Tallinna Ülikool
Digitehnoloogiate instituut

Veaseisundid ja nende määramine Testlio platvormil

Bakalaureusetöö

Autor: Ursula Jõers
Juhendaja: Romil Rõbtšenkov

Autor:……. ” …… ” 2017
Juhendaja:……..…… ” …… ” 2017
Instituudi direktor:……. ” …… ” 2017

Tallinn 2017

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……..	……
(kuupäev)	(autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Ursula Jõers (sünnikuupäev: 15.04.1992)

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Veaseisundid ja nende määratlemine Testlio platvormil", mille juhendaja on Romil Rõbtšenkov, säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____

(digitaalne) allkiri ja kuupäev

Sisukord

Sissejuhatus

Tarkvara loomise protsessile on loodud mitmeid erinevaid lähenemisi ja käsitlusi, neid nimetatakse tarkvara arenduse protsessi mudeliteks (ingl *software development process models*), näiteks kose mudel, agiilne mudel, spiraalmudel. Kõik need protsessid järgivad kindlaid tsükli osasid selleks, et tarkvara loomise protsess oleks edukas. Testimine on üks kuuest protsessi osast lisaks analüüsile, disainile ja arendusele (ISTQB Exam Certification, kuupäev puudub-a).

Tarkvara testimine on eelnevatest punktidest üks olulisematest, mille tähtsust tihtipeale alahinnatakse. Testimine on oluline, sest inimlik on teha vigu. Kuigi mõned vead võivad olla vähem olulised, siis teised võivad olla jällegi arenduse seisukohast kallid ja ohtlikud. Seega peabki arenduse jooksul kogu loodu osa alati üle kontrollima, sest igal osal võib olla suurem või väiksem viga sees, mis mõjutab üldpilti (ISTQB Exam Certification, kuupäev puudub-b). Tänapäeval teeb asja keerukamaks ka tõsiasi, et rakendusi kasutatakse paljude erinevate seadmete peal. Neil võib olla erinev operatsioonisüsteem või ekraani suurus ning kõiki neid kombinatsioone tuleb testida selleks, et kõik vead saaksid parandatud (Thomson, 2014).

Mobiilirakenduste testimisega abistab oma partnereid Testlio. See on Eestis loodud iduettevõte, mis aitab tõsta rakenduste kvaliteeti oma teenuse abiga. Testimist viib läbi rahvusvaheline testijate kogukond. Testlio platvorm on aga vahend, mida kasutavad testimisprojektiga seotud osapooled testitsüklite haldamiseks ja vigade raporteerimiseks.

Antud bakalaureusetöö käigus uuritakse veaseisundeid ja nende määratlemist Testlio platvormil[1]. Veaseisundid on vaated ja disainielemendid, mis on nähtavad peale seda kui rakenduses on esinenud viga, hoolimata sellest, kelle poolt viga on põhjustatud (Babich, 2016b). Teema on ajendatud autori huvist kasutajakogemuse ja veaseisundite vastu. Nimelt võttis töö autor osa Testlio platvormil toimunud tarkvara testimisest ning leidis, et selle töötamise voog on esmasel kasutamisel pigem keeruline. Näiteks puudusid otsesed abitekstid sisestusväljade juures. Seetõttu otsustaski antud töö autor uurida veaseisundeid, nende võimalusi, häid praktikaid ning kuidas muuta Testlio platvormil liikumine arusaadavamaks kasutajatele.

[1] https://testlio.com

Veaseisundid on olulised kasutajakogemuses, kuid nendest räägitakse disainimisel harva ning Testlio platvormil pole pööratud nende käsitlemisele hetkel väga suurt tähelepanu. Praegu on tekkinud olukord, kus Testlio platvormi kujundatakse ümber ning seetõttu on sobiv aeg üle vaadata ka veaseisundid platvormil. Saadud isikliku kogemuse ja ümberkujundamise võimaluse tekkimise tõttu otsustas töö autor teemat lähemalt uurida ning koostada ettepanekud veaseisundite ümberkujundamiseks antud bakalaureusetöö raames.

Eesmärk on anda ülevaade veaseisunditest ja luua kasutajakogemuse headele tavadele tuginedes negatiivsed testilood ning neid testida Testlio platvormil, tuues seejärel välja platvormi puudused ja soovitused nende lahendamiseks.

Eesmärgi saavutamiseks on püsitatud järgmised **uurimusküsimused**:

1. Millistest osadest koosneb Testlio testitsükkel ja kuidas on jaotatud erinevad tegevused rollide vahel?
2. Kuidas määrata ja testida veaseisundeid kasutajakogemusest lähtudes?
3. Millised veaseisundite puudused esinevad Testlio platvormil ja kuidas neid parandada?

Töö jaguneb kolmeks. Esimene osa on kirjanduse ülevaade ja selle käigus kirjeldatakse Testliot ja uuritakse kasutajakogemuse disaini ning veaseisundeid. Teises osas tutvustatakse töös kasutatud meetodeid ja kolmandas, praktilises, osas rakendatakse saadud teadmised Testlio platvormi veateadete analüüsiks ja ümberdisainimiseks. Selleks, et koostada terviklik pilt hetkel kasutatavatest veaseisunditest Testlio platvormil kasutatakse negatiivset testimist.

Töö tulemusena valmivad ülevaated koos puudustega ja soovitused edastatakse Testlio tootetiimidele, et uurimust saaks rakendada platvormi ümberkujundamisel.

1 Kirjanduse ülevaade

Käesolevas peatükis antakse ülevaade bakalaureusetöö jaoks vajalikust taustsüsteemist. Kõigepealt vaadeldakse Testlio ettevõtet ning seal toimuvaid testitsükli protsesse, seejärel on fookuses kasutajakogemuse disain ja erinevad veavaated. Peatüki lõpus vaadeldakse Google'i mobiilirakenduste veateadete suuniseid, mida kasutatakse töös veateadete määratlemiseks.

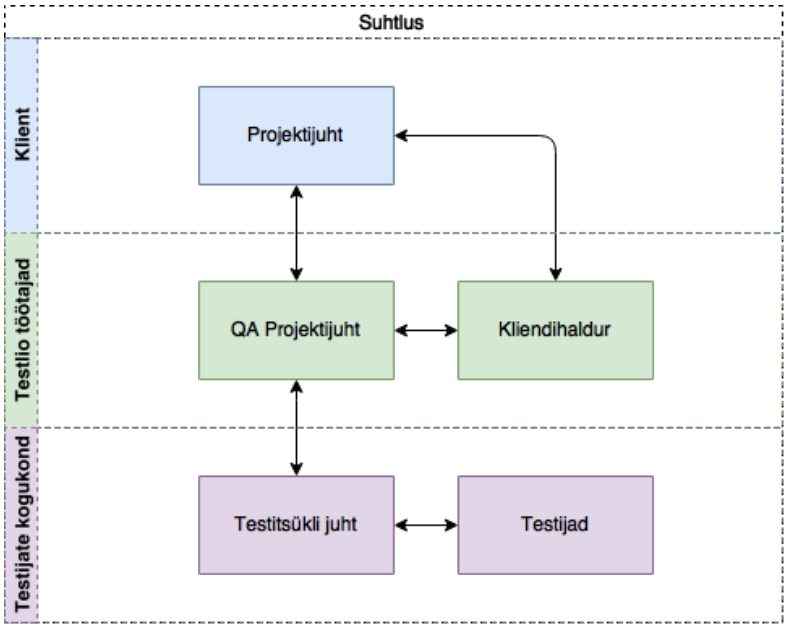
1.1 Testlio ettevõte ja testimisplatvorm

Testlio on 2012. aastal loodud iduettevõte, mille eesmärk on aidata tarkvara-arendajatel tagada loodavate mobiilirakenduste parem kvaliteet (Testlio, kuupäev puudub). Hetkel tegeleb Testlio ainult mobiilirakenduste testimisega. Selle saavutamiseks pakub firma oma klientidele teenust, milleks on testimine ja seda toetavad tegevused. Teenuse pakkumist toetab toode, mis on selles bakalaureusetöös vaadeldav platvorm.

Testlio on sündinud vastusena frustratsioonile, mille tingis 2012. aastal eksisteerinud olukord, kus testijatele maksti leitud vigade eest. Testlio asutaja Kristel Kruustük on öelnud, et tema tundis testijana töötades, et tema aega ei väärtustatud piisavalt. See tulenes sellest, et talle maksti testijana iga vea eest, mille ta leidis, kuid sealhulgas ei võetud arvesse leitud vea olulisust ega selle leidmiseks kulunud aega (Äripäev, 2016). Sellest isiklikust kogemusest on välja kasvanud Testlio tasustamise süsteem, kus testijatele makstakse kulunud tundide, mitte leitud vigade eest, mis annab testijatele võimaluse süveneda tarkvarasse.

Testlio on "testimine kui teenus" (ingl *testing-as-a-service*) platvorm. Ettevõte võtab vastutuse kogu testiplaani haldamise üle: projektide juurde värvatakse tarkvaratestijaid rahvusvahelisest testijate kogukonnast, mis on koondunud Testlio platvormile. Kliendile antakse üle valideeritud, kõrgekvaliteedilised programmivigade raportid vigade kohta, mis testimise ajal leiti (Kamps, 2016).

Testlio poolt osutatava teenuse juures on hetkel oluline faktor inimestevaheline suhtlus. Joonis 1 näitab, millised on rollid testitsüklite juures ja millised on seosed info vahetamisel.

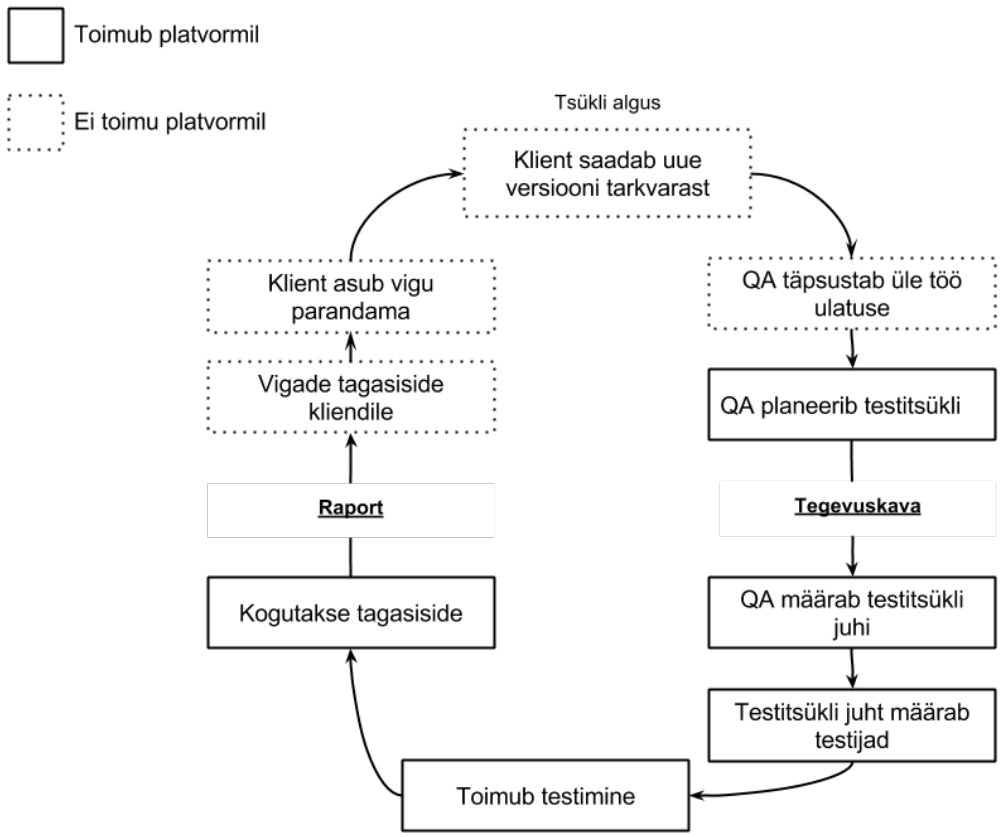


Joonis 1. Rollid ja suhtlus Testlio protssessis.

