost svrstavanja upita pod spam ili odgovaranja na njih. Moderatoru je vidljiv naslov upita te poruka koju je napisao korisnik.



Slika 23. Poruka upita (autorski rad)

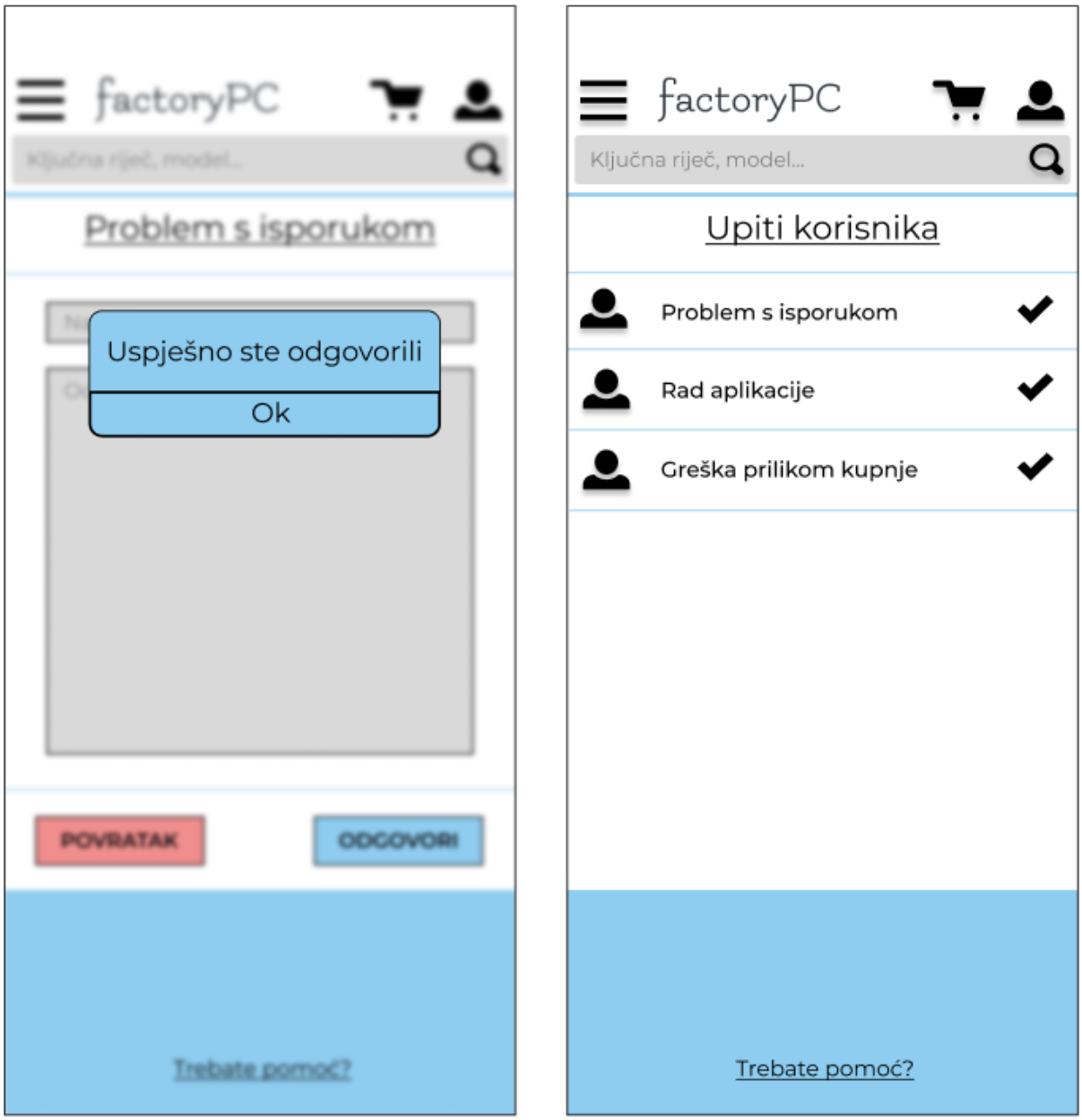
Slika 24 prikazuje zaslone na kojima su prikazane opcije ukoliko moderator odluči odgovoriti na upit korisnika. Moderatoru je dostupno polje za unos vlastitog naslova te polje za unos poruke. Ukoliko moderator ipak odluči ne odgovoriti na upit, ima mogućnost pritiska tipke „Povratak“ koja ga vodi na prethodni zaslon.



Slika 24. Odgovor na upit (autorski rad)

Slika 25 prikazuje zaslone na kojima su prikazane opcije ukoliko moderator pritisne tipku „Odgovori“. Ukoliko je korisnik uspješno odgovorio na upit dobit će obavijest o uspješnosti

odgovora. Sljedeći zaslon prikazuje novi poredak upita korisnika gdje je vidljiva kvačica pored zadnjeg odgovorenog upita.



Slika 25. Uspješan odgovor na upit i prikaz upita korisnika (autorski rad)

Prikazanim zaslonima vidljiva su i rješenja za dane zahtjeve. Prikazom i korištenjem interaktivnog prototipa moguće je utvrditi jesu li zahtjevi na prototipu zadovoljeni i hoće li biti potrebe za revizijom.