Kliendi poolt on Testliole kontaktiks isik, kes vastutab enda firmas tarkvara kvaliteedi eest. Selleks võib olla tootejuht, kvaliteedijuht või keegi kolmas. Lõputöös on valitud selleks rolliks **projektijuht**. Projektijuht suhtleb Testlio kliendihalduriga ja kvaliteedi tagamise projektijuhiga (ingl *Quality Assurance,* edaspidi lühend QA).

Järgmise grupi moodustavad Testlio töötajad, kes on testitsüklite ja kliendiga otseselt seotud. **Kliendihaldur** on nii müügi kui ka kliendi hoidmisega tegelev inimene. **QA projektijuht** on tehniliste teadmistega inimene, kes aitab klienti testitsüklite koostamisel ja nende läbiviimisel. Kui kliendiga suhtlevad mõlemad, siis testijate kogukonnaga suhtleb QA projektijuht, omavahelises suhtluses liigub suurem osa infost kliendihaldurilt QA projektijuhile.

Testijate kogukonnas toimub suur osa tehnilisest tööst ehk testimisest. Kogukond ise koosneb vabakutselistest **testijatest** (ingl *tester*) ja **testitsükli juhtidest** (ingl *test lead*). Testitsükli juhtide ülesanne on kutsuda vajalik hulk sobivaid testijaid testitsüklisse ning hoolitseda selle eest, et kliendile saadetav raport oleks asjakohane ning korrektne. Praeguse seisuga on kogukond kinnine, kuid sinna on võimalik kandideerida Testlio veebilehelt või liituda kutse alusel. Selleks tuleb kogukonnas oleval testijal uut testijat soovitada. Testitsükkel koosneb mitmest etapist, mis leiavad aset nii Testlio keskkonnas kui platvormist väljaspool (vt Joonis 2).



Joonis 2. Testlio testitsükli ülevaade.

Testitsükkel (vt Joonis 2) saab alguse, kui klient saadab Testliole uue versiooni tarkvarast (ingl *new build)*, mida ta soovib testida. Koostöös QA projektijuhiga täpsustatakse vajadusel ülesande püstitus ja maht. Sellele järgnevalt planeeritakse testitsükli tegevuskava, mis koosneb testitsükli ajalisest kestvusest, testitavate rakenduse osade ja kasutavate vahendite loendist ning vajadusel ka testijate asukoha määratlusest. Planeerimine toimub juba vähemalt osaliselt Testlio platvormil. Sellele eelnevad tegevused toimuvad e-kirja või muu suhtlusviisi kaudu, mille klient on valinud.

Lisaks toimuvad platvormil ka teised otseselt testimisega seotud tegevused: testitsükli juhi määramine, testijate otsimine ja määramine, testimine ning tagasiside kogumine. Nendesse tegevustesse on peamiselt kaasatud testitsükli juhid ja testijad ehk Testlio testijate kogukond. Viimased sammud toimuvad jälle platvormist eemal, nendeks on testimistulemuste saatmine kliendile koos leitud vigade loendiga (ingl *bug report*). Peale seda saab klient hakata leitud vigu parandama. Tihti saadab klient uuendatud tarkvara uuesti Testliole, et kontrollida kas kõik vead said parandatud ja ega uute arendustega ei ole rohkem vigu tekkinud. Selline

rotatsioon tsüklist toimub iga toote versiooni kohta ning on üheks osaks kliendi tootearendusest.

Testlio veebiplatvormile saavad sisse logida ainult registreeritud kasutajad. Hetkel on see mõeldud testijate kogukonna töö haldamiseks ja lihtsustamiseks. Testlio platvormil on erinevad kasutajagrupid:

- testijad;
- testitsükli juhid;
- QA projektijuhid;
- kliendipoolsed projektijuhid.

Vastavalt rollile on platvormil võimalikud mitmed kasutusstsenaariumid, mille tõttu kuvatakse kasutajatele erinevaid vaateid. Käesolevas töös keskendutakse testijatele ning nende kasutajakogemusele Testlio platvormil. Testlio peab enda testijate kogukonda väga oluliseks ja väärtustades nende aega ning mugavust on oluline, et platvormi kasutajakogemus toetab testijate töö tegemist efektiivselt. Selle tõttu peab platvorm olema testijatele mõeldes lihtsustatud, et nende töövoog oleks võimalikult vähe segatud antud platvormi iseärasustest. Sellepärast keskendub ka käesolev bakalaureusetöö just testijate kasutajakogemuse parendamisele. Sobiva hetke olemasoleva analüüsiks on andnud olukord, et prioriteediks on kerkinud platvormi ümberdisainimine. See annab ka hea võimaluse käesoleva bakalaureusetöö tulemuste praktiliseks rakendamiseks. Selleks, et analüüsida hetkeolukorda ning anda soovitusi muudatuseks on vaja mõista ka kasutajakogemust.

1.2 Kasutajakogemus ja selle disain

Kasutajakogemus (ingl *user experience*) on mõiste, millel leidub erinevaid seletusi. Hartson & Pyla (2012) on välja toonud, et kasutajakogemus on terviklik mõju kasutajale, mis tekib suhtlusest süsteemi, seadme või tootega. Selle alla kuuluvad kasutatavusest, kasulikkusest ning emotsioonidest tulenevad mõjud, mis tekivad kasutuse ajal ning sellele järgnevalt. “Suhtlema” on lai mõiste, mille alla kuuluvad nägemine, katsumine, süsteemist või tootest mõtlemine, selle imetlemine või välimuse hindamine enne kontakti (Hartson & Pyla, 2012).

Gube (2010) on aga kasutajakogemuse seletanud lahti järgmiselt: kasutajakogemus on see kuidas isik ennast tunneb, puutudes kokku süsteemiga. Süsteem võib olla veebileht,

veebirakendus või arvutitarkvara. Tänapäevases kontekstis on see mingisugune inimese ja arvuti vaheline suhtlus (ingl *human computer interaction*). Need, kes töötavad välja kasutajakogemust (kasutajakogemuse disainerid), uurivad ja hindavad, kuidas kasutaja tunnetab süsteemi, vaadeldes selliseid asju nagu kasutajamugavus, kasutaja süsteemi väärtuse tajumine, väärtuslikkus, süsteemi tõhusus ülesande täitmisel (Gube, 2010).

Eelmisi selgitusi koos vaadates saab öelda, et rääkides kasutajakogemusest on väga olulisel kohal inimese ja süsteemi vaheline suhtlus, kasutaja süsteemi tunnetus, toote kasulikkus ehk väärtuslikkus ja selle tõhusus. Näiteks veebilehe menüüd disainides on selleks mitmeid erinevaid võimalusi. Hetkel on disainis väga levinud burgermenüü ikoon, mis koosneb kolmes paralleelsest joonest, kuid sellele leidub ka alternatiive. Kasutades sõna "menüü" ikooni asemel suureneb klikkide arv menüüs (Sites For Profit, kuupäev puudub). Ehk see näitab, et kuigi väga populaarne on ühtmoodi kujutada mingit elementi, siis kasutades sama ülesande jaoks kasutajale sõbralikumat lahendust tõstab see veebilehe külastatavust.

Disaineri publik koosneb inimestest, kes saavad antud rakenduse disainist kasu. Publiku kogemus on sügavalt mõjutatud sellest, mida teab või ei tea antud disainer (Weinschenk, 2011). Seega on oluline, et disainer mõistab loodava rakenduse kasutajaid.

Hartson ja Pyla on oma raamatus käsitlenud kasutajakogemuse protsessi jaotades selle kaheks. Esimene osa on kasutajakogemuse elutsükkel enne disaini alustamist ja teine osa kasutajakogemuse disainimine. Enne disaini alustamist läbitakse järgmised tegevused (Hartson & Pyla, 2012):

- Empiiriline protsess ehk konteksti uurimine, mille käigus tehakse justkui eeluurimus ehk kogutakse kokku eelinfo koos kasutajatelt tuleneva infoga.
- Organiseerimise protsess ehk induktiivne protsess. Selle käigus tõlgendatakse kasutajate töötegevuse andmeid.
- Deduktiivne analüütiline protsess, mille käigus töötatakse välja vajadused ja nõuded disainitavale süsteemile.
- Süntees erinevatest disaini-teadlikkuse mudelitest. Siin koostatakse ülesande kirjeldused, stsenaariumid, töömudelid ja muu taoline.

Protsessi mõistmine on oluline selleks, et aru saada erinevatest võimalustest ja etappidest, läbi mille on võimalik kasutajakogemust muuta. Seal hulgas ka veaseisundeid, mis on üheks osaks kasutajakogemusest.

1.3 Veaseisundid

Bakalaureusetöö kontekstis mõjutavad kasutajakogemust väga palju veaseisundid, kuid enne mõiste lahti mõtestamist, tuleb vaadelda mõistet “viga”. See tekib siis kui rakendus ei soorita temalt oodatavat tegevust, näiteks (Google, kuupäev puudub):

- rakendus ei mõista kasutaja sisendit;
- rakendus jookseb kokku;
- kasutaja proovib käivitada vastuolulisi toiminguid samaaegselt.

Babich on öelnud, et iga viga, olenemata sellest, kes selle on põhjustanud, häirib kasutajat (Babich, 2016a). Sellest tuletades võib öelda, et antud kontekstis viga on ükskõik kelle või mille põhjustatud rakenduse seisak, mis häirib kasutaja tavapärast voogu rakenduse käsitlemisel.

Veaseisund (ka veaolukord) on vaade, mis näitab kui asjad ei lähe rakenduse jaoks plaanipäraselt (Babich, 2016b). Ehk veaseisundiks loetakse antud bakalaureusetöö kontekstis neid vaateid ja disainielemente, mis on nähtavad selle tõttu, et rakenduses on esinenud viga, hoolimata sellest, kelle poolt viga on põhjustatud. Selleks võib olla näiteks kliendi poolt valesti sisestatud andmed, interneti ühenduse katkestus, kuid ka rakenduse enda toimimisest tulenev viga.

Rakenduse veaseisundite disainimiseks tuleb olla tuttav antud rakenduse osadega, mis võiksid põhjustada kasutajale veateate näitamist. Nende teadmiste olemasolul tekib võimalus minimeerida vigade juhtumise ja veateadete näitamise tõenäosust. Kõige lihtsam on seda tõenäosust vähendada vigu ennetades (Babich, 2016b).

Seega selleks, et tagada kasutajale hea kogemus Testlio platvormil ei piisa ainult veaseisundite disainist. Parema kasutajakogemuse andmiseks tuleb mõelda ka nende ennetamisele ehk kuidas suunata kasutajaid nii, et juhtuks võimalikult vähe vigu.

Kui aga tekib olukord, kus ei ole võimalik veaseisundit täielikul ennetada, siis tuleb kindlasti tegeleda veaseisundite disainiga ning käsitleda vigu õigesti. Selleks on vajalikud järgmised sammud (Google, kuupäev puudub):

- tuleb selgelt kasutajale väljendada, mis juhtus või toimub, et kasutajale veaseisundit kuvatakse ehk igale veateatele tuleb kirjutada korralik ja üheselt arusaadav tekst, mis on ülevaatlik ning inimlik (Gregory, 2015);
- selgitada, mida saab kasutaja teha selleks, et olukord lahendada;
- säilitada nii palju kasutaja sisestatud sisendit kui võimalik.

Selleks, et käsitleda veaseisundeid õigesti on loodud Google'i poolt täpsed juhtnöörid, mida peab jälgima Android mobiilirakendustes. Bakalaureusetöö käigus tehakse neist kokkuvõte ning Google'i mobiilirakenduste juhtnöörid võetakse eeskujuks Testlio platvormi veaseisundite disainimisel. Testlio platvormi kasutajaliides sarnaneb mobiilirakenduse omaga - see annab eelise kui praegusest veebiplatvormist tehakse mobiilirakendus, sest siis ei ole vaja suurt ümberdisainimist. Sealhulgas pole siis ka vaja veateadete käsitlust uuesti põhjalikult üle vaadata. Seda nimetatakse eelkõige-mobiil (ingl *mobile-first*) lähenemiseks, mis on veebilehe või rakenduse disainimeetod, mille puhul eelkõige mõeldakse mobiilsele lahendusele ning seejärel laiendatakse neid vaateid suurematele ekraanidele (Gremillion, kuupäev puudub).

Google juhendab veaseisundeid mobiilirakendustes käsitlema kolmes järgnevas kategoorias, millest antakse täpsem ülevaade järgmistes peatükkides (Google, kuupäev puudub):

1. kasutaja poolt antud sisendis ilmnevad vead;
2. hoolimata kasutajast ilmnevad vead;
3. vastuolulistest toimingutest tulenevad vead.

1.3.1 Kasutaja poolt antud sisendis ilmnevad vead

Esimene kategooria on kasutaja poolt antud sisendis ilmnevad vead ja need jagunevad kaheks. Esiteks sisestusväljades esinevad veateated. Sellisel puhul võib üldjuhul veateateid kuvada kohe murekoha juures. Kui neid on enam kui kaks, siis võib seda teha ka rakenduse ülemises servas, eraldi tekstikastis väljatooduna. Üldised juhised, mida jälgida veateadete disainimisel, on toodud välja järgnevas tabelis (vt Tabel 1) (Google, kuupäev puudub).

Tabel 1. Kasutaja poolt antud sisendis ilmnevad vead (Google, kuupäev puudub)

Veateate nimetus	Veateate disain
Vormi saatmise valik	Vormi saatmise valik peab üldjuhul olema lubatud vaikimisi. Juhul kui toimub vormisisene väljade valideerimine ning veateated väljendavad väga konkreetselt, mida kasutajalt oodatakse, siis võib vormi saatmise nupu ka peita, kuni kogu sisestatud info vastab kehtestatud reeglitele.
Värvid	Veateadete värvid kajastavad loodava rakenduse identiteeti. Soovitatud on kasutada veateadete jaoks kontrastseid toone, näiteks soojemad punased ja oranžid.
Abitekstid ja veasõnumid	Sisendi väljale võib lisada abitekste enne, peale või samal ajal kui kasutaja on vormiväljaga suhtluses. Veasõnumit tuleb kasutajale näidata peale seda, kui kasutaja on andnud sisendi väljale ja teinud seda puudulikult või valesti. Iga väli ei vaja abiteksti ja/või veateadet, vormis tuleb sellised lisad minimeerida ainult vajalikeni.
Tähemärkide piirarv	Tähemärkide piirarv võib olla kasutajale kuvatud enne, peale või samal ajal kui ta kirjutab antud väljale sisendit. Pigem tuleb kasutajale näidata tähemärkide lugerit alles siis kui kasutaja hakkab limiidile lähenema.
Vastuolulised väärtused	Veateated, mis märgivad ära vastuolulised väärtused, näiteks sisestatud parool on liiga lühike või sisestatud e-posti aadress pole korrektne, tuleb kasutajale kuvada samal ajal, kui ta on väärtust sisestamas või kohe peale seda, kui ta on tegevuse lõpetanud.

Teine juhtum on see, kui vorm on jäetud pooleli - sellisel juhul tuleb kuvada nii rakenduse ülemisel serval asuv tekstikast koos vastava sisuga kui ka veaseisundi sõnum vastavate sisestusväljade juurde (Google, kuupäev puudub).

1.3.2 Hoolimata kasutajast ilmnevad vead

Teine kategooria on rakenduse vead, mis juhtuvad hoolimata kasutajast. Üldised eeskirjad, kuidas antud olukorda lahendada on toodud tabelis 2 (Google, kuupäev puudub).

Tabel 2. Rakenduse vead, mis juhtuvad hoolimata kasutajast (Google, kuupäev puudub)

Veateate nimetus	Veateate disain
Üldised rakendusesisesed vead	Kui tekib üldine rakendusesisene viga, siis esimese asjana peab rakendus näitama mingisugust indikaatorit (ikoon, animatsioon), et parasjagu toimub informatsiooni laadimine. Seda tuleb näidata kasutajale nii kaua kui talle suudetakse kuvada veateade. Kui rakenduses on mingid osad, mis vea tõttu pole kasutajale kättesaadavad, siis võib neid kuvada nagu nad oleksid rivist väljas. Näiteks sellekohane teade või logo toonide heledamaks muutmine.
Vead info laadimisel	Kui on toimunud viga info laadimisel, siis lisaks sõnumi kuvamisele peab kasutajal olema võimalus suhelda teiste rakenduse osadega edasi, mis on kättesaadavad.
Kui ühendus on probleemne või aeglane	Kui ühendus on probleemne või aeglane, siis peab kasutajal olema siiski tagatud maksimaalne võimalik ligipääs infole. Kui on võimalik, siis tuleb lisada kliendile link, mis abistab tal soovitud tegevus lõpuni viia.

1.3.3 Vastuolulistest toimingutest tulenevad vead.

Kolmas kategooria on vastuolulistest toimingutest tulenevad vead, mis esinevad siis, kui kasutaja soovib sooritada mitut erinevat konfliktset tegevust. Juhtnöörid, mida peab jälgima antud olukorras, on välja toodud tabelis 3 (Google, kuupäev puudub).

Tabel 3. Rakenduse vead, mis tulenevad vastuolulistest toimingutest (Google, kuupäev puudub)

Veateate nimetus	Veateate disain
Üldised vastuolulised toimingud	Üldiste vastuoluliste toimingute puhul tuleb kasutajale teada anda, mis veaseisundit põhjustab.
Olukord, kus klient soovib teha internetti vajavaid toiminguid, kuid tegelikult ühendus puudub	Kui klient soovib teha toiminguid internetis, kuid tegelikult ühendus puudub, siis tuleb lisada rakenduse osadele, mis pole kättesaadavad levist väljas olles, püsivad indikaatorid, et klient harjuks nendega võimalikult kiiresti.
Loa taotlemine	Kui rakendusel on vaja taotleda luba, et pääseda ligi lisainformatsioonile samas vahendis (näiteks mobiiltelefonis), siis peab loa taotlemine ja teemakohased veateated asuma rakenduse enda sees.

Kokkuvõttes on oluline, et rakendus käitub kasutaja suhtes vea olukorras empaatiliselt. Kasutajale peab andma piisavalt palju vabadust ja kasutajakogemust piirama võimalikult

vähe. Samas on kasutaja suunamine väga oluline selleks, et kasutajakogemus püsiks positiivne. Selle tõttu ei tasu ka kliendile negatiivse sisuga sõnumeid liiga vara kuvada, kui need veel asjakohased ei ole. Nendeks on näiteks vea oht või tähemärkide piirarvu piiri meelde tuletamine. Kui tekib olukord kus tuleb kasutajale kuvada mitu erinevat veateadet, näiteks kui vorm on jäetud pooleli ning täitmata on rohkem kui üks väli, siis tuleb silmas pidada olulist üldist soovitust, et veateated ja abitekstid tuleb minimeerida vaid vajalikeni. Nii ei eksitata kasutajat ega koormata üleliia disaini. Kõik veaseisundid tuleb luua nii, et need abistaksid kasutajat ning aitaksid soovitud tegevuse lõpuni viia.

2 Metoodika

Selles peatükis kirjeldatakse meetodeid, mida on kasutatud antud bakalaureusetöö tulemuse saavutamiseks, milleks on Testlio platvormi veateadete määratlemine ja hetkeolukorra parendamine.

Tarkvara testimine on igasugune tegevus, mille eesmärk on hinnata kindlat osa või võimekust rakenduses või süsteemis. Selle käigus tehakse kindlaks, kas rakendus, süsteem või selle kindel osa täidab talle omistatud tegevust koos soovitud tulemusega (Pan, 1999). Tarkvara testimist tuleb pigem vaadelda kui protsessi, mitte kui ühekordset tegevust. See protsess kestab terve tarkvara arenduse elutsükli jooksul (ISTQB Exam Certification, kuupäev puudub-c). Testimine jaguneb positiivseks ja negatiivseks testimiseks.

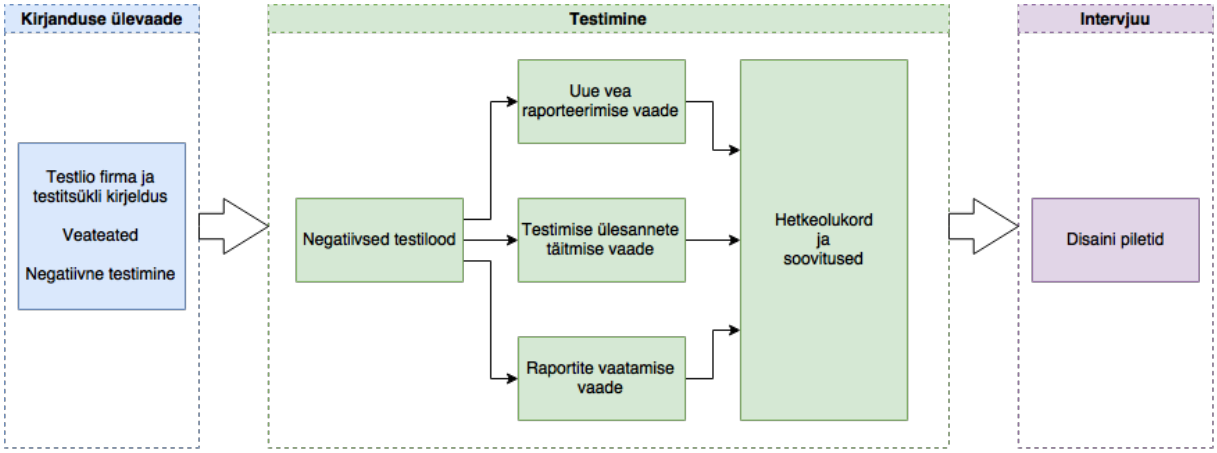
Antud töö eesmärkide saavutamiseks kasutatakse vahendina **negatiivset testimist**, mis näitab, kas tarkvara, süsteem või rakendus saab mõistlikult hakkama valesti sisestatud infoga või ootamatu kasutaja käitumisega. Näiteks kas peale valesti sisestatud telefoninumbrit oskab süsteem anda sellest kasutajale teada või lihtsalt keeldub vormis edasi liikumast. Põhiline vahe positiivse ja negatiivse testimise vahel on see, et negatiivse testimise puhul eeldame, et rakendus saab hakkama eranditega ning testime neid (Smartbear, kuupäev puudub).