5. Zaključak

Izrada prototipa nekog proizvoda ili sustava može biti jako kompleksna i dugotrajna, ali također može biti fleksibilna i kratkotrajna. Kompleksnost prototipiranja uvelike ovisi o veličini glavnog projekta i o odabranoj tehnici i tipu prototipiranja. Prototipiranje pomaže u određivanju i shvaćanju procesa specifikacije zahtjeva, a time i samom razvoju aplikacije. Izradom prototipa omogućuje se veće uključivanje krajnjih korisnika u sam projekt. Interakcijom korisnika i prototipa mogu se dobiti važne informacije potrebne za daljnji razvoj proizvoda ili sustava. Prototip nekog proizvoda ili sustava zapravo je jedini način na koji korisnik može percipirati izgled, dizajn i korištenje funkcija glavnog proizvoda ili sustava. Međutim, prototipovi nekad nisu najbolja opcija za izradu, a to ovisi o vremenu potrebnom za izradu glavnog projekta i o tome hoće li kupac proizvoda ili sustava financirati izgradnju prototipa. Prototipiranje je značajno za testiranje ideja i razumijevanja potreba krajnjeg korisnika, a omogućava odličan način za demonstraciju ideja korisnicima uz mogućnost brzog ponavljanja.

Uspješno napravljen prototip uz prezentaciju korisnicima uvelike može uštedjeti vrijeme i novac u procesu razvoja proizvoda ili sustava. Kako bi se prototip mogao što kvalitetnije napraviti potrebno je imati i kvalitetno napisanu specifikaciju zahtjeva. Pažljiva i dobro napisana specifikacija korisničkih zahtjeva štedi vrijeme i novac, a što je specifikacija korisničkih zahtjeva bolja, to je bolji rezultat.

U razradi ovoga završnog rada korištena je tehnika izrade prototipa visoke razine vjernosti prikaza pomoću programskog alata Figma koji je besplatan i javno dostupan putem interneta. Korištenjem programskog alata Figma omogućen je realističan i detaljan vizualni dizajn, a svi elementi sučelja, razmaci i grafika ukomponirani su u cjelinu. Izrada prototipa donosi više prednosti nego mana, a najbitnija prednost je kvalitetnija izrada glavnog proizvoda ili sustava koja je moguća uz pomoć dobivenih povratnih informacija krajnjeg korisnika. Prikazom zaslona prototipa omogućeno je vizualno percipirati izgled glavnog proizvoda te neke od njegovih funkcionalnosti.

Popis literature

Arnowitz, J., Arent, M., Berger, N. (2007). *Effective prototyping for software makers* (1. izd.). Diane Cerra.

Babich, N. (2017). *Prototyping 101: The difference between low-fidelity and high-fidelity prototypes and when to use each*. Preuzeto 10.9.2022. s https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use

Crinnion, J. (1991). *Evolutionary Systems Development, a practical guide to user of prototyping within a structured systems methodology*. New York: Plenum Press.

Duncan, W.R. (1999). *Requirements vs. specifications and other comparisons*. PM Network, 13(7), 21-22.

Esposito, E. (2018). *Low-fidelity vs. high-fidelity prototyping*. Preuzeto 9.9.2022. s https://www.invisionapp.com/inside-design/low-fi-vs-hi-fi-prototyping/

Gomaa, H. i Scott, D. B. H. (1981). *Prototyping as a Tool in the Specification of User Requirements*. U Jeffrey S., Stucki L. (ur.), *Proceedings of the 5th international conference on Software engineering (ICSE 1981)* (str. 333-342). San Diego, California, USA: IEEE Press.

GuidingCode (2022). *What are User and System Requirements in Software Engineering*. Preuzeto 11.9.2022. s https://guidingcode.com/user-and-system-requirements-in-software-engineering/

Ibragimova, E. (2016). *High-fidelity prototyping: What, When, Why and How?* Preuzeto 9.9.2022. s https://blog.prototypr.io/high-fidelity-prototyping-what-when-why-and-how-f5bbde6a7fd4

Interaction Design Foundation [IDF] (bez dat.) *Prototyping*. Preuzeto 13.9.2022. s https://www.interaction-design.org/literature/topics/prototyping

Jayasinghe, P. (2020). *Throwaway Prototyping vs Evolutionary Prototyping*. Preuzeto 8.9.2022. s https://medium.com/@pavithrajayasinghe9529/throwaway-prototyping-vs-evolutionary-prototyping-8302be3baf33

Jović, A., Frid, N., Ivošević, D. (2019). *Procesi programskog inženjerstva* (3. izd). FER repozitorij. Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb

Low vs high fidelity prototypes: a complete breakdown [Slika] (bez dat.) Preuzeto 10.9.2022. s https://www.justinmind.com/prototyping/low-fidelity-vs-high-fidelity-prototypes

Low-fidelity mockup image [Slika] (bez dat.) Preuzeto 10.9.2022. s https://www.lucidchart.com/blog/value-of-low-fidelity-mockups

Martinez, P. (2020). *What is Evolutionary Prototype?* Preuzeto 8.9.2022. s https://mockitt.wondershare.com/prototyping/evolutionary-prototyping.html

Martinez, P. (2020). *What is Throwaway Prototyping*. Preuzeto 8.9.2022. s https://mockitt.wondershare.com/prototyping/throwaway-prototyping.html

Ogedebe, P. M. i Jacob, B.P. (2012) *Software prototyping: A strategy to use when user lacks data processing experience*. Nigerija, Karu: AJSS.

Sahu, N. (2021). *Software prototyping model and phases*. Preuzeto 8.9.2022. s https://www.geeksforgeeks.org/software-prototyping-model-and-phases/

Wood, D., P., Kang, K., C. (1992). *A Classification and Bibliography of Software Prototyping*. Software Engineering Institute: Carnegie Mellon University.

Popis slika