Negatiivse testimise läbiviimiseks kasutatakse **negatiivseid stsenaariume**, mis on seotud Testlio veebirakenduses olevate väljade täitmisega. Negatiivsed stsenaariumid on hea moodus veaseisundite analüüsiks. Positiivsete stsenaariumitega testimise puhul ei pruugi leida kõiki testilugusid, mis aitaksid muuta tarkvara võimalikult kasutajasõbralikuks. Antud bakalaureusetöö analüüsist jäid positiivsed stsenaariumid välja, sest keskkond ise arendati lähtudes kasutajalugudest ning selleks, et tuleks välja rohkem kitsaskohti, keskenduti seekord negatiivsetele stsenaariumitele.

Töö tulemused tutvustatakse lõpuks Testlio tootetiimi disaineritele. Seda tehakse **fookusgrupi intervjuu** ehk rühmaintervjuu kaudu, mille eesmärk on uurida osalejate suhtumist ja arusaamist uuritavast teemast. Fookusgrupi intervjuu viiakse läbi väikese rühma inimestega konkreetsel teemal. Meetodi nimetus tuleneb sellest, et uurija ehk antud töö autor, juhib vestlust ja hoiab seda fookuses. Vestlus ise võib kesta poolest tunnist kuni kahe tunnini.

Intervjuu põhieesmärk on uurida seisukohtade moodustamist antud teema kohta grupi kontekstis (Laherand, 2008).

Kirjeldades bakalaureusetöö etappe (vt Joonis 3), saab välja tuua, et on tehtud eeluurimus kirjanduse ülevaate näol. Seal käsitletakse nii veateateid kui ka negatiivset testimist ja kirjeldatakse Testlio testimistsükli toimimist läbi rollide ja tegevuste. See paneb aluse praktilisele osale – testimisele. Töö käigus testitakse kolm vaadet, millele igaühele koostati testilood eraldi. Peale testimise läbiviimist, avaldatakse töös hetkeolukord ja soovitused parendamiseks, mis saadakse hetkeolukorra võrdlemisel Google'i veaseisundite juhistega. Peale seda viiakse läbi fookusgrupi intervjuu, mille käigus töö autor esitleb tulemusi Testlio disaineritele. Selle tulemusena selguvad olulised vead, mis lähevad platvormil ümbertegemisele.



Joonis 3. Etappide järjestus bakalaureusetöö vältel

3 Negatiivsed testilood Testlio platvormil

Selle peatüki eesmärk on määratleda, milliseid vaateid Testlio platvormil antud töö käigus käsitletakse, milliseid negatiivseid testilugusid saab neile rakendada ning millised väljad ja kuidas peaksid andmeid valideerima. Peale negatiivsete testilugude koostamist viiakse antud testilood ellu ning vaadatakse, kuidas erinevad väljad ja vaated toimivad. Peatüki lõpuks koostatakse kirjandusest lähtuvad soovitused, kuidas veaseisundeid Testlio platvormil parandada.

Negatiivsel testimisel saab olla palju stsenaariume. Antud töös lähtutakse kõige tavapärasematest stsenaariumitest (Smartbear, kuupäev puudub):

- Nõutud väljade täitmine (ingl *populating required fields*) – Rakendused võivad sisaldada väljade täitmist, mis on märgitud kohustuslikuks. Selleks, et seda testida tuleb kõigepealt jätta nõutud väljad tühjaks ja jälgida, kuidas rakendus sellele reageerib. Hiljem tuleb väljad täita ning hinnata, kas rakendus tunneb ära, et teisel korral on väljad täidetud.
- Lubatud arv tähemärke (ingl *allowed number of characters*) – Väljade täitmisel võib olla oluline faktor ka tähemärkide arv. Näiteks tavapärasele ette määratud piirarvuga väljadel tuleb kontrollida, mis juhtub kui sisestada nime välja rohkem kui 50 tähemärki.
- Lubatud andmete piirid ja piirangud (ingl a*llowed data bounds and limits*) – Selle testistsenaariumiga kontrollitakse väljad, mis aktsepteerivad teatud vahemikus olevaid andmeid. Seda saab hästi määrata numbreid sisaldavate väljadega kui arvestatakse aktsepteerituks teatud vahemikus olev väärtus (näiteks 10-50). Seda saab kasutada ka tekstiliste väärtuste puhul määrates sellele vajaliku pikkuse (näiteks 147 tähemärki). Testimist tuleks läbi viia nii, et sisestatakse mittevastuvõetavad andmed ja jälgitakse mis peale seda süsteemis toimub.
- Mõistlikud andmed (ingl *reasonable data*) – Mõned rakendused ja veebilehed sisaldavad väljasid, millel on küljes mõistlikkuse limiit. Näiteks ei saa vanuseks määrata negatiivset arvu ega liiga suurt arvu (nt 200).

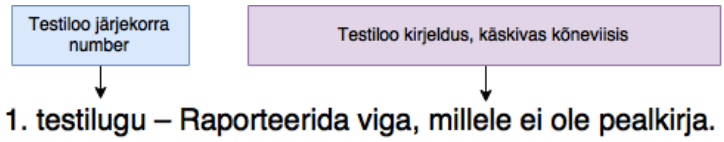
Töö käigus valitud vaated testimiseks lähtuvalt asjaolust.

et need on Testlio testijate jaoks olulised ning enim kasutatud. Nende vaadete testimine ja parendamine parandab testijatele töövoo kiirust Testlio platvormil ja edendab üleüldiselt platvormi kogemust.

Tabel 4 – Testilood keskkonna vaadete lõikes

Vaate nimetus	**Testilugude arv**	**Testilugude järjekorra numbrid**
Uue vea raporteerimine	15	1-15
Testimiste ülesannete täitmine	5	16-20
Raportite vaatamine	4	21-24

Bakalaureusetöö käigus loodi 24 testilugu (vt Lisa 1). Nende kirjeldamiseks kasutati käskivat kõneviisi ja testimisel pöörati tähelepanu kasutaja rollile – olles platvormil testija (vt Joonis 4). Seda on oluline defineerida, sest platvormil saab liikuda näiteks ka QA projektijuht, kuid tema fookus antud lehtedel on teine

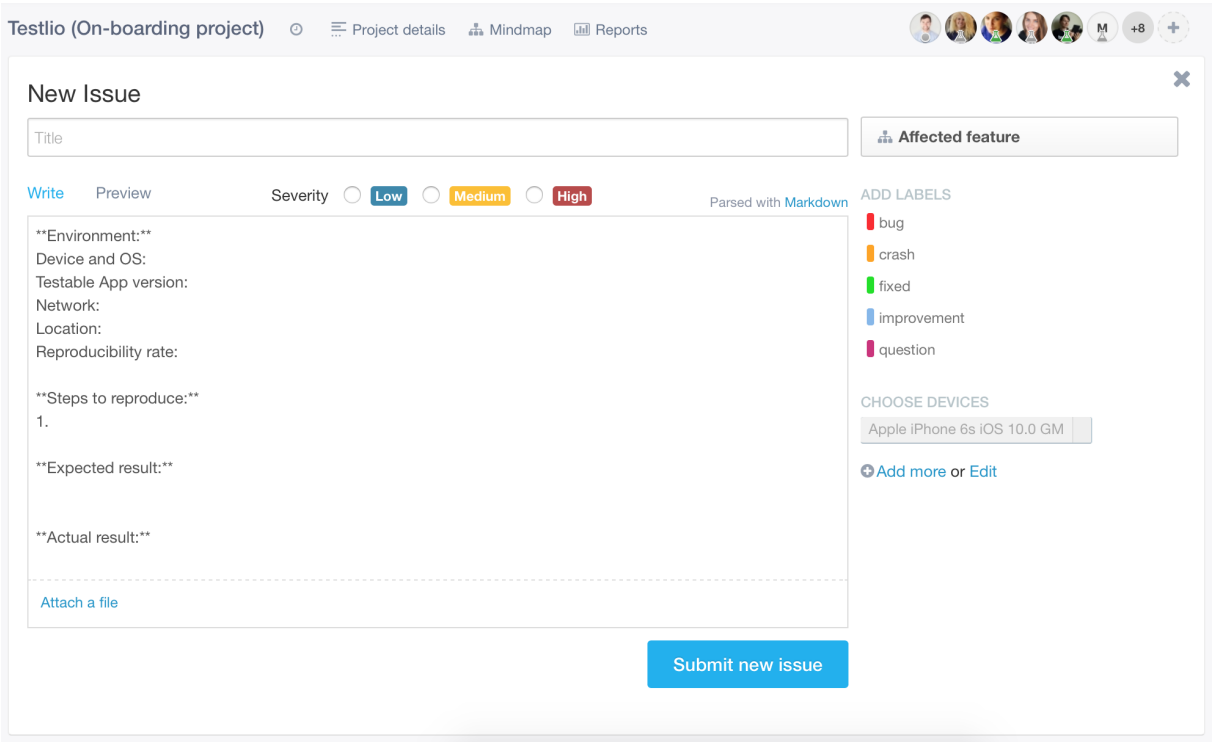


Joonis 4. Testiloo kirjelduse vorm

Järgnevalt kirjeldatakse lahti platvormi vaated, nende funktsionaalsus ja neile tehtud testilood.

3.1 Uue vea raporteerimise vaade

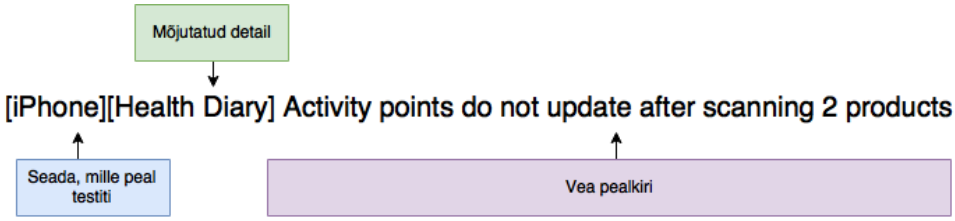
Uue vea raporteerimine (ingl *new issue*) (vt Joonis 5) on vaade, kus testija saab raporteerida kliendi rakenduses esinenud vigu. See on testija jaoks enim kasutatud vaade ja põhileht Testlio platvormil.



Joonis 5. Ekraanitõmmis testija poolt uue vea raporteerimise vaatest Testlio platvormil

Selles vaates esineb mitu välja: pealkiri (ingl *title*), mõjutatud detail (ingl *affected feature*), olulisus (ingl *severity*), vea kirjeldus (ingl *write*), failide lisamine (ingl *attach a file*), sildi lisamine (ingl *add labels*), seadmete valik (ingl *add device*) ja seadmete lisamine ja muutmine (ingl *add more or edit*). Sildi ja failide lisamine ei ole kohutuslik. Töös koostati negatiivsed testistsenaariumid kõikide ülalnimetatud väljade testimiseks.

Testlio keskkonnas sõltub vea pealkirja formaat projektist, ning see on kirjeldatud projekti ülevaate juures. See on määratud kas kliendipoolse sooviga või selliselt, et testijatel ning testtsükli juhtidel oleks lihtsam tuvastada topelt lisatud vearaporte (duplikaate). Vormi osas leidub erandeid, kuid tavaliselt koosneb testija seadme nimest, mõjutatud detaili kirjeldusest ja vea pealkirjast (vt Joonis 6).



Joonis 6. Vea pealkirja määramise valem

Koostati kaks testilugu:

1. testilugu – *Raporteerida viga, millele ei ole pealkirja.*

2. testilugu *– Sisestada pealkiri, mis on pikk (50 tähemärki).*

Mõjutatud detaili valides määrab testija, millist osa rakendusest raporteeritav viga mõjutab. Näiteks kas ta mõjutab sisselogimist, väljalogimist, maksete tegemist või mõnda muud vaadeldava rakenduse osa. Koostati kaks testilugu:

3. testilugu *– Raporteerida viga, millel ei ole mõjutatud detail määratud.*

4. testilugu *– Raportile määratakse rohkem kui üks mõjutatud detail.*

Testija saab valida leitud vea olulisuse taset. Tuleb teha valik, kas see viga on madal, keskmine või kõrge, vastavalt juhendile, mis avaneb kursoriga üle antud ala liikudes. Koostati kaks testilugu:

5. testilugu *– Raporteerida viga, millele ei ole lisatud selle olulisust.*

6. testilugu *– Ühele raportile märgitakse mitu olulisuse taset.*

Vea kirjelduse alla kirjutab testija tegeliku raporti. Esimesena tuleb kirjeldada keskkonda: millist seadet kasutati, millisesse võrku seade ühendatud oli, milline oli asukoht jms. Lisatakse juurde üksikasjalik kirjeldus, mis samme läbiti selleks, et viga esineks, milline oli oodatud tulemus ja kuidas rakendus tegelikult käitus. Koostati kaks testilugu:

7. testilugu *– Raporteerida viga, millel on eeltäidetud sisu ära kustutatud ehk tegemist on tühja väljaga.*

8. testilugu *– Raporteerida viga, jättes vea kirjelduse muutmata.*

Testijal on võimalus juurde lisada antud veakirjeldusele kokkujooksmisfaile (ingl *crash files*) ning videoid või ekraanitõmmiseid. See väli ei võta vastu suuremaid kui 100 megabaidiseid faile. Koostati kaks testilugu:

9. testilugu *– Lisada suuremahuline fail (üle 100MB).*

10. testilugu *– Lisada faili tüüp, mis ei ole seotud vea raporteerimisega.*

Testija saab lisada ka sildi, mis hiljem aitab kliendil või testitsükli juhil paremini sorteerida erinevaid tüüpi vearaporteid. Koostati üks testilugu:

11. testilugu *– Raporteerida viga, millel on rohkem kui üks silt.*

Testijal tuleb teha valik seadmetest, mida ta kasutas testimisel kirjeldatud vea leidmiseks. Koostati üks testilugu:

12. testilugu *– Raporteerida viga, millele ei ole lisatud seadet.*

Juhul kui testija on sooritanud testi seadmel, mida ta ei ole eelnevalt lisanud oma seadmete listi, siis seda saab teha ka viga raporteerides. Koostati üks testilugu:

13. testilugu *– Lisada tundmatu seade.*

Internetiühenduse katkemine võib olla suur probleem kui testija on sisestamas uut vearaportit, kui hiljem ühenduse taastamisel soovitakse antud raportit esitada. Koostati üks testilugu:

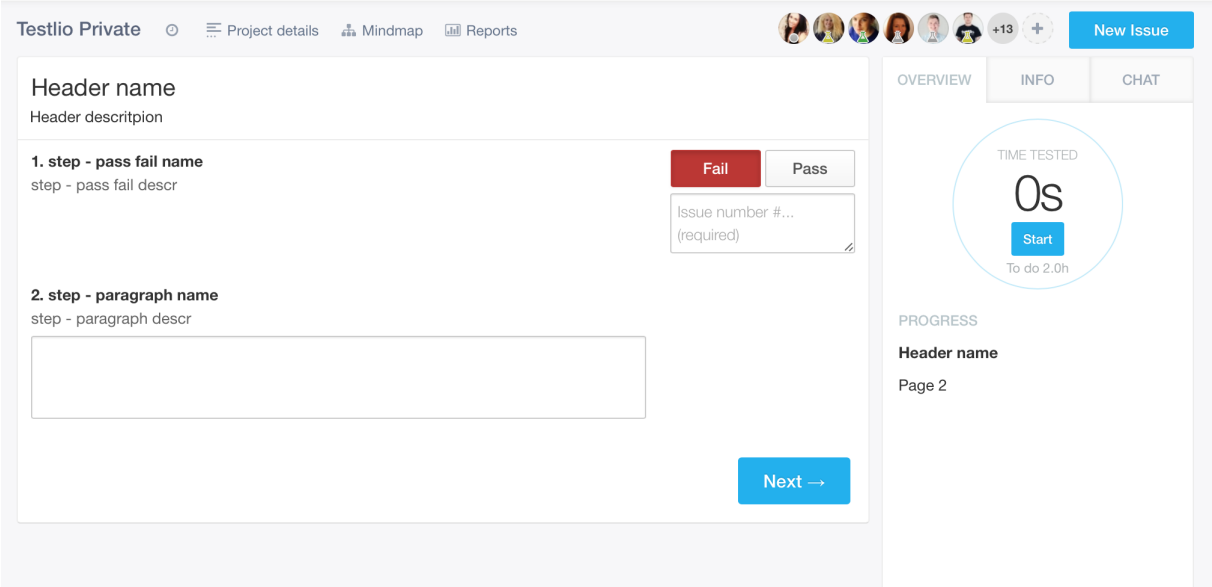
14. testilugu *– Raporti sisestamise ajal tuleb arvuti lahti ühendada internetiühendusest.*

Olukord, kus süsteem registreerib rohkem kui ühe kasutaja poolt sisestatud viga korraga, mis juhtub raportit sisestades. Koostati üks testilugu:

15. testilugu *– Raporteerida viga, millel on rohkem kui üks puudus. Näiteks puudub pealkiri ja määramata on jäänud olulisus.*

3.2 Määratud testimise ülesannete täitmise vaade

Määratud ülesannete täitmine (vt Joonis 7) käib Testlio platvormil eraldi vaates. Seda kasutavadki testijad selleks, et teada saada, mida antud testitsüklis testima peab.



Joonis 7. Ekraanitõmmis vaatest, milles testija täidab testimise ülesandeid Testlio platvormil

Lisaks sellele, et nad saavad vaatest kätte info, saavad testijad seal sisestada infot, millist tüüpi küsimused on testimiseks püstitatud. Küsimus saab olla kas jah või ei küsimus. Näiteks

kas antud osa rakendusest töötab või ei tööta. Teine võimalus on mõeldud avastamistestimisele, mis annab võimaluse testijale uurida mingit osa rakendusest ning hiljem siis kirja panna, mida ja kuidas uuriti. Koostati neli testilugu:

16. testilugu *– Liigutakse järgmise vaate juurde kui esimesele küsimusele ei määrata vastust.*

17. testilugu *– Liigutakse järgmise vaate juurde kui esimene ülesanne märgitakse läbikukkunuks, kuid juurde ei lisata kommentaari, mis vea kirjelduse number selle kohta käib.*

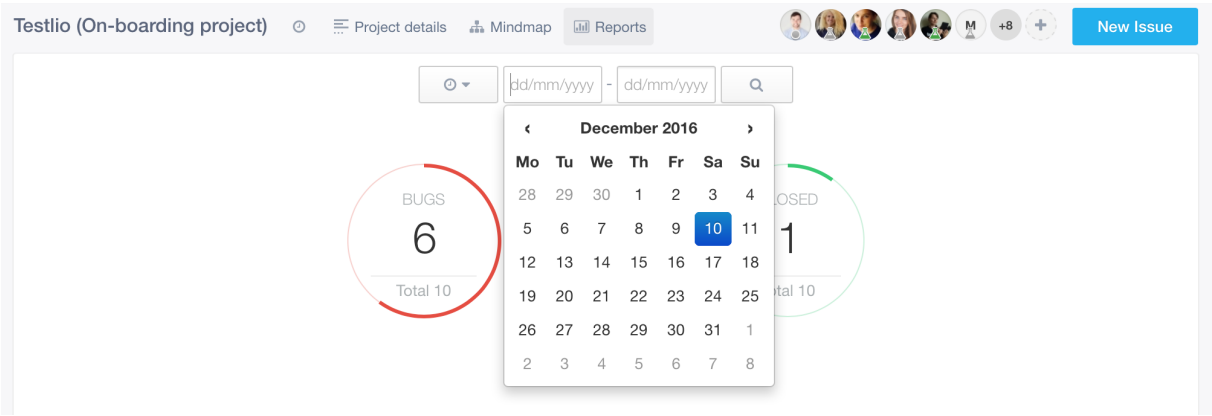
18. testilugu *– Liigutakse järgmise vaate juurde kui ülesanne märgitakse läbikukkunuks ning lisatakse juurde ka kommentaar, millise vea kirjeldusega see kokku kuulub, kuid ei märgita selle ette numbri tunnusmärki trelle (#).*

19. testilugu *– Liigutakse järgmise vaate juurde, kuid ei lisata juure teise sammu kokkuvõtet.*

20. testilugu *- Liigutakse järgmise vaate juurde kui pole eelnevalt täidetud ühtegi välja.*

3.3 Raportite vaatamise vaade

Raportite vaatamine on vaade (vt Joonis 8), mida saavad kasutada testijad selleks, et näha ülevaadet antud projekti kohta. Kuvatakse informatsioon selle kohta, mitu viga on üldse Testlio testijate poolt selles projektis leitud. Sealhulgas kuvatakse ka suletud ja endiselt avatud vigade arvu.



Joonis 8. Ekraanitõmmis raportite vaatest Testlio platvormil

Seal on ka näha ajagraafikut, millal on vigu leitud, milliste seadmetega testides on vead esinenud. Lisaks sellele on seal veel pingerida, kes on leidnud kõige rohkem vigu ning pingerida ka kõige enim mõjutatud detailidest. Neid kõiki saab vaadata erinevates ajavahemikes, mida kasutaja saab määrata. Ajavahemike määramiseks kasutatakse lihtsat kalendrivaadet. Koostati kolm testilugu:

21. testilugu *– Vaadata tulevikus asuvate kuupäevade vahelist perioodi.*

22. testilugu *– Valida kuupäevad, mille alguskuupäev on kalendriliselt eespool kui lõpukuupäev.*

23. testilugu *– jätta valimata alguskuupäev.*

24. testilugu *– jätta valimata lõppkuupäev.*

4 Testimine ja tulemused

Veebirakenduse testimine toimus eelpoolkirjeldatud negatiivsete testilugude järgi. Testimist viis läbi antud bakalaureusetöö autor. Kuigi keskkonnas testitakse mobiilirakendusi, siis platvormid ja brauserid, kus leitud vigu kirjeldatakse, on teised. Testimiseks kasutati Safari ja Chrome veebibrausereid, sest need on Testlio testijate ja töötajate seas kõige levinumad brauserid. Testimise tulemused jäädvustati testilugude kaupa kirjalikult ning on leitavad antud bakalaureusetöö lisadest (vt. Lisa 1). Järgnevalt on välja toodud testimise tulemused koondstatistikana (vt Tabel 5).

Tabel 5. Testimise tulemuste koondstatistika

Vaate nimetus	**Testilugude arv**	**Leitud vigade arv**	**Soovituste arv**
Uue vea raporteerimine	15	3	5
Testiülesannete täitmine	5	1	2
Raportite vaatamine	4	2	2

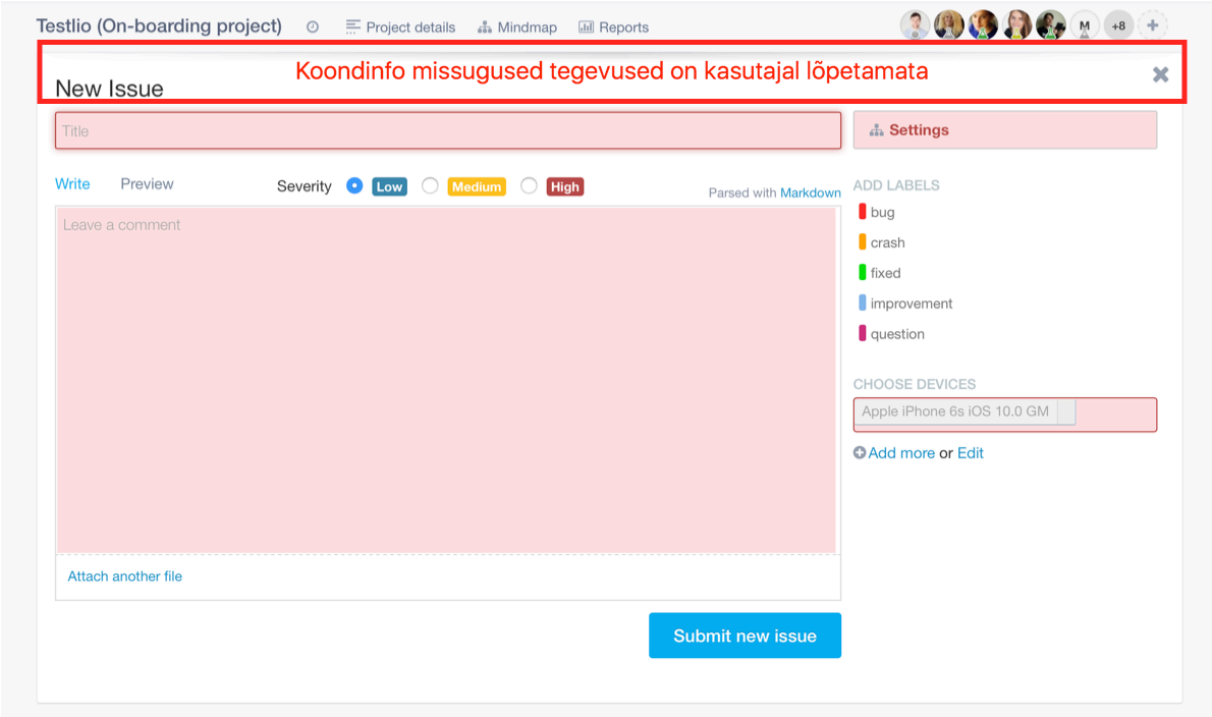
Esimene vaade, mida testiti, on mõeldud uue vea raporteerimiseks platvormil. Vaadet testides leiti nii hästi lahenduvaid stsenaariume kui ka stsenaariume, mida saaks parendada. Antud vaate veateated töötasid ja esinesid ühtlaselt ehk kõik olid disainitud ühetaoliselt, jälgides samasuguseid põhimõtteid. Järjepidevust disainis loetakse heaks kasutajakogemuseks. Lisaks kasutab vorm Google'i poolt soovitatud värvilahendust – kasutusel on soojades toonides punane, mis paistab keskkonnas hästi silma.

Küll aga ei kuvata antud vaates ühtegi abiteksti. Selline süsteemi käitumine on mõningal määral mõistetav, sest paljud vigade rapoteerijad on lehel mitmekordsed külastajad ning õpivad süsteemi käitumismustri selgeks. Küll aga võib süsteemiga harjumine olla uuele kasutajale seetõttu keerukam. Lisaks kuvatakse veatekste harva. Seda tehakse ainult kahel juhul kui kasutaja soovib üles laadida liiga suurt faili või kasutataval arvutil katkeb ühendus internetiga.

Kui juhtub aga, et kasutaja jätab vormi pooleli ning soovib edasi liikuda järgmisele vaatele, siis kuvatakse kõik erinevad veaseisundid ilma kommentaarideta ning jäetakse tegemata

kokkuvõte vormi ülaserva, mis on Google poolt soovitatav tegevus. Positiivne on see, et vaates ei kustutata kasutaja poolt sisestatud välju, vaid need jäetakse vormi alles, juhuks kui kasutaja soovib antud vormiga edasi tegeleda.

Lahendused, mida antud vaates võiks kaaluda on abitekstide ja veateadete kuvamine. See on oluline just esmakordsetele või süsteemi vähekasutanud külalistele. Lisaks tuleks mõelda sellele, kas lisada kokkuvõte veebilehe ülaserval kui vorm on jäetud pooleli (vt Joonis 9). See sisaldaks kokkuvõtet nendest lahtritest, mis on veel vajalik täita, et oleks võimalik antud viga salvestada.



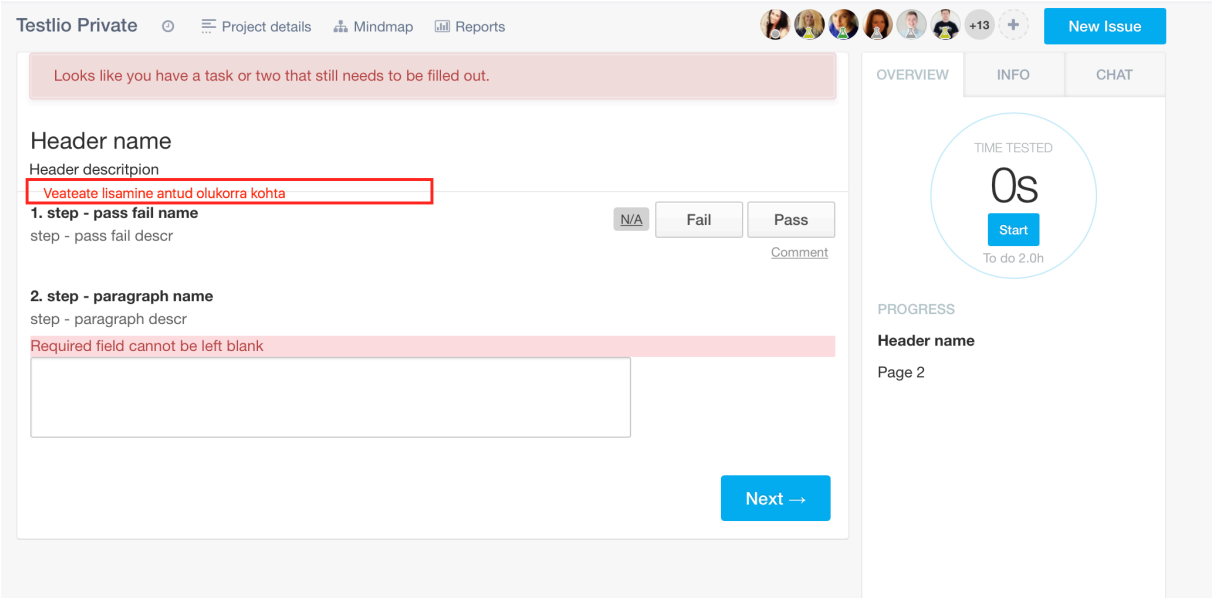
Joonis 9. Lõpetamata tegevuste koondkokkuvõte, kus punase ristkülikuga on märgitud ala koondinfo kuvamiseks.

Lisaks nendele muudatustele tuleks kaaluda kahe veaseisundi lisamist. Esiteks pealkirja pikkuse limiteerimine ja teiseks teade kui veakirjelduse sisusse pole tegelikult muudatusi tehtud. Esimene päästaks süsteemi liiga pikkadest pealkirjadest, mida ta niikuinii terviklikult hiljem ei kuva. Teisel juhul eemaldatakse oht, et keegi sisestab veakokkuvõtte ilma konkreetse sisuta.

Teine vaade, mida testiti oli testimisülesannete lahendamise vaade. Selle veaseisundid on konkreetsed ning ühtlaselt lahendatud üle vaate - veaseisundid esinevad koos veateadetega.

Ka abitekstid on olemas, juhendamaks testijaid antud vaates. Ainukene murekoht tekib siis kui lehel esineb rohkem kui üks viga. Selles olukorras kuvatakse kokkuvõttev sõnum lehe üleval, kuid esimese punkti juures jääb täpsemalt seletamata, mis seal puudu on. Teise punkti juures on antud tekst olemas.

Antud vaate juures võib kaaluda just ülalpoolkirjeldatud olukorra ühtlustamist. Lisada üldisele kokkuvõttele, mitu välja on puudulikud ning kuvata iga puuduliku välja juures veateade, mida kasutajalt oodatakse (vt Joonis 10).



Joonis 10. Veateate näitamine, et olukord ühtlustuks, kus punase ristkülikuga on märgitud veateate lisamise asukoht.

Lisaks võiks mõelda uuele veaseisundile. Nimelt võiks süsteem ära tunda, kas jah/ei vastuse juures nõutav veakirjelduse number on kindlasti sisestatud nõutavas vormis ehk algab kindlasti trellidega. See kindlustaks selle, et kasutaja teab kindlasti, mida ta tegema peab ning jätab ka vähem tööd testitsükli juhtidele, kes hiljem antud raporteid kontrollima peavad.

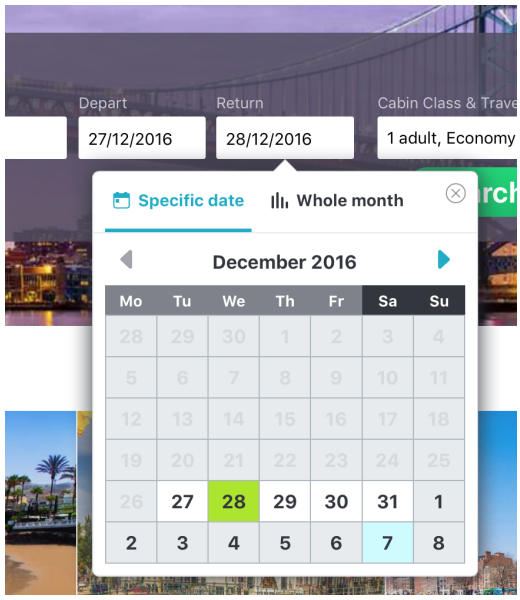
Kolmas ehk viimane vaade mida testiti, oli raportite vaatamine. Seal oli fookuses kuupäevade valimine kalendri kaudu. Vaates ei kuvatud ühtegi veaseisundit, isegi kui valitud olid ainult tulevikus asuvad kuupäevad. Ühest küljest saab seda lugeda heaks omaduseks, sest see näitab,

et süsteem on paindlik. Kuid teisest küljest ei näidata veateateid ka olukorras, mil nad oleksid vajalikud, sest sisu tegelikult päringule ei vasta.

Lahendus, mida antud vaate juures tuleb kaaluda on veaseisundite lisamine, et kasutajal ei oleks võimalik sooritada tegevust, millel tegelikult otstarvet ei ole (näiteks kuupäevade valimine ainult tulevikus). See oleks vajalik selleks, et ka kasutaja saaks aru, mida ta teeb valesti, et andmeid ei kuvata. Lahendusi on mitmeid, kuid mõelda võiks kindlasti järgmiste võimaluste peale:

- kui kalendrist valitakse alguskuupäevaks mingi kuupäev, siis lõppkuupäeva ei lasta valida varasemaks kui alguskuupäev;
- kalender ei lase valida tulevikus asuvaid kuupäevasid.

Selle jaoks kuvatakse mittesobilikud kuupäevad kalendris varjatud olekus näiteks heledamalt või tuhmimalt, et kasutaja saaks aru, et antud kuupäevasid ei saa kasutada. Näiteks nagu on seda tehtud SkyScanner[2] kodulehel kuupäevade valimise puhul (vt Joonis 11). Testilugude nr 23 ja 24 jaoks tuleks lisada hoiatussõnum, mis annaks kasutajale teada, et ta unustas valida alguse- või lõppkuupäeva ning seetõttu kuvatakse just teatud kuupäevade vaheline periood. Vajalik selleks, et kasutajal oleks ülevaade infost, mis talle kuvatakse.



Joonis 11. Ekraanitõmmis SkyScanner kodulehel asuva kuupäevade valimisest kalendrist.

[2] https://www.skyscanner.net

Kokkuvõtvalt saab öelda, et veateated on Testlio lehel kuvatud võrdlemisi stabiilselt ehk ühtemoodi. Leiduvad küll mõned olukorrad, kus juurde tuleks lisada abitekstid ja veateated, kuid üldpilt on hea. Lisaks puuduvatele abitekstidele ning veateadetele tuleks mõelda ka veaseisundite lisamistele, et rakenduse kasutamine oleks lihtsustatud ja arusaadavam kasutajale. Põhilised soovitused on:

- kuvada vormide täitmise juures abitekste;
- lisada veasõnumid puuduvatesse kohtadesse;
- lisada koondkokkuvõtted puudulikest väljadest kui vorm on jäetud pooleli;
- lisada veateadete olukordi rakendusele, et kasutaja ei mõistaks rakenduses toimuvat valesti.

Antud kokkuvõte viidi arutlemisele ka Testlio tootetiimile, mille tulemusena otsustatakse, millised vaated saavad bakalaureusetöös välja tulnud ettepanekute järgi täiendatud ja uuendatud, ning kas sellisel lähenemisel oleks ka potentsiaali Testlio keskkonna teiste vaadete jaoks.

5 Arutelu Testlio tootetiimiga

Arutlus viidi läbi antud töö autori poolt ja seal osales lisaks autorile 3 inimest Testlio tootetiimist, kaks kasutajakogemuse- ja tootedisainerit ning tootejuht. Ajaliselt kestis intervjuu 50 minutit ning selle jooksul tutvustas autor oma tööd ja selle tulemusi esitlemise teel. Selle jaoks jälgiti päevakava, mis oli järgmine:

1. bakalaureusetöö tutvustus;
2. töö käigus kasutatud metoodika tutvustus;
3. leitud puuduste tuvustus;
4. soovituste esitlemine;
5. küsimused ja vastused puuduste ja soovituste kohta;
6. tulevikuplaanide tegemine.

Selleks, et esitlus oleks ilmekam, vaadati puudused ja soovitused koos üle ühisel ekraanil Testlio platvormi kasutades.

Vigade väljatoomisesse ja soovitustesse suhtuti fookusgrupis soosivalt. Nimelt oli osalenutel hea meel, et olukorda oldi üksikasjalikult hinnatud ning pakuti välja ka lahendusi. Head vastukaja said peaaegu kõik soovitused, eriti hinnati abitekstide ja veateadete lisamise väljapakkumist. Näiteks viimaste puudumisest raportite vaates ei olnud osalejad teadlikudki. Lisaks arutati veel ka selle üle, et uue vea raporteerimise vaatel ei pruugigi kohe kasutajale selge olla, millised väljad on kasulikud. Seega otsustati ka see tulevaste tööde nimekirja lisada. Ainus lahendus, mis intervjuul sai kahetist tagasisidet oli pealkirja pikkuse limiteerimine. Selle puhul otsustati tulevikus uurida lähemalt Testlio erivaadete disaine, mida tähemärkide piiritlemine muuta võiks.

Koostatud 24-st testiloost kujunes lõpuks välja kaheksa soovitust, mille kaudu parendatakse platvormi kasutajakogemust:

- abi- ja veatekstide lisamine;
- veateadete kokkuvõtte lisamine uue raporti lisamise vaatele, juhul kui vorm on jäetud pooleli;
- veakirjelduse sisu mittetäiendamise puhul lisanduv veateade;
- märgistus, mis näitab, milline väli on kohustuslik;
- veateadete ühtlustamine testimise ülesannete täitmise vaates;

- veateate lisamine vea raporti numbri äratundmiseks;
- veateadete lisamine raportite vaatamise vaatesse;
- kalendri kuupäevavalikute loogika muutmine.

Et veaseisundite muudatustest ja arendusest oleks kõigile võimalikult palju kasu, siis võiks juba lisaks uuritule teha ka fookusgrupi uuring Testlio platvormi kasutajatega ning nendega läbi viia ka interneti vahendusel kasutatavuse testid peale prototüüpide valmimist. Need viiakse läbi tulevikus nii töö autori kui ka Testlio disainerite poolt. Lisaks lepiti kokku, et abi- ja veatekstide kirjutamisel võetakse appi professionaalne tekstikirjutaja, kelle emakeeleks on platvormil kasutatav inglise keel. Järeldati, et see aitab kõige paremini jõuda soovitud tulemuseni - selged ja üheselt mõistetavad sõnumid kasutajale.

Antud arutelu oli kokkuvõtlik ning andis uusi ideid mõlemale osapoolele. Ühest küljest võeti autori poolt pakutud uuendused hästi vastu ja teisest küljest lepiti koos kokku tulevastes sammudes, mida teha, et kasutusele võetavast lahendusest oleks kõigile osapooltele võimalikult palju kasu.

Kokkuvõte

Käesolev bakalaureusetöö lähtus probleemist, et Testlio platvormil puudub ühtne ning üheselt läbimõeldud veateadete süsteem, mis juhendaks ning hoiaks kasutajat informeerituna platvormi kasutades.

Töö eesmärk oli anda ülevaade veaseisunditest, luua negatiivsed testilood ning neid testida Testlio platvormil, et tuleksid välja platvormil esinevad puudused. Sellele lisaks koostati ka soovitused kitsaskohtade lahendamiseks. Selle eesmärgi saavutamiseks püstitati uurimusküsimused, millele vastuste leidmiseks anti ülevaade Testlio ettevõttest, seal toimuvate testitsüklite toimimist ja rollide vahelist tegevuste jaotumist. Lisaks uuriti veel veaseisundite määramist kasutajakogemusest lähtudes millised veateadete puudused esinevad Testlio platvormil. Töö tulemusena tehti soovitused hetkeolukorra parendamiseks.

Eesmärkideni jõudmiseks loodi töö käigus kokku 24 negatiivset testilugu, mille käigus uuriti kolme Testlio vaadet:

1. uue vea raporteerimine;
2. testimise ülesannete täitmine;
3. raportite vaatamine.

Töös on kasutatud negatiivset testimist, sest platvorm ise on loodud positiivset testimist kasutades, mistõttu annab negatiivne testimine uue lähenemise ning võimaluse märgata ka selliseid veakohti, mis rakenduse loomise ajal jäid varjatuks. Peale testilugude täitmist koostati soovitused tuginedes kasutajakogemuse headele tavadele. Kokku leiti testimise käigus kuus viga ning tehti üheksa soovitust, mille kaudu paraneks Testlio platvormi kasutajakogemus. Tulemusi analüüsiti arutelu käigus Testlio disaineritega ning ühiselt koostati tegevusplaan edaspidiseks. Töö tulemusena lähevad kaheksa töö käigus kujunenud soovitust tulevikus arendusse, et täiustada Testlio platvormi kasutajakogemust. Need soovitused on järgmised:

- abi- ja veatekstide lisamine;
- veateadete kokkuvõtte lisamine uue raporti lisamise vaatele, juhul kui vorm on jäetud pooleli;

- veakirjelduse sisu mittetäiendamise puhul lisanduv veateade;
- märgistus, mis näitab milline väli on kohustuslik;
- veateadete ühtlustamine testimise ülesannete täitmise vaates;
- veateate lisamine vea raporti numbri ära tundmiseks;
- veateadete lisamine raportite vaatamise vaatesse;
- kalendri kuupäevavalikute loogika muutmine.

Kindlasti ei saa lugeda antud töös käsitletud lähenemist ainuõigeks või lõplikuks. Ka arutelu käigus loodi plaan, kuidas edasi tegutseda, et antud tööst oleks võimalikult palju kasu kõikidele osapooltele. Selleks tuleb antud töö tulemusi analüüsida Testlio platvormi testijate ehk kasutajatega. Peale kindlate prototüüpide valmimist tuleks testijatega läbi viia ka kasutatavuse testid. Lisaks sellele tuleks abi otsida ka professionaalselt tekstikirjutajalt, et loodavad abi- ja veateated oleksid selged ja üheselt mõistetavad sõnumid.

Kasutatud kirjandus

Babich, N. (2016a). *Mobile UX Design: User Errors*. Loetud aadressil https://uxplanet.org/mobile-ux-design-user-errors

Babich, N. (2016b). *How To Design Error States For Mobile Apps.* Loetud aadressil: https://www.smashingmagazine.com/2016/09/how-to-design-error-states-for-mobile-apps/

Google (kuupäev puudub). *Google Guidelines – Errors*. Loetud aadressil https://material.google.com/patterns/errors.html

Gregory, S.C. (2015). *5 User Experience Tips for Creating the Best Error Messages*. Loetud aadressil http://freshsparks.com/user-experience-tips-best-error-messages/

Gremillion, B. (kuupäev puudub). *A Hands-On Guide to Mobile-First Responsive Design*. Loetud aadressil https://www.uxpin.com/studio/blog/a-hands-on-guide-to-mobile-first-design/

Gube, J. (2010). *What Is User Experience Design? Overview, Tools And Resources.* Smashing magazine. Loetud aadressil https://www.smashingmagazine.com/2010/10/what-is-user-experience-design-overview-tools-and-resources/

Hartson, R. & Pyla, P. S. (2012). Introduction. *The UX Book: Process and Guidelines for Ensuring a Quality User Experience* (lk 1- 47). Waltham: Morgan Kaufmann.

ISTQB Exam Certification. (kuupäev puudub-a). *What are the Software Development Life Cycle (SDLC) phases?* Loetud aadressil http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/

ISTQB Exam Certification. (kuupäev puudub-b). *Why is Testing necessary?* Loetud aadressil http://istqbexamcertification.com/why-is-testing-necessary/

ISTQB Exam Certification. (kuupäev puudub-c). *What is Software Testing?* Loetud aadressil http://istqbexamcertification.com/what-is-a-software-testing/

Kamps, H. (2016). Testlio raises $6.25M to expand its testing-as-a-service empire. Loetud aadressil https://techcrunch.com/2016/04/12/testlio-raises-6-25m-to-expand-its-testing-as-a-service-empire/

Laherand, M-L. (2008). *Kvalitatiivne uurimismeetod* (lk 219). Tallinn : M.-L. Laherand

Pan, J. (1999). *Software Testing*. Loetud aadressil https://users.ece.cmu.edu/~koopman/des_s99/sw_testing/

Sites For Profit. (kuupäev puudub). *Hamburger vs Menu: The Final AB Test.* Loetud aadressil http://sitesforprofit.com/menu-eats-hamburger

Smartbear. (kuupäev puudub). *Negative Testing.* Loetud aadressil https://smartbear.com/learn/automated-testing/negative-testing/

Testlio. (kuupäev puudub). *About Testlio*. Loetud aadressil https://testlio.com/about/

Thomson, C. (2014). *5 Reasons We Need Software Testing.* Loetud aadressil http://www.te52.com/testtalk/2014/08/07/5-reasons-we-need-software-testing/

Weinschenk, S. M. (2011). *100 Things Every Designer Needs to Know About People.* Berkeley: New Riders.

Äripäev. (2016). *Eesti idufirma kaasas 5,5 miljonit eurot.* Loetud aadressil http://www.aripaev.ee/uudised/2016/04/12/eesti-idufirma-kaasas-55-miljoni-eurot

Summary

Error States and Their Determination on Testlio's Platform

Bachelor's Thesis

This bachelor's thesis focuses on error states on Testlio's platform. The aim of this paper is to explore the possibilities to improve Testlio's platform's user experience through the re-designing error state. Testlio is an Estonian start-up founded in 2012. Its purpose is to be the best outsourced quality assurance partner for software developers to have a better mobile-app quality. This particular topic was chosen by the author due to the author's interest in user experience regarding error states after having problems trying to navigate on Testlio's platform as a tester.

To determine the current situation, an overview of Testlio is given and existing literature and guidelines on user experience, error states and negative testing is analysed. This set a base on how to use negative testing for evaluating the current situation. 24 different negative testing scenarios were created and executed to find the main complications in the platforms existing error states. In the end, six mistakes were found and nine recommendations given as three main views from the testers' point of view were explored. Lastly, a focus group interview was held to present the outcome of this thesis to Testlio's product team.

In the future almost all the recommendations will be added to the platform. However, some additional research and help from other Testlio's team members is needed. For example, a native speaker copywriter should be used, to set a clear and unambiguous message for every error state. Future research should include focus groups with Testlio's testers and usability testing with the main user group.

Lisa 1. Testimise tulemused testilugude järgi

Uue vea raporteerimise vaade

Testiloo number	Testilugu	Testiloo tulemus
1	Raporteerida viga, millele ei ole pealkirja.	Süsteem ei lase salvestada raportit enne kui on lisatud pealkiri. Veateadet näitab platvorm peale vormi esitamist. Juurde ei kuvata teadvustavat teksti, ainult pealkirja sisestamise kast muutub punaseks, mis viitab veale. Kui lisada veale pealkiri, siis peale seda on võimalik vea raport salvestada. Sealhulgas süsteem ei tee vahet, kas on sisestatud pealkiri veale korrektselt ehk jälgides nö. valemit või on seda tehtud valemit jälgimata.
2	Sisestada pealkiri, mis on pikk (50 tähemärki).	Süsteem laseb lisada pikka pealkirja (50 tähemärki).
3	Raporteerida viga, millel ei ole mõjutatud detail määratud.	Testija ei saa esitada raportit kui ta ei ole määranud mõjutatud detaili. Kui kasutaja üritab enne raportit salvestada kui see on lisatud, siis toimub taoline tegevus kui esimese testiloo ajal - kast läheb punaseks, kuid seletavat teksti ei ilmu juurde. Enne Salvestamis tegevuse üritamist ei kuvata kliendile ei indikaatorit, et midagi on valesti ega ka abiteksti.
4	Raportile määratakse rohkem kui üks mõjutatud detail.	Süsteem ei lase lisada korraga rohkem kui ühte mõjutatud detaili. Kui kõige pealt määrata esimene detail ning siis tahta lisada ka teine, salvestub ainult teisena valitud detail.
5	Raporteerida viga, millele ei ole lisatud selle olulisust.	Keskkond ei lase salvestada raportit enne kui olulisuse tase on määratud. Käitub taoliselt esimese ja kolmanda testilooga, kus puudub hoiatussõnum ning peale "esita" nupu vajutamist muutub kast punaseks.
6	Ühele raportile märgitakse mitu olulisuse taset.	Keskkond ei lase määrata ka mitut erinevat olulisuse taset, vaid salvestab viimasena valitud.
7	Raporteerida viga, millel on eeltäidetud sisu ära kustutatud ehk tegemist on tühja väljaga.	Kui jätta sisu väli tühjaks, ehk kustutatakse ära ka eelkirjutatud tekst, siis ei lase süsteem raporti salvestada. Veaseisund näeb visuaalselt välja samasugune nagu eelnevatel kordadel.
8	Raporteerida viga, kui vea kirjeldus jääb muutmata.	Kui jätta sisu väli puutumata, laseb süsteem selle salvestada. Ei näidata ühtegi veaseisundit.
9	Lisada suuremahuline fail (üle 100MB).	Kui lisada suurem fail kui 100MB, siis peale "esita" nupu vajutamist tuleb ekraani ülaosasse tekstikast tekstiga "Fail on liiga suur. Üleslaadimis limiit on 100MB".

Testiloo number	Testilugu	Testiloo tulemus
10	Lisada faili tüüp, mis ei ole seotud vea raporteerimisega.	Raportile sai lisada bdoc faili tüüpi ja seega ei limiteerita millist failitüüpi võib lisada ja millist mitte.
11	Raporteerida viga, millel on rohkem kui üks silt.	Raportile saab lisada enam kui ühe veatüübi. Sellisel juhul veaseisundit ei kuvata.
12	Raporteerida viga, millele ei ole lisatud seadet.	Selleks, et raportit esitada on vaja kindlasti määrata vähemalt üks seade. Ilma selleta näidatakse samasugust veaseisundit nagu eelmistel kordadel.
13	Lisada tundmatu seade.	Süsteem ei anna võimalust lisada tundmatut seadet, sest valiku saab teha eelnevalt arendajate poolt lisatud seadmetest. Annab tekstiliselt märku, et antud seadet ei leitud nimekirjast.
14	Raporti sisestamise ajal tuleb arvuti lahti ühendada internetiühendusest.	Kui ühendus kaob, siis Safari brauseris antakse sellest kasutajale märku lehekülje ülevale ilmuva tekstikastiga, millel on teade, et kasutaja ei ole internetiga ühendatud, ning tehtavad muudatused ei pruugi salvestuda. Chrome brauseris antud sõnumit ei tekkinud.
15	Raporteerida viga, millel on rohkem kui üks puudus	Kui testija teeb rohkem kui ühe vea korraga, siis kuvatakse kõik kastid korraga punaselt. Ei toimu eraldi vigade väljatoomist.

Testimise ülesannete täitmise vaade

Testiloo number	Testilugu	Testiloo tulemus
16	Liigutakse järgmise vaate juurde kui esimesele küsimusele ei määrata vastust.	Süsteem ei lase edasi liikuda kui on jäätud täitmata esimene ülesanne. Peab valima kas ülesanne sai täidetud või mitte ehk kas rakenduses esines viga või mitte. Tekib veateade, et antud ülesanne tuleb täita.
17	Liigutakse järgmise vaate juurde kui esimene ülesanne märgitakse läbikukkunuks, kuid juurde ei lisata kommentaari, mis vea kirjelduse number selle kohta käib.	Süsteem ei lase edasi liikuda järgmisesse vaatesse, kui ei ole lisatud kommentaar peale seda, kui antud ülesanne on valitud läbikukkunuks. Süsteem kuvab veatekstiga punast veateadet.
18	Liigutakse järgmise vaate juurde kui ülesanne märgitakse	Süsteem laseb läbi kommentaari, mis ei sisalda tegeliku numbrit koos trellidega. Veateadet ei kuvata

Testiloo number	Testilugu	Testiloo tulemus
	läbikukkunuks ning lisatakse juurde ka kommentaar, millise vea kirjeldusega see kokku kuulub, kuid ei märgita selle ette numbri tunnusmärki trelle (#).	
19	Liigutakse järgmise vaate juurde, kuid ei lisata juure teise sammu kokkuvõtet	Süsteem ei lase järgmise vaate juurde liikuda. Kuvatakse vastava infoga veateade.
20	Liigutakse järgmise vaate juurde kui pole eelnevalt täidetud ühtegi välja.	Juhul kui soovitakse edasi liikuda nii, et mõlemad veateated on tühjad, siis kuvatakse veateade üleval lehe ääres, mis annab kokkuvõtte kui mitu erinevat ülesannet on jäänud täitmata ning kuvab ka teise ülesande juures märget, et antud sisend on vajalik. Esimese juures veateadet ei näidata.

Raportite vaatamise vaade

Testiloo number	Testilugu	Testiloo tulemus
21	Vaadata tulevikus asuvate kuupäevade vahelist perioodi.	Kui otsida tulevikus asuvate kuupäevade vahelist jäävat ajaperioodi, siis platvorm laseb seda teha. Ei kuvata hoiatusi ega veateateid lihtsalt pole ühtegi tulemust, mida kuvada.
22	Valida kuupäevad, mille alguskuupäev on kalendriliselt eespool kui lõppkuupäev.	Kui üritatakse valida kuupäevad, mille alguskuupäev on kalendriliselt eespool kui lõpukuupäev, siis süsteem laseb seda teha. Nagu eelnevaltki, siis ka siin puhul ei näidata tulemust ega ka hoiatusi või veateateid.
23	Jätta valimata alguskuupäev,	Kui jäetakse valimata alguskuupäev, siis kuvatakse statistika kuni valitud lõpukuupäevani. Veateadet ega hoiatust ei kuvata.
24	Jätta valimata lõppkuupäev	Kui jäetakse valimata lõpukuupäev, siis kuvatakse statistika alates valitud alguskuupäevast. Veateadet ega hoiatust ei kuvata.